

TOOTLIST
CAHIER DES CHARGES

VERSION 1.0

20 DECEMBRE 2009

ING 3 SIGL

PAJOT Christophe

ROUCHES NICOLAS

SANTANGELI Yohann

ROGER Thomas

Table des matières

Notion	v
Front Office (FO)	v
Back Office (BO)	v
Ascendant (ASC)	v
Descendante (DESC)	v
Différent de (!=)	v
Liste	v
Item	v
Acteurs	v
Internaute	v
Utilisateur	v
Modérateur	vi
Administrateur	vi
Idée générale du projet	vii
Description de l’outil	vii
Besoin général	vii
Périmètre du projet	vii
Structure Générale du Projet	viii
Architecture	viii

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Technologies utilisées	viii
Serveur	viii
Serveur Web	ix
Site internet	ix
Base de données	xii
Client Lourd	xii
Analyses des tâches	xii
Tâches obligatoires	xii
Lots Obligatoires	xiv
Tâches Bonus	xv
Client Lourd	xvi
Etats des lieux des fonctionnalités au 14 Mai 2010	xvi
Tâches obligatoires	xvi
Lots Obligatoires	xviii
Tâches Bonus	xviii
Client Lourd	xix
Livrable	xx
Modélisation	xxi
UML	xxi
Diagramme de cas d'utilisation Internaute et Utilisateur	xxi
Diagramme de cas d'utilisation Modérateur et Administrateur	xxii
Base de données	xxiii
Modélisation	xxiii
Spécifications des tâches du site Internet	xxv

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Tâches obligatoires	xxv
Lots Obligatoires	xxxiii
Tâches Bonus	xxxix
Spécifications des tâches du client lourd	xliii
Documentation	xliv
Développement	xliv
Fichier de configuration	xlv
Scripts à exécuter	xlv
Documentation BO	xlv
Documentation FO	xlv
Installation	xlv
How To	xlv
Test	xlvi
Scénario 1	xlvi
Scénario 2	xlvi
Planning des tâches	xlvii
Spécification	xlvii
Architecture	xlvii
Développement	xlvii
Tests	xlvii
Coût	xlix
Relation Humaine	xlix
Hardware / Software	xlix
Serveur (Base de données et Hébergement)	xlix

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Site Internet	xlix
Client Lourd	xlix
Autre prestation	xlix
Analyse des risques	1
Technique	1
Personnel	1
Juridique	li

Notion

Front Office (FO)

Le Front Office concerne la partie du site à laquelle les internautes et les utilisateurs ont accès.

Back Office (BO)

Le Back Office concerne la partie de l'outil qui sera accessible par les modérateurs et les administrateurs.

Ascendant (ASC)

La liste sera triée de manière ascendante.

Descendante (DESC)

La liste sera triée de manière descendante.

Différent de (!=)

$A \neq B$

A est différent de B

Liste

Une liste est un regroupement d'objet de même type ou d'objet de type différent.

Item

Un item est un objet d'une liste quelconque

Acteurs

Internaute

Un internaute est une personne qui n'est pas connecté à l'outil.

Utilisateur

Un utilisateur est une personne qui est connecté à l'outil.

Modérateur

Un modérateur a accès au BO pour accepter la publication d'une liste publique, ou pour gérer le profil d'une personne.

Administrateur

Un administrateur possède un accès BO pour configurer l'application.

Idée générale du projet

Description de l'outil

Le projet "TOOTLIST" permet de lister tous types d'objets. Ces listes, une fois, constituées peut être complétées, mises à jour, organisées, triées, hiérarchisées, ... Cet outil sera entièrement personnalisable. Il sera consultable à partir de n'importe quel accès internet, ou d'une application portable.

Besoin général

Ce cahier des charges reprend les demandes et les spécificités de l'appel d'offre "Enonce de projet de Fin d'année : Outil "TOOTLIST".

Périmètre du projet

Site internet

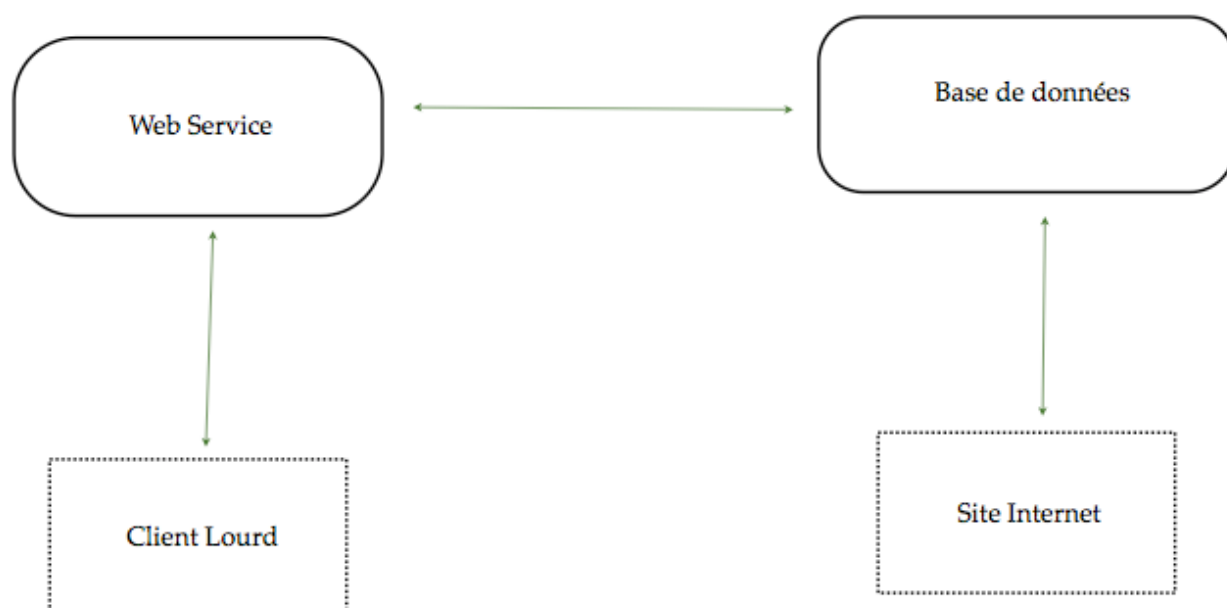
Le projet est basé sur la création de listes. L'objectif de l'application est de créer toutes sortes de listes. Ainsi un utilisateur pourra créer que des listes.

Client Lourd

Installé via un installateur sur un ordinateur, l'application devra gérer les listes principales, celle des lots obligatoires. On pourra rechercher de façon simplifier les listes et l'utilisateur verra les notifications.

Structure Générale du Projet

Architecture



Le client léger (Site internet) sera hébergé sur un serveur UNIX ayant comme serveur web Apache. Ce serveur possèdera une base de donnée MySQL. Le site Internet se reposera sur PHP.

Le client lourd accèdera à la base de donnée via un Webservice. Cette application pourra fonctionner en mode "off-line" et se synchronisera et synchronisera lorsqu'elle sera "on-line". Ces deux applications seront fait en JAVA.

Technologies utilisées

Serveur

UNIX

UNIX (ou Unix) est le nom d'un système d'exploitation multitâche et multi-utilisateur créé en 1969, conceptuellement ouvert et fondé sur une approche par laquelle il offre de nombreux petits outils chacun dotés d'une mission spécifique. Il a donné naissance à une famille de systèmes, dont les plus populaires en 2009 sont Linux, BSD et Mac OS X. On nomme "famille Unix" l'ensemble de ces systèmes. On dit encore qu'ils sont de "type

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Unix” et on les qualifie d’Unices. Il existe en 2008 un ensemble de standards réunis sous la norme POSIX qui vise à unifier certains aspects de leur fonctionnement. Le nom UNIX dérive de Unics, un jeu de mot avec Multics, car contrairement à ce dernier qui visait à offrir simultanément plusieurs services à un ensemble d'utilisateurs, le système initial de Kenneth Thompson se voulait moins ambitieux et utilisable par une seule personne à la fois avec des outils réalisant une seule tâche. Le système UNIX est multi-utilisateur et multitâche, il permet donc à un ordinateur mono ou multi-processeurs d'exécuter apparemment simultanément plusieurs programmes dans des zones protégées appartenant chacune à un utilisateur.

Serveur Web

Apache

Apache (ou Apache HTTP Serveur) est un logiciel de serveur HTTP qui est le plus populaire sur le Web. Apache est conçu pour prendre en charge de nombreux modules lui donnant des fonctionnalités supplémentaires : interprétation du langage Perl, PHP, Python et Ruby, serveur proxy, Common Gateway Interface, Server Side Includes, réécriture d'URL, négociation de contenu, protocoles de communication additionnels, etc. Néanmoins, il est à noter que l'existence de nombreux modules Apache complexifie la configuration du serveur web. En effet, les bonnes pratiques recommandent de ne charger que les modules utiles : de nombreuses failles de sécurité affectant uniquement les modules d'Apache sont régulièrement découvertes.

La possibilité de configuration d'Apache est un avantage non négligeable. Le principe repose sur une hiérarchie de fichiers de configuration, qui peuvent être gérés indépendamment. Cette caractéristique est notamment utile aux hébergeurs qui peuvent ainsi servir les sites de plusieurs clients à l'aide d'un seul serveur HTTP. Pour les clients, cette fonctionnalité est rendue visible par le fichier .htaccess.

Parmi les logiciels aidant la maintenance d'Apache, les fichiers de log peuvent s'analyser à l'aide de nombreux scripts et logiciels libres tels que AWStats, Webalizer ou W3Perl. Plusieurs interfaces graphiques facilitent la configuration du serveur.

Site internet

PHP 5.3 avec utilisation d’un framework :

_ Zend Framework

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Zend Framework est un framework PHP5 créé par Zend Technologies. Il est distribué sous la New BSD license. Zend Framework a été développé dans le but de simplifier le développement WEB tout en recommandant les bonnes pratiques et la conception orientée objet en offrant des outils puissants aux développeurs. ZF permet aussi nativement le principe de MVC (Modèle - Vue - Contrôleur). Ce framework est très flexible : en effet, l'ensemble des composants ne sont pas dépendants. De plus, ce framework est totalement personnalisable en fonction des besoins spécifiques de chaque projet grâce à un code simple et évolutif. Une très grande communauté est derrière ce framework, en commençant par Zend, lui-même. Le framework Zend utilise la programmation objet.

xHTML 1.0

eXtensible HyperText Markup Language est un langage de balisage servant à écrire des pages pour le World Wide Web. Conçu à l'origine comme le successeur d'HTML, XHTML se fonde sur la syntaxe définie par XML, plus récente, mais plus exigeante que celle définie par SGML sur laquelle repose HTML : il s'agit en effet de présenter un contenu affichable non seulement par les ordinateurs classiques, mais également sans trop de dégradation par des PDA beaucoup moins puissants.

CSS 2.0

CSS (Cascading Style Sheets : Feuille de style en cascade) sert à décrire la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). CSS est utilisé dans la conception des sites web et est à peu près bien supporté par les navigateurs web récents. Le principal atout du CSS est de regrouper tous les effets de style dans un document annexe, afin de séparer le contenu et la mise en page. Grâce à cela, on gagne en accessibilité, en mise à jour au niveau graphique d'un site et en réduction de la complexité d'un document. Par rapport au sujet de notre projet, on voit très clairement l'utilité d'utiliser le CSS. Dès les premiers navigateurs internet, on avait la notion de design, mais elle était mal incorporée, avec les feuilles de style, on voit la différence entre la structure et le contenu d'un document. La version 3 de CSS est en cours de développement.

Utilisation d'un framework Javascript :

_ Mootools

Javascript est un langage de programmation de scripts qui est utilisé principalement pour les pages web interactives. C'est un langage orienté objet à prototype, c'est-à-dire que les base du langage et ses principales interfaces sont des objets. Il a été crée en 1995 et en est à sa version 1.7. Le Javascript est exécuté sur le poste client grâce aux navigateurs web, cependant ces derniers ne sont pas très efficaces au niveau temps d'exécution. Javascript sert généralement à contrôler les données saisies dans un formulaire (interaction grâce au DOM), mais peut servir a réaliser des services dynamiques. Il est actuellement un des composants essentiels de la technologie AJAX (Asynchronous Javascript And XML). La plupart des applications AJAX utilisent l'objet XMLHttpRequest (XHR) pour envoyer une requête à un script serveur, et parser dynamiquement les résultats de ce dernier via le DOM. Il prend son envol vers les années 2003. AJAX est une technologie qui est utilisé dans les applications dites "WEB 2.0". Ce langage peut être simplifié dans sa syntaxe grâce à des bibliothèques Javascript. Ces bibliothèques permettent d'ajouter des fonctionnalités comme des transitions graphiques par exemple. Cependant, il n'existe pas que ces bibliothèques, il en existe plusieurs, tous ont des avantages et des inconvénients et correspondent à des besoins différents, en plus de l'appréciation du développeur.

Mootools appartient à ces bibliothèques, au même titre que Prototype et Scriptaculous. Cette bibliothèque est, certes, moins connue mais tout aussi efficace. Grâce à un ensemble de classes et de fonctions compatibles avec les navigateurs web les plus utilisés, Mootools offre une réponse aux problématiques du développement de Rich Internet Applications. L'ensemble de la bibliothèque est contenu dans un fichier Javascript unique. Lors du téléchargement à partir du site officiel (www.mootools.net), il est possible de spécifier les composants à inclure et de choisir le type de compression pour le code Javascript. Le poids du fichier est donc réduit au nécessaire, le transfert et le chargement de la bibliothèque dans les navigateurs sont accélérés. Mootools rassemble des fonctionnalités que l'on trouve dans Scriptaculous et Prototype, notamment plusieurs classes dédiées à l'AJAX, aux animations graphiques ou au drag and drop. Mootools bénéficie d'une communauté active qui a développé de nombreux plugins et qui assure le support de cette bibliothèque, disponible sous licence MIT. Les avantages de Mootools est le suivi de la programmation orienté objet, le composant

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

des effets avancés avec transition et enfin la très grande communauté. Tous les navigateurs récents supportent Mootools qui est actuellement en version 1.2.4.

Base de données

MySQL

MySQL est un système de gestion de base de données (SGBD). Selon le type d'application, sa licence est libre ou propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle ou Microsoft SQL Server. MySQL a été acheté le 16 janvier 2008 par Sun Microsystems pour un milliard de dollar. En 2009, Sun Microsystems a été acquis par Oracle Corporation, mettant entre les mains d'une même société les deux produits concurrents que sont Oracle et MySQL.

Client Lourd

Le client lourd sera fait en J2SE (Java 2 Standard Edition). J2SE est destiné aux applications pour poste de travail. CE framework contient toutes les bibliothèques de base, mais celle spécialisées pour le poste client (bibliothèque graphique, etc).

Le client lourd utilisera le framework Hibernate qui permet de gérer les persistances des objets en base de données relationnelles. Il est adaptable en terme d'architecture : il peut aussi bien être utilisé par les clients lourds ou par les interfaces web. Il propose le langage HQL qui intègre la prise en compte d'aspect objet comme les attributs des entités et le polymorphisme.

Le Web Service sera basé sur l'API de Webservice développé par Sun.

Analyses des tâches

Tâches obligatoires

Gestion des utilisateurs :

- _ Enregistrement
- _ Identification
- _ Privilège

- _ Aspect communautaire (Ajout d'ami)
- _ Quota : Nombre de listes par utilisateur

Back Office :

- _ Modération des utilisateurs (Rôle du modérateur)
- _ Modération des listes mises à jour (Rôle du modérateur)
- _ Configuration de l'application (Rôle de l'administrateur)

Gestion des Listes :

- _ Création de la liste
- _ Mise à jour de la liste
- _ Accessibilité des listes (private, protected, public)
- _ Liste = Modèle
- _ Duplication de la liste
- _ Versionning des listes (Modération des mises à jour)
- _ Vues des listes (Skins)
- _ Partager les listes via Facebook, Twitter, Mail

Aspect Graphique :

- _ Drag n Drop
- _ Identification
- _ Partie fixe : (Modèle de liste, liste les plus populaires, etc)
- _ Partie variable
- _ Modèle de base
- _ Skins variable (css)
- _ Modèle de page (Skins des listes)
- _ Design Clair et Fonctionnel
- _ Différenciation entre Back et Front Office

Moteur de Recherche :

- _ Recherche sur les listes
- _ Recherche sur les utilisateurs
- _ Recherche sur les contenus des listes
- _ Utilisation de filtres

Sécurisation :

- _ Inscription
- _ Service lié a l'utilisateur (Oublie de mot de passe)
- _ Identification de l'utilisateur sur le site
- _ Identification de l'administrateur sur le Back-Office
- _ Injections SQL

Autres fonctionnalités :

- _ Sitemap
- _ Aide
- _ Toute page est imprimable (Skin print)
- _ Lors d'un événement entre 2 utilisateurs, notion de notification (cf Facebook) + mail
- _ Mise en place d'un calendrier avec des types de calendriers existants

Mise en place d'un Client Lourd :

- _ Webservice
- _ Communication avec la base de donnée par le Webservice
- _ Stockage alternatif des données
- _ Gestion de la synchronisation des données (Exportation, Importation)
- _ Mise à jour des listes de l'utilisateur

Lots Obligatoires

Lot 1 :

- _ Liste d'envies/idées/résolutions
- _ Liste de courses
- _ Liste de tâches simples
- _ Répertoire téléphonique

Lot 2 :

- _ Liste de document de travail avec gestion de droit
- _ Liste de lecture en ligne avec lecteur embarqué sur le site

Lot 3 :

- _ Liste d'anniversaire ou événement avec un système de rappel et lié au calendrier
- _ Album Photo avec Zoom, légendes et animations

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

- _ Liste d'ami avec messagerie interne (mini-chat & envoi de message)
- _ Liste de Bookmarks avec Aperçu

Lot 4 :

- _ Liste de villes et vue de la météo de ceux-ci
- _ Liste de Pokémons avec un élevage de ceux-ci

Tâches Bonus

- _ Accès à Google Map :
 - _ Localisation une adresse
 - _ Voir l'itinéraire pour aller à une adresse
 - => Utilisé pour un événement, une adresse d'un ami, etc
- _ Internalisation :
 - _ Anglais
 - _ Français
- _ Si le lot 2 ne comprend pas un accès au document, Amélioration de ce lot en rajoutant un accès pour modifier ce document
- _ Export CSV, XML des données
- _ Sécurisation :
 - _ Message de confirmation
 - _ Captcha lors de l'inscription
- _ Gestion des encarts de publicité depuis le Back-Office
- _ Priorité des tâches
- _ AJAX
- _ Communication :
 - _ SMS
 - _ Mail
- _ Lecteur vidéo
- _ Sauvegarde
- _ Liste avec le statut protégé : Choisir les personnes qui peuvent y accéder
- _ Statistique pour l'administrateur

Client Lourd

- _ Utilisation d'un Webservice
- _ Stockage alternatif des informations
- _ Gestion de la synchronisation des listes
- _ Gestion des listes principales (Lots obligatoires)
- _ Gestion d'une recherche simplifiée
- _ Notification des évènements

Etats des lieux des fonctionnalités au 14 Mai 2010

Tâches obligatoires

Gestion des utilisateurs :

- _ Enregistrement [fait]
- _ Identification [fait]
- _ Privilège [fait]
- _ Aspect communautaire (Ajout d'ami) [fait]
- _ Quota : Nombre de listes par utilisateur [fait]

Back Office :

- _ Modération des utilisateurs (Rôle du modérateur) [fait]
- _ Modération des listes mises à jour (Rôle du modérateur) [fait]
- _ Configuration de l'application (Rôle de l'administrateur) [fait]

Gestion des Listes :

- _ Création de la liste [fait]
- _ Mise à jour de la liste [fait]
- _ Accessibilité des listes (private, protected, public) [fait]
- _ Liste = Modèle [fait]
- _ Duplication de la liste [fait]
- _ Versionning des listes (Modération des mises à jour) [fait]

- _ Vues des listes (Skins) [fait]
- _ Partager les listes via Facebook, Twitter, Mail [fait]

Aspect Graphique :

- _ Drag n Drop [fait]
- _ Identification [fait]
- _ Partie fixe : (Modèle de liste, liste les plus populaires, etc) [fait]
- _ Partie variable [fait]
- _ Modèle de base [fait]
- _ Skins variable (css) [fait]
- _ Modèle de page (Skins des listes) [fait]
- _ Design Clair et Fonctionnel [fait]
- _ Différenciation entre Back et Front Office [fait]

Moteur de Recherche :

- _ Recherche sur les listes [fait]
- _ Recherche sur les utilisateurs [fait]
- _ Recherche sur les contenus des listes [fait]
- _ Utilisation de filtres [non fait]

Sécurisation :

- _ Inscription [fait]
- _ Service lié a l'utilisateur (Oublie de mot de passe) [fait]
- _ Identification de l'utilisateur sur le site [fait]
- _ Identification de l'administrateur sur le Back-Office [fait]
- _ Injections SQL [fait]

Autres fonctionnalités :

- _ Sitemap [non fait]
- _ Aide [fait]
- _ Toute page est imprimable (Skin print) [fait]
- _ Lors d'un événement entre 2 utilisateurs, notion de notification (cf Facebook) + mail [fait]
- _ Mise en place d'un calendrier avec des types de calendriers existants [fait]

Mise en place d'un Client Lourd :

- _ Webservice [fait]
- _ Communication avec la base de donnée par le Webservice [fait]
- _ Stockage alternatif des données [fait]
- _ Gestion de la synchronisation des données (Exportation, Importation) [fait]
- _ Mise à jour des listes de l'utilisateur [fait]

Lots Obligatoires

Lot 1 :

- _ Liste d'envies/idées/résolutions [fait]
- _ Liste de courses [fait]
- _ Liste de tâches simples [fait]
- _ Répertoire téléphonique [fait]

Lot 2 :

- _ Liste de document de travail avec gestion de droit [fait]
- _ Liste de lecture en ligne avec lecteur embarqué sur le site [fait]

Lot 3 :

- _ Liste d'anniversaire ou événement avec un système de rappel et lié au calendrier [fait]
- _ Album Photo avec Zoom, légendes et animations [fait]
- _ Liste d'ami avec messagerie interne (mini-chat & envoi de message) [fait]
- _ Liste de Bookmarks avec Aperçu [fait]

Lot 4 :

- _ Liste de villes et vue de la météo de ceux-ci [fait]
- _ Liste de Pokémons avec un élevage de ceux-ci [fait]

Tâches Bonus

- _ Accès à Google Map :
 - _ Localisation une adresse [fait]
 - _ Voir l'itinéraire pour aller à une adresse [non fait]
 - => Utilisé pour un événement, une adresse d'un ami, etc [fait]
- _ Internalisation :

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

- _ Anglais [fait]
- _ Français [fait]
- _ Si le lot 2 ne comprend pas un accès au document, Amélioration de ce lot en rajoutant un accès pour modifier ce document [non fait]
- _ Export CSV, XML des données [fait]
- _ Sécurisation :
 - _ Message de confirmation [fait]
 - _ Captcha lors de l'inscription [fait]
- _ Gestion des encarts de publicité depuis le Back-Office [fait]
- _ Priorité des tâches [fait]
- _ AJAX [fait]
- _ Communication :
 - _ SMS [non fait]
 - _ Mail [fait]
- _ Lecteur vidéo [fait]
- _ Sauvegarde [fait]
- _ Liste avec le statut protégé : Choisir les personnes qui peuvent y accéder [non fait]
- _ Statistique pour l'administrateur [fait]

Client Lourd

- _ Utilisation d'un Webservice [fait]
- _ Stockage alternatif des informations [fait]
- _ Gestion de la synchronisation des listes [en fait]
- _ Gestion des listes principales (Lots obligatoires) [fait]
- _ Gestion d'une recherche simplifiée [fait]
- _ Notification des évènements [non fait]

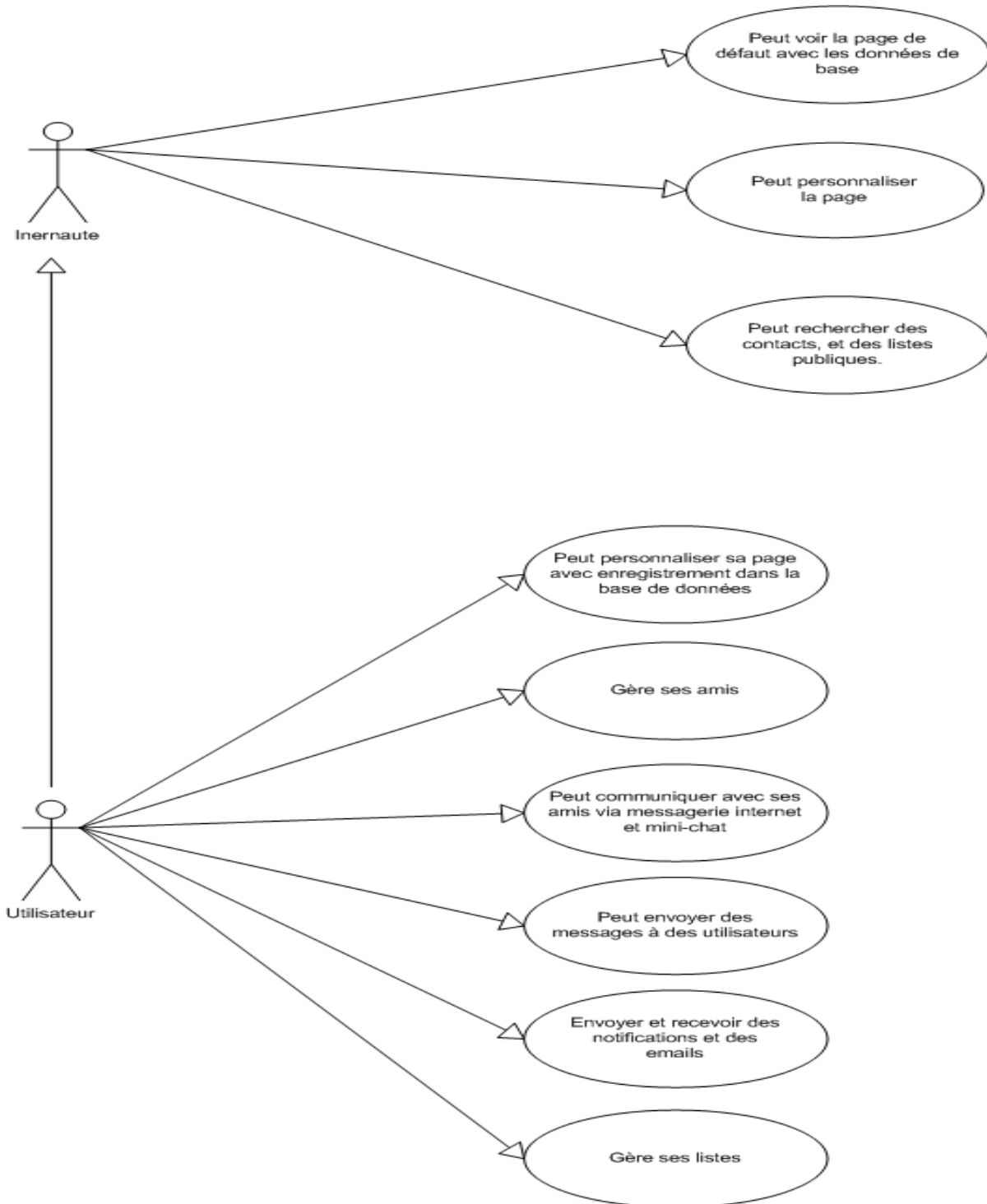
Livrable

Les livraisons seront définies par la mise en production des lots. Ces lots auront plusieurs versions qui s'intégreront à l'outil.

Modélisation

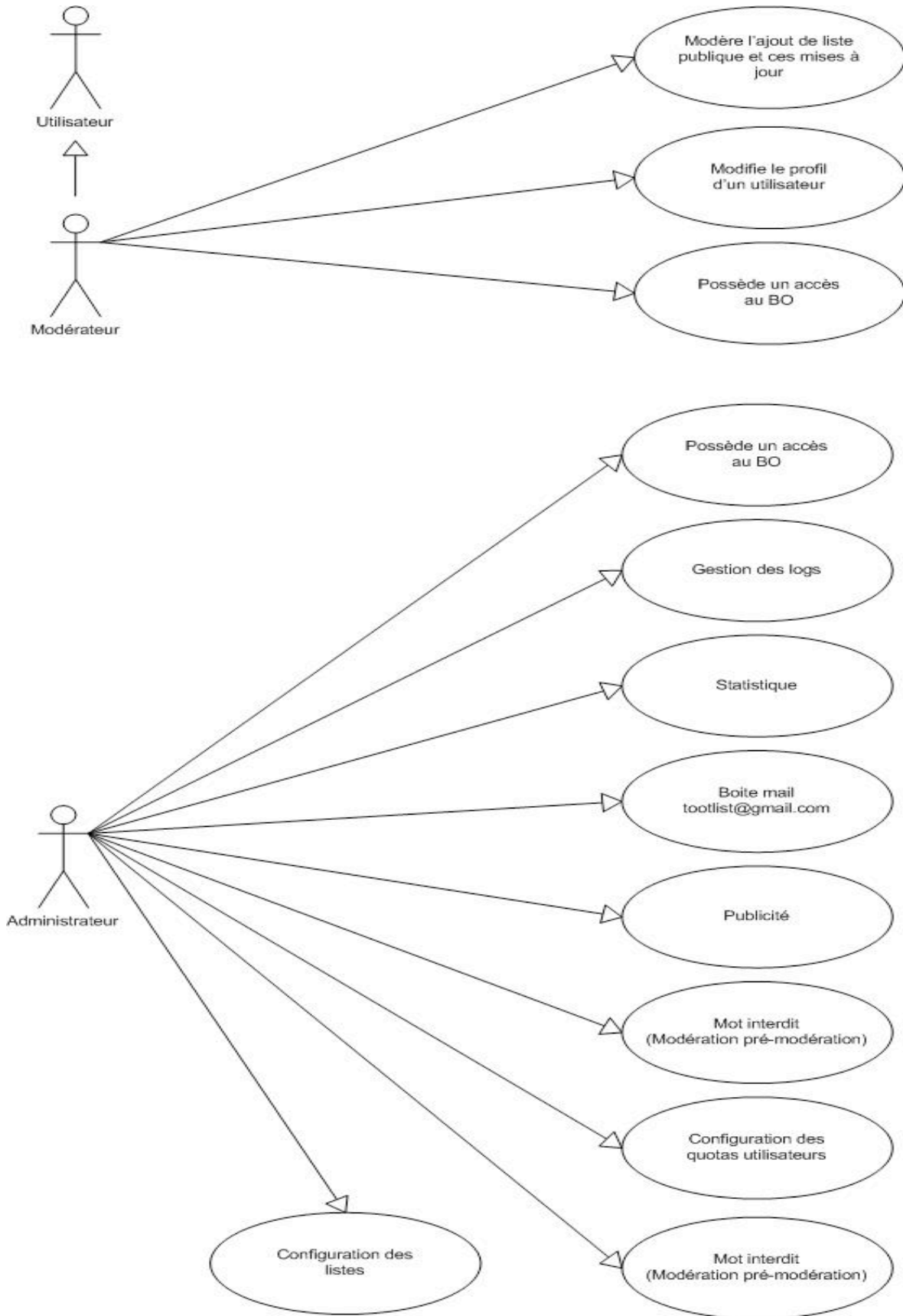
UML

Diagramme de cas d'utilisation Internaute et Utilisateur



SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

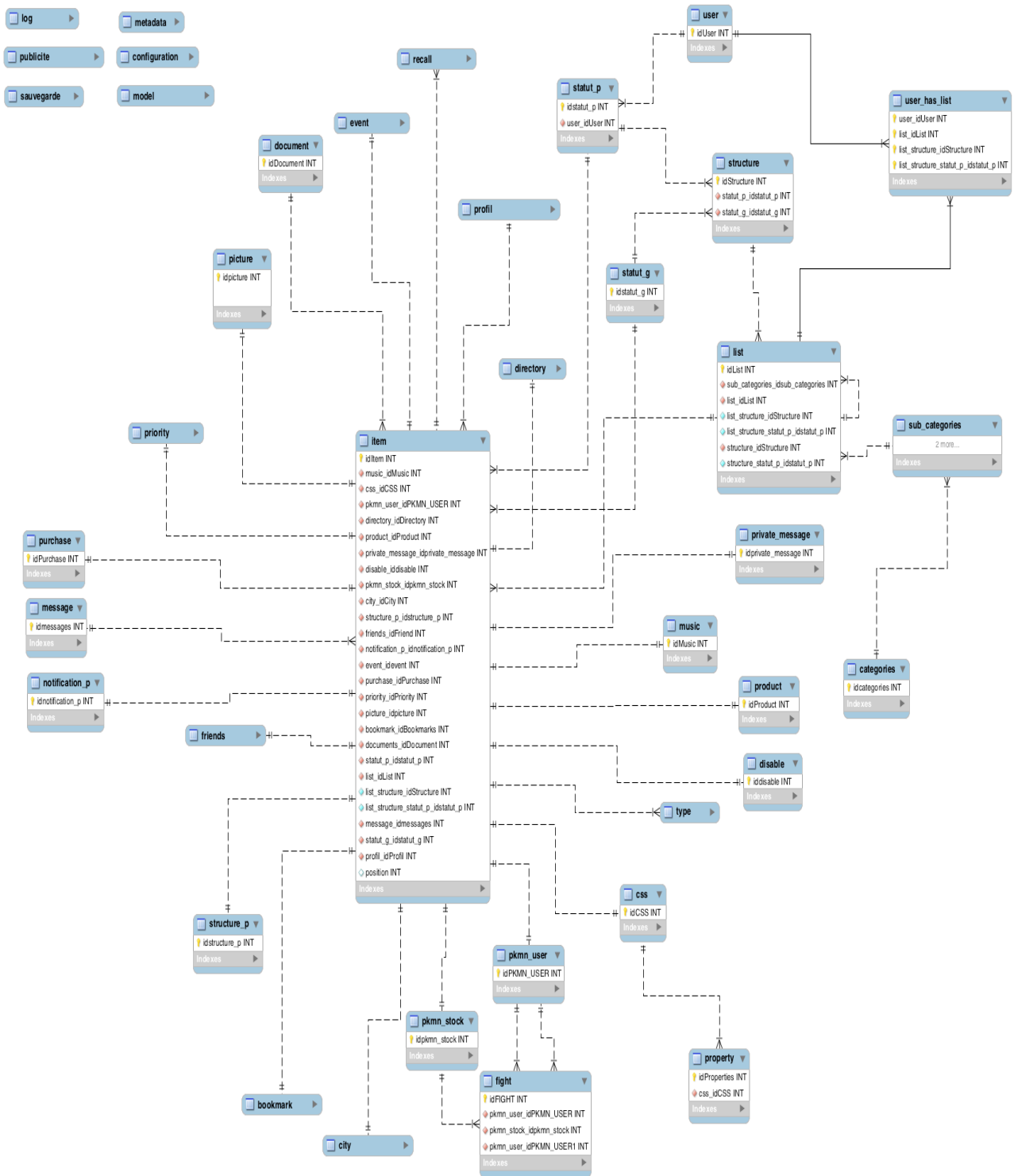
Diagramme de cas d'utilisation Modérateur et Administrateur



SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Base de données

Modélisation



SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Un utilisateur pourra créer N listes. Ces listes comporteront des items de types différents. Ainsi un item aura obligatoirement une entrée dans la tables Type qui identifiera son type. Ainsi grâce à son type, on pourra récupérer les informations spécifiques de cet item. Par exemple, un profil "Utilisateur" sera représenté par une liste d'item de type "profil".

Spécifications des tâches du site Internet

Tâches obligatoires

Gestion des utilisateurs :

L'utilisateur est un internaute qui s'est inscrit sur l'outil " TOOTLIST ". Il peut s'identifier grâce à son email et son mot de passe. L'utilisateur pourra alors configurer le style de toutes les pages en modifiant la couleur de fond de la page et des blocks, la police de caractères, la couleur des titres via la page de profils. Cette page recensera toutes ses informations. Dans cette page sera présente les informations confidentielles correspondant aux comptes Facebook et Twitter pour le partage de tout item d'une liste. Un utilisateur aura une liste d'ami : il pourra ajouter un ami, ce qui enverra une notification et un email au contact concerné. Il peut aussi accepter un ami, et lui donner accès à ses informations (Répertoire téléphonique, Anniversaire, etc). Ainsi, son nouvel ami pourra voir ses informations à partir de sa page « profil », ou depuis le répertoire téléphonique. Il peut rejeter une demande d'ami, supprimer un ami.

Au niveau communication, un utilisateur pourra envoyer un message à un utilisateur quelconque, mais ne pourra parler via le chat qu'à ses amis (Lot 3). Un utilisateur recevra une notification ou/et un email lorsque l'application ou un autre utilisateur intervient sur un objet auquel il est relié (appartenance, mise à jour, etc).

Afin de réguler la consommation des ressources de stockage de l'application, chaque utilisateur possédera un quota en nombre de liste, configurable depuis le BO par l'administrateur. L'administrateur pourra aussi rajouter des mots interdits afin de modérer en amont les contenus rentrés par l'utilisateur, avant la modération des modérateurs.

Internaute

- _ Peut voir la page de défaut avec les données de base.
- _ Peut personnaliser la page (Position des box, Skin de la page) avec enregistrement par Cookie
- _ Peut rechercher des contacts, et des listes publiques. Pour y accéder, il devra s'inscrire.

Utilisateur

- _ Peut faire la même chose qu'un internaute
- _ Peut personnaliser sa page avec enregistrement dans la base de données
- _ Gère ses amis

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

- _ Peut communiquer avec ses amis via messagerie internet et mini-chat
- _ Peut envoyer des messages à des utilisateurs
- _ Envoyer et recevoir des notifications et des emails (de manière involontaire (Application))
- _ Gère ses listes

Modérateurs

- _ Peut faire la même chose que l'utilisateur
- _ Modère l'ajout de liste publique et ces mises à jour
- _ Modifie le profil d'un utilisateur
- _ Possède un accès au BO

Administrateur

- _ Possède un accès au BO
- _ Gestion des logs
- _ Statistique
- _ Boite mail tootlist@gmail.com
- _ Publicité
- _ Configuration des quotas utilisateurs
- _ Mot interdit (Modération pré-modération)
- _ Configuration des lots 4 (Météo, Pokémon)
 - Nombre maximum de pokémon dans la liste principale
 - Algorithme des combats et des captures (Pourcentage)
 - Gestion des produits (Prix, caractéristique)
- _ Configuration des listes
 - Nombre d'item par page (Pagination)
 - Accès au liste pour les modifier

Back Office :

Deux types de personnes pourront avoir un accès BO. Ce sont les modérateurs et les administrateurs. Les modérateurs verront les listes publiques qui ne sont pas encore validées. Ils pourront donc voir le contenu de chaque liste et son design (Il aura une preview de la liste en quelque sorte). A partir de cette liste, il pourra ainsi valider et rendre publique la liste. Une notification sera alors envoyé à l'utilisateur pour lui dire que sa liste de-

vient publique. Pour la mise à jour d'une liste, le procédé est le même sauf que le modérateur pourra voir la liste originale pour voir les évolutions de celle-ci. Il pourra à tout moment envoyé un avertissement à un utilisateur au cas où la liste ne peut être publique. Si une mise à jour est rendue publique, alors l'utilisateur et l'utilisateur de la liste originale recevront une notification.

Un modérateur pourra aussi voir la liste des tous les utilisateurs. Il verra alors toutes ses informations. Il pourra ajouter des notes dessus qui pourront porter sur les avertissements par exemple. De plus, il pourra régénérer le mot de passe de chaque utilisateur. Il peut aussi le bannir ou le supprimer.

L'administrateur pourra faire exactement la même chose que le modérateur. Cependant, il aura accès à une configuration plus avancée par rapport à l'application. En effet, il pourra voir les logs de l'application qui concernera l'inscription d'un membre, la connexion d'un utilisateur, la création (mise à jour et la destruction) d'une liste, la demande d'ami (ou le rejet), l'ajout d'un événement dans un calendrier. En clair, il pourra voir ce que tous les utilisateurs font sur l'application.

Il devra aussi gérer les différents items d'une liste. En effet, il pourra rajouter des types d'items. Par exemple, pour les fichiers de type .mp3, l'application devra lancer le lecteur multimédia, pour un fichier de type .jpeg, elle devra le mettre dans une galerie photo, etc. De plus, il aura accès au statistique de l'application : Ces statistiques donneront des informations sur le nombre de connexions, le nombre d'inscription, le nombre de création de listes, les mots les plus recherchés en faisant la différence entre le site internet et le client lourd.

Une adresse email est à la disposition de l'administrateur, il pourra alors « checker » ses mails depuis le back-office de l'application. Cette adresse email servira de contact auprès des internautes, des utilisateurs et auprès des professionnels pour, par exemple, mettre de la publicité sur le site. Cette publicité sera un thème important du BO, car il pourra à tout moment choisir où une publicité sera sur l'application.

Un utilisateur aura un quota qui correspondra au nombre maximal de listes créés. Cette configuration sera modifiable par l'administrateur.

L'administrateur pourra aussi faciliter la modération des modérateurs en listant des mots qui seront interdits. Si dans la liste, le mot est trouvé, cette liste sera automatiquement rejetée par l'application. L'administrateur pourra alors choisir la "punition" pour l'utilisateur

en fonction du nombre et de la gravité des mots trouvés.

Il pourra en plus configurer l'application de manière générale, c'est-à-dire qu'il pourra choisir quel élément sera placé dans la partie fixe de la page d'accueil, le nombre d'item par page.

Un administrateur pourra rajouter un type simple. Il pourra donc insérer des types. Ce nouveau type devra être configuré pour que l'application puisse le gérer. Ainsi, le BO devra pouvoir gérer l'insertion de table dans la base de données.

Enfin, il pourra configurer à sa guise chaque lot de l'application : Pour le lot sur la liste des Pokémons (lot de type 4 obligatoire), il pourra donner le nombre maximum de pokémons dans la liste principale, gèrera l'algorithme des combats et des captures par le biais de pourcentage, et la gestion des produits dans la boutique (prix, caractéristiques, etc).

Notion d'accès d'une liste ou d'un modèle :

Il existe trois types d'accès pour une liste dans l'application :

_ Privée (ou private) : Seul l'auteur de la liste peut la voir, la modifier et la supprimer.

_ Protégée (ou protected) : L'auteur de la liste donne un accès en lecture seule à ses amis. Ainsi ses amis pourront voir la liste. Seul l'auteur peut la modifier et la supprimer.

_ Publique : Tous les utilisateurs de l'application pourront voir la liste. Seul l'auteur peut la modifier et la supprimer.

Pour les accès protégés et publics, une modération aura lieu au préalable afin d'éviter tout débordement dans le contenu.

Pour l'accès protégé, une évolution de l'application permettrait de donner des accès spécifiques à chaque ami. En effet, l'auteur de la liste pourra donner à un accès (lecture seule) à sa liste pour un ami A et pourra refuser l'accès pour un ami B.

La structure d'une liste sera alors un modèle.

Il existe trois types d'accès pour la structure d'une liste dans l'application :

_ Privée : Seul l'auteur de la liste peut la voir, la modifier et la supprimer.

_ Protégée (ou protected) : L'auteur de la liste donne un accès en lecture/écriture à ses amis. Ainsi tous ses amis peuvent reprendre la structure de sa liste pour la modifier par exemple.

_ Publique : Tous les utilisateurs de l'application peuvent voir et modifier la struc-

ture de la liste.

Une nouvelle structure de liste sera d'abord modérée par un modérateur avant d'éviter tous débordements. De plus, pour chaque nouvelle mise à jour de liste, nous pourrons voir la liste originale.

Gestion des Listes :

Tout internaute connecté pourra créer une liste. Une liste est un regroupement d'item de types différents. En effet, une liste pourra regrouper des musiques, des photos ou encore des items simples comme des chaînes de caractères. Les différents types d'items se verront évolutifs, en effet l'administrateur pourra en rajouter et préciser ce que l'application devra faire avec ce type. Ces listes s'appelleront les listes génériques. En plus de cela, des listes de type défini seront aussi présentes (voir les différents lots). Pour chaque item d'une liste générique, si l'item correspond à un type d'une liste spécifique (liste des lots), il devra avoir le même comportement que les items de celle-ci. Une liste sera donc rattaché à un utilisateur, et sera défini par un titre, une description, des tags, une catégorie, une sous-catégorie et surtout un ensemble d'items. L'utilisateur pourra personnaliser la structure de la liste et surtout le visuel de celle-ci en modifiant par exemple la couleur des items, ou la disposition des éléments dans la page. Avant de finaliser la création d'une liste, il devra spécifier les accès de celle-ci (Publique, Protégée, Privée) pour le contenu de la liste (lecture seule), et la structure ou appeler « Modèle » de la liste (lecture/écriture). A partir de l'accès protégé, la publication de la liste se fera après validation d'un modérateur. Dans le cadre d'une mise à jour, une notification sera alors envoyée à l'auteur d'origine qui lui expliquera qu'un utilisateur l'a pris comme modèle. Lors d'un refus d'un modérateur de rendre accessible la liste, une notification sera envoyée au propriétaire afin de lui annoncer pourquoi, il sera aussi averti de sa sanction. Avant de créer une nouvelle liste, il pourra rechercher si une liste correspond à ses attentes. Si c'est le cas, il pourra donc la dupliquer et mettre à jour la liste pour qu'elle corresponde exactement à ses attentes. De plus, sur cette liste, nous aurons un accès à la liste originelle. Ainsi pour chaque liste, nous aurons plusieurs versions d'une même liste. Chaque utilisateur qui utilise une version de cette liste pourra donc voir toutes les versions de sa liste. La structure d'une liste concerne son squelette mais aussi son visuel. Donc un utilisateur pourra modifier le squelette de façon simple tel que l'alignement des informations, la police, la couleur du contenu des différentes

balises. Enfin, l'utilisateur pourra partager ses listes via le site communautaire Facebook, Twitter. Il pourra aussi envoyer ses listes via e-mail.

Aspect Graphique :

Le drag N drop (ou cliquer déplacer) est dans un environnement graphique, une méthode consistant à utiliser la souris pour déplacer d'un endroit à un autre un élément. Pour ce projet, le drag N drop est une fonctionnalité majeure. Il permet sur la page d'accueil de personnaliser cette page, en modifier la structure visuelle. De plus, on pourra gérer le classement des items dans une liste donnée grâce à cette méthode. Là où il sera utilisé, il devra être mis en place. Techniquement, le drag N drop sera fait grâce au langage Javascript.

L'Identification est présentée sous forme de boîte accessible sur chaque page si l'utilisateur n'est pas connecté. Elle se fait grâce à un email et un mot de passe. À partir de cette boîte, l'internaute pourra s'inscrire. Il devra donc rentrer les informations essentielles pour la création de son profil. Un email sera alors envoyé pour valider son inscription. Lorsqu'un utilisateur ne peut pas s'identifier car il a oublié son mot de passe, un mail lui sera alors envoyé avec dans son mail, un nouveau mot de passe que l'application aura régénéré.

Partie Fixe

La partie fixe sera représentée par des modèles de listes. Elle comportera les listes les plus populaires (c'est-à-dire, celles qui sont les plus vues), les modèles de listes les plus populaires (c'est-à-dire les listes qui sont le plus dupliquées) et les listes les plus récentes. Enfin nous aurons un accès rapide aux catégories qui représentent le plus de listes.

Partie variable

Cette partie est différente en fonction de la personne qui voit cette page. Dans le cas d'un internaute (personne non connectée), il aura accès aux données de base : une liste de recette, une liste de todo, une liste d'envie, et une liste dynamique qui aura des items de types différents afin de voir les possibilités de l'application TOOTLIST. Dans le cas d'un utilisateur (personne connectée), l'utilisateur verra l'ensemble de ces listes.

Skin variable

L'utilisateur, lors de la première connexion, aura un skin par défaut. Il pourra le transformer via son profil, afin de le changer. Il aura le choix entre divers skins qui seront des packages qui changera le design entier de la page. La personnalisation d'un skin sera impossible.

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Modèle de listes

L'utilisateur, lors de la création de la liste, verra un skin par défaut. Il pourra alors personnaliser sa page en fonction de divers critères. Il pourra modifier l'alignement (Centré, gauche, droite), la police (type de police : Verdana, Arial, Georgia) , la typologie (Souligné, Gras, Italique) et la couleur des textes de sa liste.

Design clair et fonctionnel

La design sera fait de façon à rendre accessible le site internet. Il devra aussi permettre une certaine ergonomie afin de ne pas perdre les différents utilisateurs sur l'application.

Différenciation entre BO et FO

La structure et les couleurs seront différentes entre le BO et le FO

Moteur de Recherche

Tout résultat donné par le moteur de recherche sera une liste de types différents. On pourra mélanger des listes avec des modèles et même des utilisateurs, ou tout simplement des items de listes. Ainsi, nous mettrons en place un surlignage du pattern recherche à la manière de la recherche sous Firefox.

_ Recherche sur les listes

La recherche devra être effectuée sur les titres, les catégories, les sous catégories, les mots-clés et la description de celles-ci. Cependant, la recherche gèrera les différents statuts des listes (Publique, Protégée, et Privée). Lors du survol sur le titre, nous pourrons avoir un aperçu de la liste via une boîte.

_ Recherche sur les modèles

La recherche devra être effectuée sur les types de skins (Couleur, utilisation des balises). Cependant, la recherche gèrera les différents statuts des modèles (Publique, Protégée, et Privée). Lors du survol sur le titre, nous pourrons avoir un aperçu du modèle via une boîte.

_ Recherche sur les utilisateurs

La recherche des utilisateurs devra permettre d'envoyer des messages ou de trouver des amis. Elle se fera en fonction de nom, prénom, ville, et même pays. On peut aussi rechercher des mails via les mails Yahoo, hotmail et Gmail.

_ Recherche sur les contenus des listes

Le moteur de recherche devra aussi rechercher dans le contenu des listes, c'est-à-dire dans le contenu des items, pour les faire ressortir.

_ Utilisation de filtres

Par défaut, la recherche s'effectuera sur l'ensemble des données publiques de l'application. On pourra accéder à la recherche avancée pour peaufiner la recherche grâce aux différents filtres proposées.

Sécurisation :

La validation de l'inscription de l'utilisateur se fait à partir d'un lien contenu dans l'email d'inscription. Seul ce lien pourra valider le compte et donnera accès à la partie « utilisateur » de l'application. Le service qui permet de générer le mot de passe en cas de perte envoie un email à l'adresse email du profil concerné. Ce service demande en entrée une adresse email et compare dans la base de donnée si cet email existe. Si oui, il envoie un email avec le nouveau mot de passe. L'identification de l'utilisateur sur le site se fait à partir d'un email et d'un mot de passe. Tous les mots de passe, dans la base de donnée, sera crypté. L'utilisation d'une Session (Stockage coté serveur) permettra à l'utilisateur de naviguer sur le site en étant connecté, de même pour le Back Office. L'application devra traiter les injections ainsi, chaque chaîne de caractère devra être protégée.

Autres fonctionnalités :

Un utilisateur aura accès au sitemap afin de trouver plus facilement les pages de l'application. Un sitemap sera aussi effectué afin de faciliter l'indexation des listes publiques et les utilisateurs notamment pour le moteur de recherche Google.

L'aide sera divisée en deux parties : La première sera représentée par une FAQ, elle concernera une aide générale concernant l'application. La deuxième sera spécialisée en fonction de la page ou de l'action. Un mail expliquant les possibilités de l'application sera envoyé lors de l'inscription. Enfin l'utilisateur aura accès à un module d'aide qui sera représenté par une liste, lors de la première connexion.

Chaque page devra être imprimable. Le design d'impression devra donc être épuré et simplifié afin d'imprimer que le contenu essentiel de la page : la liste, les informations d'un utilisateur, une semaine d'un agenda, etc.

Une notification correspond à un mail et à un message. Le message informe l'utilisateur qu'un autre utilisateur a eu une interaction avec lui. Ce message devra apparaître pendant quelques secondes. Il pourra aussi accéder à l'historique de ses notifications qui seront dé-

jà lues. Les mails auront les mêmes contenus que les messages. Le système enverra automatiquement les messages.

_ Mise en place d'un calendrier avec des types de calendriers existants

Le calendrier aura divers listes d'événements pré-existantes comme les jours fériés en France, ou les saints.

Mise en place d'un Client Lourd :

Le client lourd sera associé à un Webservice. Ce Webservice permettra à l'application de communiquer avec la base de donnée afin d'importer ou d'exporter des données (informations utilisateurs, informations listes). Il devra gérer de la même manière que le site internet, les listes principales (celle des lots obligatoires sauf les lots 4). Il pourra rechercher des listes, ou des utilisateurs via un moteur de recherches. Il aura accès aux notifications. Cependant, l'aspect communautaire sera assez réduit, il pourra seulement voir les utilisateurs. Il ne pourra pas inviter d'autres personnes à venir sur le site internet. L'application devra être autonome : si l'utilisateur perd sa connexion internet, l'application devra savoir gérer ses listes, ses utilisateurs, etc. Il pourra ajouter, modifier et supprimer des listes. Ainsi le stockage des données se feront via des fichiers xml stockés sur le poste de travail. L'application avec l'apparition de la connexion internet, devra synchroniser ses données avec la base de données via Webservice toutes les minutes. Il faut savoir que l'application Web prime sur le client lourd.

Lots Obligatoires

Lot 1 :

_ Liste d'envies / idées / résolutions

L'idée générale de ce lot est d'avoir d'une liste simple d'item. Ces items seront répertoriés en trois catégories : Envie, Idée et Résolution. Ils posséderont un titre, une description, une priorisation (Vert, Orange, Rouge). Cette liste sera de type "Sortable" avec un enregistrement des positions dans la base de données. L'utilisateur pourra trier la liste en fonction du titre ASC/DESC, de la position (Trie Par défaut) et de la priorité. Un effet toggle lors du click permettra à l'utilisateur de voir la description de l'item.

_ Liste de courses

L'idée de ce lot est de gérer une liste de course. Ce lot sera composé de deux parties différentes.

L'outil (Site internet) permettra à l'utilisateur d'ajouter des produits dans la liste de course et de le lister à la manière d'une liste " Sortable " en enregistrant les données dans la base de données.

L'outil (Client Lourd) permettra de récupérer cette même liste de course et de rayer ou supprimer les différents items de la liste. L'utilisateur, depuis son client lourd, pourra créer des listes de courses et la mettre à jour. Enfin, il devra synchroniser les données avec la base de donnée pour ne perdre aucune information.

_ Liste de tâches simples

Dans un premier temps, la liste de tâches simples ressemble à une «TODO». Cette liste pourra se synchroniser avec le calendrier. L'application demandera à l'utilisateur s'il doit le mettre dans le calendrier. Si oui, il demandera aussi s'il doit y avoir un rappel. Enfin l'utilisateur pourra mettre une priorité pour chaque tâche. Ces priorités seront identifiables par des couleurs : rouge, jaune, vert. La liste sera de type sortable, chaque item pourra être classé par Drag N Drop. Comme tout type d'item, il pourra donner un accès spécifique à sa liste, et il pourra donc partager ces tâches via le calendrier avec ses amis.

_ Répertoire téléphonique

L'idée générale de ce lot est de gérer une liste d'information de contacts. Il existe deux sortes d'ajout dans le répertoire :

L'ajout automatique lors de l'acceptation d'une demande d'ami par un autre utilisateur et qui a autorisé l'accès à ses informations, ou l'ajout manuel des informations d'un nouveau contact par l'utilisateur (Nom, Prénom, Adresse, email, date de naissance (Liaison avec le calendrier (Confirmation)). Il pourra alors ajouter un nombre illimité de numéros de téléphone en fonction de son type (Domicile, travail, Fax, Portable, Fixe, etc). Il pourra aussi modifier et supprimer les informations rentrées par un contact. De plus, il pourra pour les numéros de téléphone de type Portable envoyer des SMS. Enfin, L'utilisateur pourra exporter ses données (en choisissant les colonnes) via un format csv, xml, vCard et importer des contacts via csv et vCard.

Lot 2 :

_ Liste de document de travail avec gestion de droit

Un utilisateur pourra uploader des fichiers au format pdf, doc, txt, html. Comme toutes listes, ces documents auront un statut qui pourra être privée, publique ou protégé. Pour chaque document, il pourra donner des accès en lecture et écriture pour chaque personne ayant accès à celui-ci. Par défaut, un utilisateur peut télécharger le document. Seul un utilisateur qui peut écrire dans le document pourra uploader une nouvelle version de ce document. Les différents utilisateurs auront donc plusieurs versions d'un même document : Il verra donc une liste d'un même document mais de versions différentes.

_ Liste de lecture en ligne avec lecteur embarqué sur le site

Un utilisateur aura accès à des items de type musique. Ces items devront être lu dans un lecteur embarqué fait en Javascript, ou Flash. Lors de l'ajout d'un item de ce type, l'utilisateur devra associer un flux via un lien, ou un fichier qui sera uploader sur le serveur. Les formats seront déterminés via la configuration de l'application du BO par l'administrateur. Le module devra permettre de lire, mettre pause, ou arrêter la lecture. Il devra aussi naviguer entre les différents items de type musique de la liste grâce au menu suivant et précédent.

Lot 3 :

_ Liste d'anniversaire ou événement avec un système de rappel et lié au calendrier

L'idée générale de ce lot est de gérer des événements sur le calendrier existant. En effet, sur l'outil TOOTLIST, un calendrier est prévu de base.

L'ajout d'un événement ou tâche dans le calendrier se fait par le remplissage d'un formulaire afin de donner plus amples informations (titre, date de début, date de fin, lieu, description). Pour chaque tâche, on peut choisir un nombre illimité de rappel qui prendra en compte la façon et le temps pour l'émission du rappel : Popup, Email, SMS, 10 minutes avant, 1 heure avant, etc. Chaque tâche pourra être partager en reprenant le même principe que les listes (Privé, Protégé et Public). De plus, un calendrier d'un utilisateur aura plusieurs agendas (Personnel, Professionnel, ...), il pourra donc associer une tâche à l'un de ces calendriers. L'utilisateur qui a créé la tâche pourra la supprimer. L'ajout d'anniversaire peut être considéré comme une tâche que l'utilisateur avec la différence qu'il peut envoyer un message personnalisé sous forme d'Email ou de SMS. Le rappel par mail sera automatique pour que les utilisateurs n'oublient plus les anniversaires de leurs contacts.

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Dans ce mail, nous aurons accès à un envoi de SMS, d'Email afin de souhaiter un joyeux anniversaire au contact concerné. Lors de l'ajout d'un contact qui sera validé, le nouveau contact peut donner l'accès à ses informations afin que le calendrier rajoute de lui-même un évènement de type Anniversaire qui correspondra à son anniversaire.

_ Album Photo avec Zoom, légendes et animations

L'album photo permettra de voir des items de type image. On pourra accéder à une diaporama afin de voir toutes les images. Lors d'un clic sur la photo, si on est sur le diaporama, permettra de voir la photo via un full-screen. Lors d'un clic sur la photo, sur la page de la photo, on pourra voir un zoom de celle-ci en fonction de la position du curseur de la souris. L'utilisateur, pour ajouter une image, devra donner un lien ou une fichier à uploader sur le serveur.

_ Liste d'ami avec messagerie interne (mini-chat & envoi de message)

La liste d'ami sera proposé de base par l'outil "TOOTLIST". Cet lot obligatoire permettra de renforcer la communication entre les utilisateurs qui sont amis. Cette communication sera découpée en deux parties :

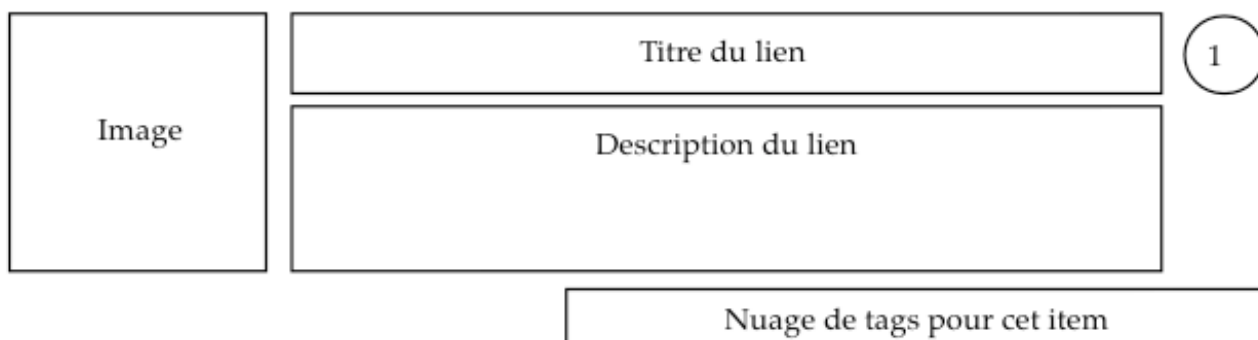
Messagerie interne : un utilisateur X peut envoyer un message à un utilisateur Y ou /et à N utilisateurs. Cela sera représenté par une liste de messages qui seront compris dans une discussion. Un message est composé d'une description (contenu du message). Une discussion possèdera un titre, un destinataire ou des destinataires. La création d'un discussion entrainera la création d'un message. Lorsqu'un utilisateur accèdera à sa messagerie interne, il pourra aussi accéder à ses messages envoyés, à ses messages reçus lus, à ses messages reçus non lus et à ses brouillons (Messages qui n'est pas encore envoyés).

Utilisation d'un mini-chat : Un utilisateur verra à n'importe quel moment si ses amis sont eux-aussi sur l'outil. Si c'est le cas, il pourra alors commencer un chat avec eux. Le chat proposera dans un premier temps une discussion entre deux personnes. Un utilisateur pourra à tout moment se rendre invisible auprès de ces amis. Lors d'un chat entre deux personnes, si l'une des deux personnes perd la connexion avec l'outil, celui-ci transférera automatique la discussion "chat" en une discussion "messagerie interne", afin de ne perdre aucun flux de données.

_ Liste de Bookmarks avec Aperçu

Un Bookmarks est un item qui correspondra à une page internet quelconque. L'idée géné-

rale est donc de lister les différentes pages internet que l'utilisateur voudra à la manière des marques pages du navigateur Firefox. Ainsi, il pourra ajouter, modifier ou supprimer un item. Cette liste sera aussi "sortable" et sera sauvegardée dans la base de données. Lors de l'ajout d'un item, on pourra donner un lien (http://), un titre, une liste de tags, une description et une image. La création de la liste possédera un nom et son design pourra être changé.



Représentation d'un item dans la liste Bookmarks avec Aperçu

Au survol de la souris dans la box "Image", nous pourrions avoir un aperçu du site dans une fenêtre intégrée dans l'application. Lors de la création d'un item, si l'utilisateur ne met pas d'image, l'application mettra à la place une vue en temps réel de l'url.

(1) : La box (1) existe si l'utilisateur n'a pas encore créé de liste de Bookmarks, et que c'est la page par défaut. Cette box correspond au nombre de personne qui ont mis ce lien dans leurs listes. Cela sera le trie par défaut (ASC)

Lot 4 :

_ Liste de villes et vue de la météo de ceux-ci

L'idée générale de ce lot est de lister un certain nombre de villes afin de voir la météo. Cette liste sera administrable par le propriétaire de la liste : il pourra ajouter, modifier et supprimer une ville. Pour chaque ville, il aura accès à la météo pour un certain nombre de jour qu'il pourra configurer. Nous utiliserons le service de "The Weather Channel" (TWC) qui est fiable, détaillé et à l'avantage d'avoir une couverture mondiale. Lors de la création de la liste, l'utilisateur aura 1 item qui est sa ville où il se trouve. De plus, au fur et à mesure de la création d'item, il pourra gérer la position de ses items par "liste sortable" (Drag & Drop des items dans une liste) et cette configuration de position sera enregistrée dans la

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

base de données. L'administrateur pourra lui aussi configurer l'application en donnant le nombre maximum de ville qu'on peut mettre dans cette liste.

_ Liste de Pokémons avec un élevage de ceux-ci

L'idée générale de ce lot est de faire une liste mais d'une façon originale. On pourra ainsi jouer au célèbre jeu Pokémon avec une implémentation dans une liste. Ce jeu proposera à l'utilisateur d'élever un pokémon ou plusieurs pokémons. Ainsi l'utilisateur aura accès à deux listes : sa liste principale, et sa liste de stockage. Cette liste principale sera composée de N pokémons, dont N sera configurable depuis le BO par l'administrateur. Ces pokémons sont des petits animaux qui possèdent une caractéristique spécifique (Normal, Feu, Eau, Plante, Électrique, Glace, Combat, Poison, Sol, Vol, Psychique, Insecte, Roche, Spectre, Dragon, Ténèbres, Acier) et des caractéristiques globales comme des points de vie, une attaque, un niveau, une défense. Il y a 151 pokémons en tout. L'utilisateur pourra acheter des produits pour ses pokémons afin de le soigner. Il aura donc de l'argent. Chaque produit sera alors ajouté par l'administrateur (en donnant son nom, ses effets et son prix) depuis le BO. Un pokémon peut évoluer en gagnant des niveaux en fonction de ses points d'expérience. Ces niveaux sont gagnés en remportant des points d'expérience lors des défis. Il peut aussi évoluer lors d'échanges entre amis. Lorsqu'un pokémon monte de niveau il est possible que le pokémon apprenne de nouvelles techniques. S'il a déjà 4 techniques l'utilisateur peut faire oublier au pokémon une technique.

Un défi est composé de deux choses :

_ Un combat contre un ami : Un ami décide de défier vos pokémons. Ainsi il y a un combat entre tous les pokémons et la personne qui possède des pokémons qui peuvent encore combattre gagne. Ainsi chaque pokémon gagnant gagne des points de vie selon cette fonction mathématique. Le combat entre deux pokémons adoptés est en fonction de sa caractéristique spécifique, de son niveau et de l'aléatoire. A la fin du combat, la moitié de l'argent que possède le perdant est donné au gagnant.

_ Un combat avec un pokémon sauvage : Lorsqu'une personne n'a pas de défi, cela peut arriver qu'il croise le chemin d'un pokémon sauvage. L'utilisateur possède deux choix : soit il le capture et le met dans la liste principale s'il peut, sinon sur sa liste de stockage, soit il l'attaque tant qu'il peut combattre, soit il s'enfuit. Si l'utilisateur possède déjà ce pokémon, un indicateur sous forme de pokéball sera affiché. Le combat entre deux poké-

mons est en fonction de sa caractéristique spécifique, de son niveau et de l'aléatoire. Il pourra aussi la capturer pendant le combat. Cette adoption est en fonction de la prise en compte du niveau de vie actuelle du pokémon sauvage, de son moral et de l'aléatoire. L'aléatoire sera aussi modifié en fonction de type de pokéball utilisé. L'utilisateur pourra à tout moment modifier sa liste principale avec sa liste de stockage.

Tâches Bonus

_ Accès à Google Map

Pour chaque adresse présentée à l'utilisateur (ami, événement, album photo, répertoire téléphonique, ville (météo)), en cliquant sur celle-ci, une fenêtre modale s'affichera avec dedans une Google Map donnant l'adresse. On utilisera l'API Google Map parce qu'elle est simple d'utilisation et surtout elle est fiable.

On pourra voir l'itinéraire entre l'adresse de l'utilisateur et l'adresse cliqué pour les cas suivant : adresse d'un évènement, adresse d'un ami, adresse d'un répertoire téléphonique. On donnera toutefois le choix à l'utilisateur avant l'ouverture de la modale.

_ Internalisation

Un menu déroulant (toggle) sera disponible à l'utilisateur pour choisir sa langue. Par défaut, la langue sera la langue de son navigateur, ou de son système d'exploitation. L'internalisation sera composée de deux langues : anglais et français. L'internalisation sera géré par le Framework PHP Zend pour le site internet, et l'utilisation de tableaux et d'une fonction helpers sur le client lourd.

_ Export CSV, XML des données

L'idée générale de ce lot est d'exporter toutes listes au format CSV et XML. L'utilisateur pourra choisir les colonnes à exporter. L'importation implique une connaissance des colonnes et sera fonctionnelle pour certaines types de listes notamment le répertoire téléphonique.

_ Sécurisation

Un utilisateur devra rentrer un code pour pouvoir s'enregistrer (CAPTCHA). Ensuite, un email lui sera envoyé avec un lien qui permettra de confirmer son inscription. Ce processus sera obligatoire pour qu'il puisse se connecter à l'application (Client Lourd et Site in-

ternet).

Lorsqu'un utilisateur aura oublié son mot de passe, il devra rentrer son mot de passe. Un email sera alors envoyé afin de régénérer son mot de passe depuis l'application.

L'utilisateur sera identifié par des variables de types SESSION sur le site internet qui sera codé en MD5 pour le FO et le BO, contrairement au client lourd qui récupérera des informations afin dans des fichiers ayant des permissions spécifiques.

Lors des requêtes SQL, l'application devra faire face aux injections SQL ou à tout type de script qui altérerait l'application (comme des scripts Javascript).

L'application internet, basé sur le framework PHP Zend, utilise l'Access Control List (ACL), qui signifie liste de contrôle d'accès. Elle désigne un système permettant de faire une gestion plus fine des droits d'accès aux fichiers que ne le permet les méthodes employées par les systèmes classiques. Sous Zend, ACL est représenté par un module. Ce module a été conçu pour ne pas nécessiter de technologie spécifique comme une base de données ou un serveur de cache pour conserver les données ACL. Le module d'authentification de Zend est décidé par Zend_ACL.

_ Gestion des encarts de publicité depuis le Back-Office

Chaque page disposera d'un ou plusieurs encarts pour la publicité. La publicité ne devra pas empêcher le site de fonctionner, ni même d'entraver la navigation d'un utilisateur.

L'administrateur pourra ajouter depuis le BO des publicité, en donnant une image ou un flash, une url, une date de début, une date de fin et un statut (Activé, Désactivé) . Il associera la publicité à des lots qui regroupent plusieurs encarts sur différentes pages. La mise en place des lots se fera en même temps que les pages seront créées.

Exemple de lots :

- _ Site internet : Home + page de niveau 1
- _ Site internet : page de niveau 2
- _ Site internet : Page de configuration
- _ Client Lourd : Home + page de niveau 1
- _ Client Lourd : page de niveau 2
- _ Client Lourd : Page de configuration

Cela impliquera que l'administrateur pourra recevoir par mail des demandes pour mettre des publicités sur le site internet. tootlist@gmail.com // ing3sigl. L'administrateur aura,

dans la BO, la possibilité de voir ces emails du compte tootlist@gmail.com afin de vérifier les nouvelles demandes de publicités.

_ Priorité des tâches

Chaque item d'une liste pourra avoir une propriété concernant la priorité. Cette priorité sera représentée par une couleur : Rouge pour une priorité haute, Orange pour une priorité moyenne et Vert pour une priorité basse. Chaque liste présentant une notion de priorité pourra être triée en fonction de celle-ci.

_ Communication :

_ SMS

Un SMS sera envoyé gratuitement par l'application afin de souhaiter un joyeux anniversaire à chaque membre.

_ Mail

L'application devra envoyer des mails lors des actions des utilisateurs. Un email de contact sera disponible afin de communiquer avec l'administrateur via le BO tootlist@gmail.com

_ Lecteur vidéo

L'application possédera un module de lecteur vidéo. Il devra lire les formats .flv. Ce module devra posséder les fonctions suivantes : Lire, Pause, Arrêter, Suivant, Précédent.

_ Sauvegarde

La sauvegarde concerne la base de donnée et les fichiers uploadés par tous les utilisateurs. Un script .sh permettra de récupérer la base de donnée via un dump. Ajouté à ce fichier, il ajoutera les différents fichiers du site internet avec les fichiers utilisateurs. Il compressera ce dossier fraîchement créé et lui donnera un nom défini comme celui-ci : sauvegarde_jj_mm_aaaa.zip. Il sera exécuté tous les jours via une tâche cron.

_ Liste avec le statut protégé

Cette fonctionnalité se base sur le statut protégé des listes. Le statut "protégé" de base permet à tous les amis de l'utilisateur de voir les listes. La fonctionnalité permet donc à l'utilisateur de choisir les amis qui verront sa liste. Ainsi seuls les amis autorisés à voir la liste verront cette liste.

_Statistique

Voir la description du BO

_ Notion de parrainage

Chaque utilisateur pourra envoyer des mails afin d'inciter des amis à s'inscrire à l'application. Il pourra donc regarder dans les listes de ses contacts Yahoo, Gmail et Hotmail. Pour les utilisateurs utilisant l'application, il pourra les demander en amis. S'il n'existe pas de compte associé au mail, alors l'utilisateur peut envoyer un mail de parrainage afin d'agrandir son cercle d'amis et de faire connaître l'application. Le parrainage permettra à l'utilisateur de gagner du quota pour la création de liste.

Spécifications des tâches du client lourd

Le client lourd sera associé à un Webservice. Ce Webservice permettra à l'application de communiquer avec la base de donnée afin d'importer ou d'exporter des données (informations utilisateurs, informations listes). Il devra gérer de la même manière que le site internet, les listes principales (celle des lots obligatoires sauf les lots 4). Il pourra rechercher dans ces listes grâce au moteur de recherche. Il aura accès aux notifications. Cependant, l'aspect communautaire sera assez réduit, il pourra seulement voir les utilisateurs. Il ne pourra pas inviter d'autres personnes à venir sur le site internet. L'application devra être autonome : si l'utilisateur perd sa connexion internet, l'application devra savoir gérer ses listes, ses utilisateurs, etc. Il pourra ajouter, modifier et supprimer des listes. Ainsi le stockage des données se fera via des fichiers xml stockés sur le poste de travail. L'application avec l'apparition de la connexion internet, devra synchroniser ses données avec la base de données via Webservice toutes les minutes. Il faut savoir que l'application Web prime sur le client lourd.

Le client lourd utilisera le framework Hibernate pour accéder aux données.

Documentation

Développement

Le développement se fera grâce à des outils particuliers. En effet, notre groupe utilisera les outils de google code (SVN). Le SVN est un outil de gestion de version. En effet, chaque évolution de l'application est stockée. De plus, l'essentiel des développeurs du groupe est sous Mac OS X, ainsi notre groupe utilisera Coda comme IDE qui intègre le support de SVN.

Syntaxe PHP :

- _ Utiliser `<?php $toto = 'toto'; ?>` au lieu de `<? $toto = 'toto'; ?>`
- _ Utilisation de la camelCase pour les méthodes
- _ Utilisation des méthodes magiques pour gérer les relations entre les tables de la bases de données (Méthodes Zend)
- _ Un commentaire sera fait pour chaque fonction écrite.

Syntaxe HTML :

- _ Externationalisation des styles, des scripts javascripts

Modèle MVC :

- _ Limiter le PHP au boucle et au condition dans les templates (View)

Helpers :

- _ Un helper permet d'avoir des fonctions pouvant être appelées dans toutes les vues sans être relié à un model. Les helpers sont généralement utilisés pour les tâches récurrentes (Formatage des téléphones).

Modèle :

- _ Un modèle est une représentation d'une table (Design pattern : Active Record)
- Dans ce model, nous aurons les propriétés (Attributs de la table) et les méthodes (assesseurs, fonctions relatives au modèle (filtres)) liés à ce modèle.

Internationalisation :

_ Deux fichiers (/ applications / languages) permettront de faire la traduction. Dans ces fichiers, nous aurons un tableau avec les différentes traductions. Nous récupérerons la traduction grâce à `$this->translate($key);`

Fichier de configuration

La configuration de l'application Web se fera par le biais de la configuration des application sous le Framework Zend.

Scripts à exécuter

Documentation BO

Documentation FO

Installation

How To

Test

Scénario 1

Scénario 2

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

• Projet TOOTLIST • ING3 SIGL • INSIA •

Planning des tâches

Spécification

Durée : 2 mois

Analyse : 10/09 => 01/11

Cahier des charges de la partie fonctionnelle I (Tâches Obligatoire) 01/11 => 10/11

Spécification théorique des Lots et des Bonus 01/11 => 15/11

Cahier des charges de la partie fonctionnelle II (Partie I + Lots + Bonus) 15/11 => 01/12

Modélisation de la base 01/12 => 31/12

Architecture

Durée : 15 jours

Mise en place du serveur et du matériel requis 15/12 => 31/12

Développement

Durée : 4 mois et demi

Gestion des Utilisateurs

Gestion des Listes

Gestion de la Personnification

Gestion du Back-Office

Création des Vues

Gestion de la Communication

Création du Webservice

Création du Client Lourd

Internalisation

Création des Lots

Création des Lots Bonus

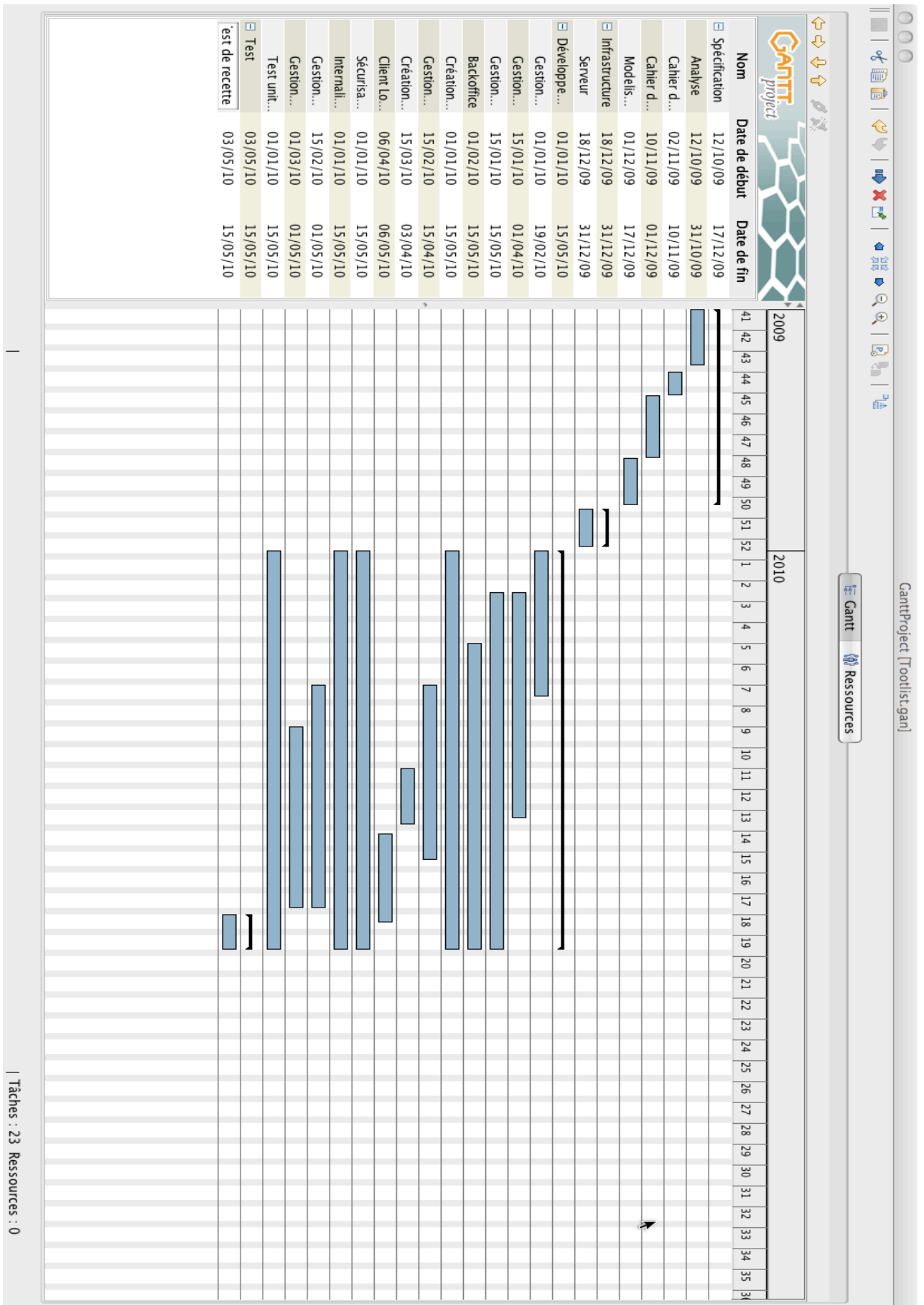
Tests

Durée : 15 jours

Test d'intégration

Test de Scénario

SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas



SANTANGELI Yohann, PAJOT Christophe, ROUCHES Nicolas, ROGER Thomas

Coût

Relation Humaine

Equipe de 4 personnes

Chef de projet : 650 Euros / j

Utilisation à 10% : 10 mois

Développeur : 500 Euros / j

Utilisation à 100% : 4 mois

Intégrateur : 400 Euros / j

Utilisation à 100% : 10 jours

Hardware / Software

Serveur (Base de données et Hébergement)

Système d'exploitation : Unix (Gratuit)

Serveur Web : Apache (Gratuit)

Système de gestion de base de données : MySQL (Gratuit)

Site Internet

Programmation : PHP (Gratuit), XHTML (Gratuit), CSS (Gratuit)

Utilisation d'un Framework PHP : Zend (Gratuit)

Framework Javascript : Mootools

Client Lourd

Programmation : J2SE (Gratuit)

Autre prestation

Envoi de SMS via un prestataire externe qui sera rentabilisé par la publicité.

Analyse des risques

Technique

Néant

Persomnel

Les risques concernant ce projet sont :

- _ Le manque d'organisation
- _ Le manque de temps

Juridique

Le projet aura comme propriétaire l'INSIA dans le cadre d'un projet de fin d'année.

Suite à la loi du 6 janvier 1976, les utilisateurs disposeront du droit de modification, de suppression de leurs informations les concernant.

Un retard dans la livraison aura pour conséquence une diminution de la note concernant le côté technique de l'outil.