



c e r o  

---

**repository**

***Rapport de Projet***

***Projet n°25***

***Tuteur de projet : Peter SANDER***  
***Année universitaire 2006/2007***

***Michael LAGUERRE – Camille ROUX – Matthieu SEGRET – Mathieu SIVADE***

## SOMMAIRE

Introduction .....	4
Problématique .....	5
Organisation de Cero Core avec les plugins.....	5
Nombre de plugins .....	5
Solutions possibles.....	5
Cahier des charges .....	7
Interface simple .....	7
Installation facile.....	7
Durée de développement courte .....	7
Gestion des paquets .....	7
Différents niveaux d'utilisation.....	7
Modération.....	7
Gestion des relations entre paquets .....	8
Mise à jour .....	8
Open source.....	8
Les dépôts existants .....	8
Qu'est-ce qu'un dépôt ? .....	8
Qu'est-ce qu'un paquet ? .....	8
Caractéristiques .....	8
Solutions implémentées.....	9
Implementation du Site Web.....	9
Structure MVC .....	9
Modèles.....	9
Versionning.....	10
Authentification et droits .....	10
Relations inter-branches .....	11
Organisation des pages .....	11
Implémentation du client java.....	11
Structure du client.....	11
Utilisation de l' XML .....	12
Utilisation de JXTable .....	12
Installation d'une release .....	13
Processus d'update .....	13
Points à rajouter.....	14
La modération.....	14

Les flux RSS .....	14
La partie visiteur .....	14
Amélioration générale du design et de l'ergonomie .....	14
Système de commentaires et de notes .....	15
Localisation .....	15
Système de gestion de tags .....	15
Statistiques .....	15
Journalisation des actions.....	15
Personnalisation des dépôts.....	15
Conclusion .....	16
Installation du depot.....	18
Les fonctionnalités offertes à l'utilisateur.....	19
Site Web.....	19
Client Java .....	22
Scénarios d'utilisation .....	23
Application Web .....	23
Navigation en tant que Visiteur.....	23
Navigation en tant qu'utilisateur.....	24
Navigation en tant que modérateur.....	27
Navigation en tant que super-modérateur/administrateur.....	27
Client JAVA.....	28
Mise à jour de la liste des releases disponibles.....	28
Rechercher et filtrer les releases .....	28
Affichage de description.....	29
Installation d'une release .....	29
Glossaire.....	30
Annexes.....	32
Structure de la base de données .....	32
Exemple de XML transmis lors de l'update du client Java .....	33
Comparatif des frameworks Ajax .....	34
Mootools .....	34
Prototype.....	34
Scriptaculous (+ Prototype).....	35
Dojo .....	35
Répartition temporelle des tâches .....	36

## 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'arrivée 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 :



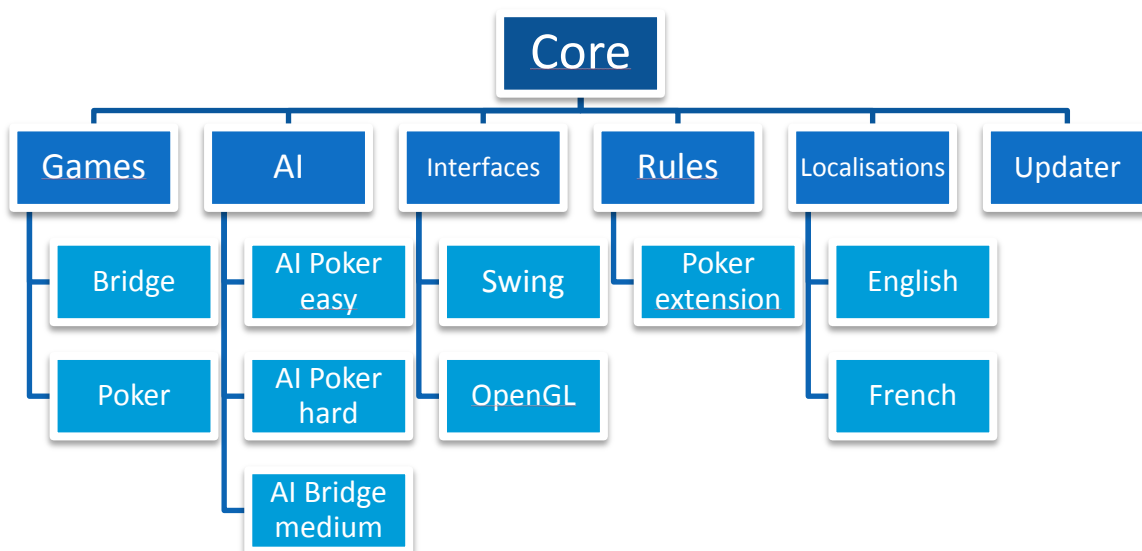
Nous ne pouvons rester de simples spectateurs de cette révolution du cybermonde.

## PROBLEMATIQUE

Durant le premier semestre de 2006, nous avons développé un manager de jeu de cartes open source: [Cero Project](#). Il s'agit d'un programme Java proposant différentes interfaces pour faciliter le développement de divers plugins : jeux, intelligences artificielles, interfaces (graphiques, ligne de commande, IRC, ...), règles du jeu, traductions, client pour la mise à jour, ...



## ORGANISATION DE CERO CORE AVEC LES PLUGINS



## NOMBRE DE PLUGINS

Le noyau Cero Core est très modulaire. De plus, une communauté de développeurs, de designers, de rédacteurs, de traducteurs, ... risque de se créer autour de ce projet open source. Le noyau rend la création de plugins relativement simple, on peut donc s'attendre à l'apparition d'un grand nombre d'extensions. Un problème se pose alors : comment les mettre à disposition de manière simple et facile d'accès ?

## SOLUTIONS POSSIBLES

Les premières solutions auxquelles on a pensé étaient d'utiliser un serveur de téléchargements avec éventuellement une interface web pour le gérer ou un simple serveur FTP. Nous avons rapidement oublié cette première solution car il n'était pas simple de l'adapter à notre problème (pas de gestion de dépendances possible par exemple). Nous nous sommes alors penchés sur des solutions plus proches de ce qu'il nous faudrait, les systèmes de dépôt linux.

# CERO REPOSITORY

## CAHIER DES CHARGES

### INTERFACE SIMPLE

Notre gestionnaire de paquets n'est pas réservé à des informaticiens. Effectivement, des designers ou des traducteurs d'une application doivent pouvoir ajouter, eux aussi, facilement des paquets. On souhaite que l'ajout puisse être fait de manière graphique. Pour ce faire, nous avons choisis de développer un site web, ce qui permet de rendre la gestion des paquets facile et convient parfaitement pour un partage simple de données.

### INSTALLATION FACILE

Cero Project est un projet open source et nous espérons avoir une importante communauté qui gravite autour assez rapidement. Une fois le projet terminé, nous souhaitons que Cero Repository puisse être très facilement installé. Nous voulons que quelqu'un qui a quelques connaissances niveau web, puisse installer notre application sur un serveur sans aucun problème.

### DUREE DE DEVELOPPEMENT COURTE

Pour ce projet, seulement trois semaines nous étaient réservées. Beaucoup de choix devaient être pris avant de commencer l'implémentation : choix du langage (PHP, Ruby, Python ou JSP?), choix de l'utilisation de l'AJAX, choix de l'utilisation de Framework,...

Pour respecter les temps prévus, nous avons effectués toutes les recherches et les choix nécessaires durant les deux mois précédents les 3 semaines de projets.

Nous avons donc choisis d'utiliser le PHP car nous voulions que Cero Repository puisse être installé chez la plupart des hébergeurs. Nous avons aussi fait le choix d'utiliser des Frameworks PHP (Code Igniter) et AJAX (Prototype & Scriptaculous) afin de nous fournir une bonne structure pour le développement et de ne pas avoir à faire de réécriture.

### GESTION DES PAQUETS

Une autre spécification du projet est de pouvoir facilement gérer les paquets. On doit pouvoir ajouter, supprimer ou encore éditer les paquets de manière simple et conviviale.

### DIFFERENTS NIVEAUX D'UTILISATION

Notre application doit être capable de protéger le contenu de certaines pages (ex : édition de profil,...). Pour proposer cette fonctionnalité, Cero Repository doit permettre de définir le rôle des utilisateurs : simple utilisateur, modérateur ou administrateur par exemple.

### MODERATION

Comme tout site qui permet aux utilisateurs d'ajouter ou modifier du contenu, on se devait de mettre en place un système de modération. Tout paquet ajouté doit ainsi pouvoir être vérifié par un modérateur dans un premier temps avant sa mise en ligne effective.

On souhaite aussi pouvoir faire de la modération sur les utilisateurs. Par exemple, il doit être possible de bannir un bot, de rappeler certaines règles à un utilisateur ou encore modifier ses droits.

## GESTION DES RELATIONS ENTRE PAQUETS

Nous voulons aussi que notre gestionnaire de paquets ne se résume pas à un simple serveur de téléchargement. Nous souhaitons de plus être à même de gérer les relations entre deux paquets. En effet, un paquet doit pouvoir préciser qu'il a besoin d'un autre pour fonctionner par exemple, ou encore que deux paquets ne peuvent être installés en même temps sous peine de conflits.

## MISE A JOUR

Une autre caractéristique du projet doit être la possibilité de télécharger les informations contenues dans la base de données et ceci en minimisant l'utilisation de la bande passante. Pour ce faire, un fichier décrivant uniquement la mise à jour à effectuer sur la base de données locale (chez le client) doit être envoyé.

## OPEN SOURCE

Enfin, Cero Project étant libre (licence GPL), Cero Repository se doit de l'être également. Ceci permettra entre autres à d'autres personnes de participer au projet et permettra donc à ce dernier d'évoluer beaucoup plus vite.

## LES DEPOTS EXISTANTS

### QU'EST-CE QU'UN DEPOT ?

En informatique, un dépôt (de l'anglais « repository »), est un stockage centralisé et organisé de données. Ce peut être une ou plusieurs bases de données où les fichiers sont localisés en vue de leur distribution sur le réseau, ou bien un endroit directement accessible aux utilisateurs.

La plupart des distributions Linux utilisent des dépôts accessibles sur Internet, officiels et non-officiels, permettant aux utilisateurs de télécharger et de mettre à jour des logiciels compatibles. Ces dépôts distribuent les logiciels sous forme de paquets

### QU'EST-CE QU'UN PAQUET ?

Un paquet est un fichier informatique encapsulant le ou les fichiers nécessaires au fonctionnement d'un logiciel. Ils incluent aussi généralement des informations de gestion, comme le nom complet, la version, le nom de la personne qui a créé le paquetage, la somme de contrôle, et la liste d'autres paquetages (appelés dépendances) qui sont nécessaires à ce logiciel pour fonctionner correctement.

## CARACTERISTIQUES

Quelque soit le système de dépôt utilisé, il est nécessaire de créer un (des) fichier(s) de description, ainsi que de regrouper les fichiers utiles au programme. Ceci est une opération relativement longue et complexe, et on

ne peut donc attendre qu'elle soit réalisée par des designers ou des traducteurs par exemple. De plus, les systèmes de dépôt actuels ne proposent pas de mise en œuvre d'un système de modération, il faut donc l'implémenter soi-même en fonction des besoins.

Pour ce faire, Debian doit mettre en place de très gros moyens pour gérer les dépôts de base. Ils disposent de plusieurs serveurs FTP (pour pouvoir envoyer des paquets et les mettre à disposition), serveurs mail (pour envoyer des informations aux utilisateurs : ajout de paquet, bugs, ...), serveurs web, ...

Bien évidemment, nous ne pouvons pas espérer avoir de tels moyens et donc cette solution ne peut nous satisfaire. Nous avons donc du trouver un autre moyen de résoudre ce problème.



## SOLUTIONS IMPLÉMENTÉES

### IMPLEMENTATION DU SITE WEB

#### STRUCTURE MVC

Nous nous sommes reposés sur l'architecture MVC (Modèle-Vue-Contrôleur) proposée par Code Igniter pour développer l'interface web. Nous vous recommandons fortement de lire la documentation en ligne de Code Igniter ([http://www.codeigniter.com/user\\_guide/](http://www.codeigniter.com/user_guide/)) pour appréhender ce fonctionnement, auquel nous nous sommes conformés autant que possible.

#### MODELES

Chaque modèle sert donc d'interface avec la table correspondante dans la base de données. Pour la plupart de ces modèles, nous avons utilisé premièrement le système « Active Record » de Code Igniter, qui fournit des abstractions pour l'accès à la base de données quel que soit le SGBD (MySQL, PostgreSQL, etc). Nous avons de plus factorisé les fonctionnalités de base de nos modèles dans une classe mère (BaseModel) grâce au mécanisme d'héritage de PHP. Celle-ci exploite la similarité entre nos tables (une seule clé primaire, des données à protéger et d'autres non, le versionning des données) pour fournir des fonctions génériques :

- **Constructeur** : Prend en paramètre le nom de la table liée au modèle, le nom de champ de la clé primaire, un tableau contenant les noms des champs pouvant être modifiés par l'utilisateur (nom, prénom, e-mail par exemple), et un paramètre optionnel indiquant si les données doivent être versionnées (voir plus bas). Par défaut, les champs qui ne sont pas modifiables par l'utilisateur ne seront pas pris en compte lors de l'enregistrement des données dans la table.
- **Load** : charge les données de la table indiquée dans le constructeur, et en stocke la première ligne dans le tableau data. En utilisant la fonction 'where' de l'active record avant d'appeler 'load', les classes filles peuvent filtrer les données obtenues, par exemple pour obtenir l'enregistrement d'un utilisateur précis, ou encore faire une jointure avec la fonction 'join'.

- `New_data` : elle prend en premier paramètre les données non sécurisées, et en second paramètre les données déjà vérifiées par la classe fille. Après avoir fait vérifier les données par la classe fille, elle les insère dans la table.
- `Save_data` : Fonctionne similairement à `new_data`, mais prend en paramètre supplémentaire la valeur de l'id du champ à modifier (pour filtrer automatiquement).
- `Fields_are_valid` : Cette fonction doit dans tout les cas être surchargée par la classe fille : elle prend en paramètre un tableau faisant correspondre, à un champ de la table, la valeur à lui attribuer. Elle renvoie vrai si les valeurs sont valides, faux sinon.
- `Exists` : Indique s'il existe des tuples correspondant aux contraintes passées en paramètre.
- `Get_data` : permet de récupérer un ensemble de données de la table. On peut en limiter la quantité, définir le sens de tri et des contraintes de ressemblances (LIKE de SQL).

Cependant, certains modèles spécifiques ne reposent pas sur cette architecture, comme par exemple le modèle de mise à jour. Pour plus d'informations sur ces modèles, vous devriez trouver les réponses à vos questions dans la documentation de chacun des fonctions située dans le source.

---

## VERSIONNING

Le versionning permet de savoir si une donnée de la base est antérieure à une autre. Il consiste à renseigner un champ « `db_version` » dans chaque table souhaitant être versionnée. Le numéro de version actuel de la base de données est stocké dans la table `cr_config`. Ce sont ces valeurs de version qui permettent d'envoyer le strict minimum lors de la fonctionnalité de mise à jour par XML.

---

## AUTHENTIFICATION ET DROITS

N'ayant trouvé de bibliothèques pour Code Igniter proposant les fonctionnalités d'authentification et une documentation suffisante, nous avons décidé d'implémenter celle-ci en utilisant la bibliothèque de session de Code Igniter. Nous identifions donc les utilisateurs en stockant dans leurs données de session leur numéro d'utilisateur, ce qui permet de les reconnaître de manière sécurisée.

Pour vérifier les droits d'un utilisateur d'accéder à une page ou d'effectuer une action, nous utilisons la fonction « `has_rights_to` » de la bibliothèque Auth, prenant en paramètre le type d'action (édition, création, ...), le type de cible (utilisateur, paquet, article, ...) et un id optionnel dans le cas d'opérations sur une cible unitaire. Celle-ci utilise une table de droits qui contient un nombre sur deux chiffres : le chiffre des dizaines indique quel degré d'utilisateur peut accéder directement à la page. Le chiffre des unités indique quel degré d'utilisateur peut accéder à la page si l'id optionnel correspond à l'utilisateur connecté (par exemple, s'il s'agit d'une action sur un package, le responsable du package dont l'id est donné doit être l'utilisateur connecté).

## RELATIONS INTER-BRANCHES

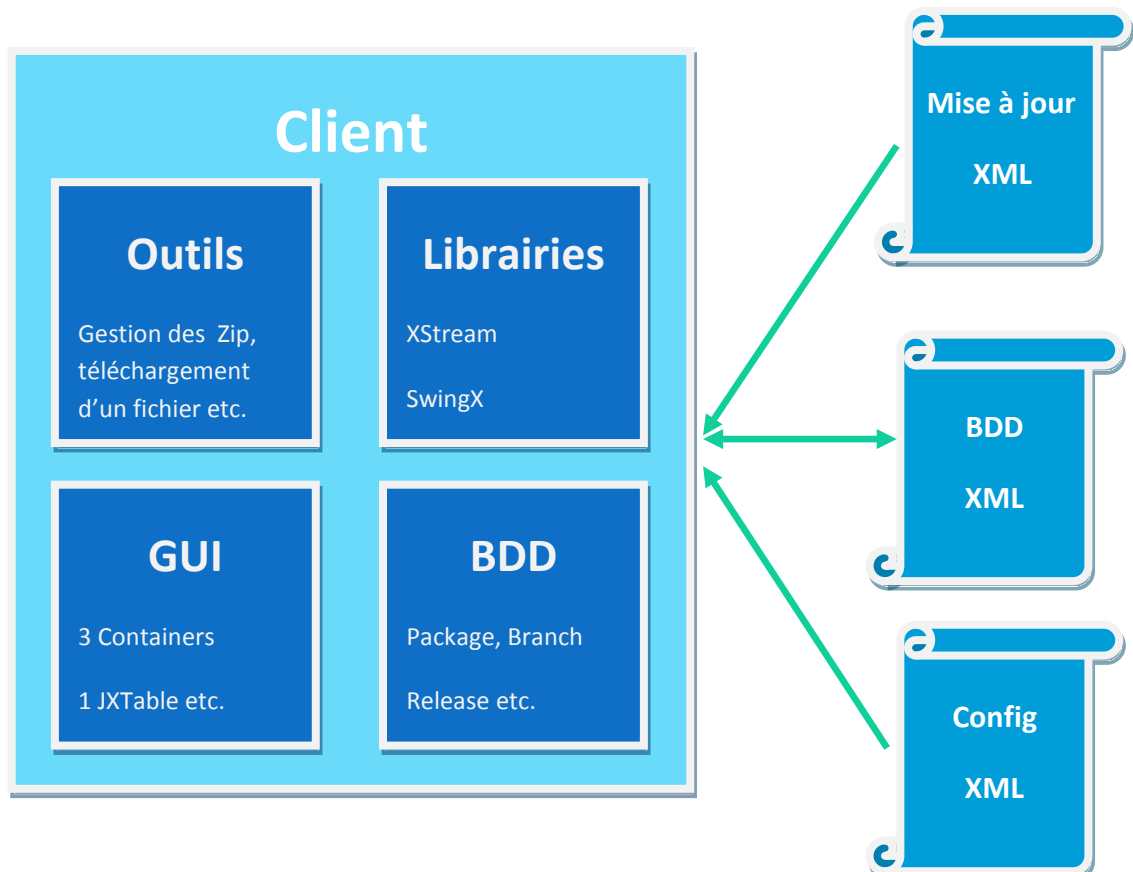
Pour permettre un certain confort dans l'ajout de relations entre les branches, nous avons utilisé l'Ajax. Ceci nous permet de recharger dynamiquement la liste des relations sans recharger entièrement la page d'édition de branche. Cela nous permet aussi de charger de manière dynamique les branches correspondantes au package que l'utilisateur a choisi et vers lequel il souhaite rajouter une dépendance.

## ORGANISATION DES PAGES

Mis à part le contrôleur générant la première page (welcome), chaque contrôleur produit seulement une partie de la page qui sera ensuite affichée dans la div XHTML 'content'. Chaque page doit donc être fait dans cette optique, et chargée en utilisant la fonction JavaScript « update\_content ».

## IMPLEMENTATION DU CLIENT JAVA

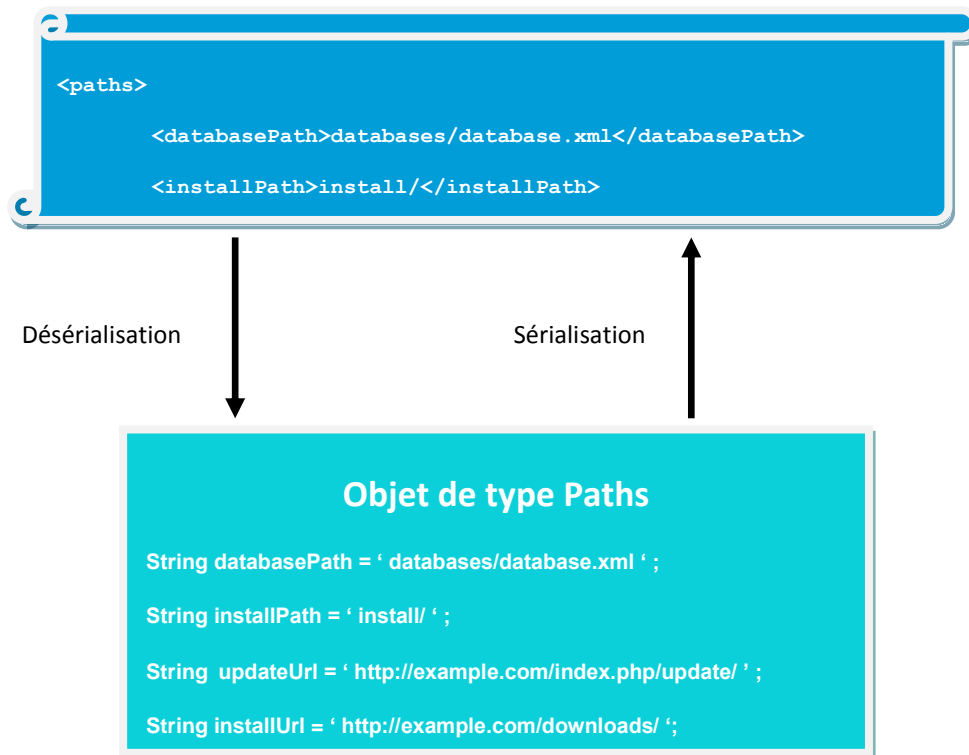
### STRUCTURE DU CLIENT



## UTILISATION DE L' XML

Pour répondre à notre besoin de stocker les informations sur les releases coté client, nous avons choisi le langage XML, pour sa flexibilité et sa popularité.

La librairie XStream, pour la (dé)sérialisation XML, s'est rapidement imposée grâce à sa simplicité d'utilisation et sa légèreté.



## UTILISATION DE JXTABLE

JXTable est une version plus étendue des JTables. Elle possède un système de filtres, de permutation dynamique des colonnes etc... Elle nous permet donc d'afficher la liste des releases de manière plus agréable.

La classe JXTable dépend de la librairie SwingX.

Name	Type	Description	Branch Name	Version	State	Category	Install
bezique	game	a great game	1.x	1.0	active	stable	not installed
belote	game	a great game	1.x	1.0	active	stable	not installed
poker	game		1.x	1.0	active	stable	not installed
testpkg	test	This is a test	testbch	1.0	active	development	not installed
tarot	game	An other Tarot game	2.x	2.1	active	stable	not installed
tarot	game	An other Tarot game	1.x	1.1	active	development	not installed
tarot	game	An other Tarot game	1.x	1.0	active	stable	not installed

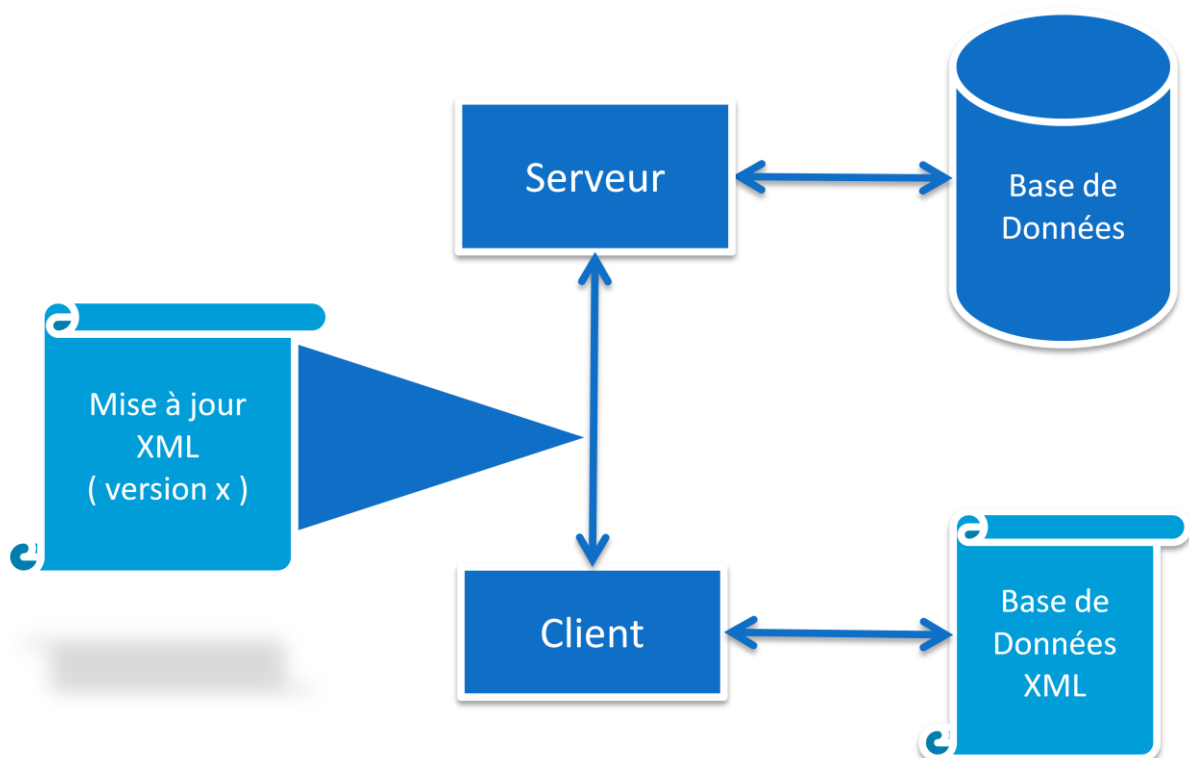
## INSTALLATION D'UNE RELEASE

Lorsque l'on souhaite installer une release, on télécharge le fichier « .cpkg » correspondant sur le serveur à l'adresse de la forme :

**<installUrl> / <package\_ID> / <branch\_ID> / <release\_ID> / <package\_Name> <version> .cpkg**  
(Ce fichier « .cpkg » est un simple fichier zip.)

Ce fichier est par la suite dézippé et placé dans le répertoire **installpath**.

## PROCESSUS D'UPDATE



1. Le client effectue une requête à l'url :  
**<updateUrl> / <version>**
2. Le serveur traite la demande :  
Il recherche dans la base de données SQL toutes les informations possédant une version comprise entre la version demandée et la version courante (coté serveur).  
Un fichier XML est généré regroupant toutes ces informations.
3. Le client récupère ce fichier de mise à jour (écrit en XML) et le déserialise en objets Java.
4. La base de données Java est mise à jour puis sérialisée en XML puis finalement enregistrée à l'emplacement : **<databasePath>** .

## POINTS A RAJOUTER

Réaliser en trois semaines de projet l'intégralité des fonctionnalités du cahier des charges initial était chose impossible en cela que nous n'avions pas assez de temps. Bien que les fonctionnalités de bases soient implémentées, de nombreuses autres parties du projet initial devront être réalisées ultérieurement.

## LA MODERATION

Tout d'abord, une des grosses parties encore non implémentée est la partie modération en général. Son principe est le suivant : lors d'un ajout d'une entité sur le dépôt, celle-ci est placée dans une file de modération qui apparaîtra sur la page d'accueil des modérateurs et super-modérateurs. Ces ajouts ne seront pas disponibles pour les utilisateurs jusqu'à ce qu'un modérateur les valide. Cela permettra d'éviter l'ajout de contenu à caractère illégal ou ne respectant pas la charte du dépôt.

Une fois la release téléchargée pour tests, le modérateur aura la possibilité de la valider ou de la supprimer du dépôt, en y ajoutant dans ce cas là les raisons de ce rejet, qui seront envoyées par mail au responsable du paquet.

## LES FLUX RSS

Une autre fonctionnalité que nous avons prévue de rajouter est celle de la gestion de flux RSS permettant la diffusion d'informations concernant le dépôt, par exemple un flux dédié à lister les nouveaux paquets. Un utilisateur pourrait ainsi s'abonner à différents flux, afin de se tenir au courant de l'évolution du dépôt. Parmi la liste des flux disponibles, on aurait ainsi : les nouveaux paquets, les commentaires d'un paquet particulier, les articles postés sur le dépôt, etc ...

## LA PARTIE VISITEUR

Pour le moment, notre projet consiste uniquement en une interface d'administration du dépôt, et est composé en majorité de PHP ainsi que de JavaScript. Cependant, du fait de l'utilisation de cette technologie, notre gestionnaire de dépôt n'est pas adapté à un référencement optimum. En effet, les robots indexeurs des moteurs de recherche (par exemple Google), ne peuvent parcourir que la première page de notre site et ne peuvent accéder aux diverses informations contenues dans nos pages, car ils ne sont pas conçus pour suivre le JavaScript.

Dans cette optique, nous avons décidé la mise en place futur d'une partie visiteur entièrement en HTML+PHP (ou avec le moins de JavaScript possible), afin de permettre de maximiser notre potentiel de référencement. En effet, les différents robots auront la possibilité de parcourir l'intégralité des données présentes dans nos pages. De plus, cela permettra d'augmenter la compatibilité avec les différents navigateurs Web, certains gérant très mal l'Ajax présent dans notre partie administration (comme Opéra). Ainsi, notre dépôt pourra être accessible au plus grand nombre, seule la partie administration proposant des effets Ajax.

## AMELIORATION GENERALE DU DESIGN ET DE L'ERGONOMIE

Dans son état actuel, notre gestionnaire de paquets possède un look clairement orientée Web 2.0, avec des couleurs pastel et un design général sobre et épuré. Cependant, trois semaines étant un délai assez court, nous n'avons pas trop porté notre attention sur l'ergonomie et le design du site. Ainsi, pour le moment le design n'est pas très travaillé, et l'ergonomie reste quelque peu à revoir. Certains menus gagneraient en lisibilité en étant placés à un autre endroit par exemple ...

## SYSTEME DE COMMENTAIRES ET DE NOTES

Une autre amélioration significative de notre gestionnaire de dépôt que nous envisageons d'implémenter dans un avenir proche est celui de la gestion des commentaires ainsi que des notes des différents paquets déposés. En effet, pour le moment, l'utilisateur désirant télécharger une release n'a accès à aucune information extérieure concernant le paquet. Avec ce système, n'importe quel utilisateur aurait la possibilité de mettre une note à un paquet qu'il aurait téléchargé, ainsi qu'une appréciation globale sur le paquet à travers les commentaires. Ainsi, cela permettrait à un utilisateur désirant télécharger cette version de se faire une idée de la qualité du paquet à travers les appréciations des autres utilisateurs.

## LOCALISATION

Dans l'état actuel des choses, notre dépôt est uniquement disponible en anglais. Or, afin de toucher le plus grand nombre, nous projetons dans un futur proche d'internationaliser notre site web afin de permettre l'utilisation de plusieurs langues différentes, que l'utilisateur pourra choisir de manière dynamique à partir de la page d'édition de son profil utilisateur.

## SYSTEME DE GESTION DE TAGS

Les systèmes à base de tags (comme les nuages de tags) sont passés à la mode et sont aujourd'hui assimilés au Web 2.0. Qu'est-ce qu'un nuage de tags ? C'est en fait un ensemble de mots-clés, où chaque mot possède une taille d'écriture différente fonction de son importance et du nombre de fois où celui-ci a été cliqué. Le tout étant bien sûr en perpétuelle évolution. Ainsi, on remarque d'un simple coup d'œil le sujet « à la mode » et le plus consulté, car celui-ci sera le plus gros sur la page. On pourra donc accéder à un paquet en cliquant sur un mot-clé s'y rapportant. Cette gestion par tags peut donc être opposée à celle par arbre. Nous envisageons d'insérer ce système sur le dépôt afin de permettre aux utilisateurs de repérer très facilement quels paquets sont les plus à la mode à chaque instant.

## STATISTIQUES

Une autre fonction nécessaire à la gestion d'un dépôt de paquets est bien sûr tout le système de statistiques. En effet, il peut être utile de connaître à tout moment le nombre de téléchargements de tel ou tel paquet, le nombre de commentaires et la note de chaque version, le nombre de visiteurs, etc ... Ceci n'a pas encore été implémenté faute de temps mais le sera dans la prochaine révision de Cero Repository.

## JOURNALISATION DES ACTIONS

Une autre fonctionnalité qu'il serait intéressant d'implémenter est celle de la journalisation des actions effectuées sur chaque élément du dépôt (par exemple paquets, branches, versions, articles ...). Cela permettrait entre autres d'avoir une trace des modifications apportées sur ces éléments et ainsi savoir par exemple qui a modifié tel paquet à tel heure.

## PERSONNALISATION DES DEPOTS

Dans le cas de la modération par exemple, il serait pratique de pouvoir créer un « sous-dépôts ne contenant dans ce cas que les dépôts devant être modérés. Cela permettrait à un modérateur désirant apporter sa contribution au dépôt de ne voir que les paquets l'intéressant dans sa fonction de modérateur. Cette fonctionnalité de création de dépôts personnalisés gagnerait donc à être implémentée dans le futur.

## 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 PHP, le SQL 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.

La réalisation de Cero Repository nous a également permis d'apprendre à utiliser une nouvelle architecture de développement d'application web, le modèle 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 cette expérience.

# NOTICE D'UTILISATION

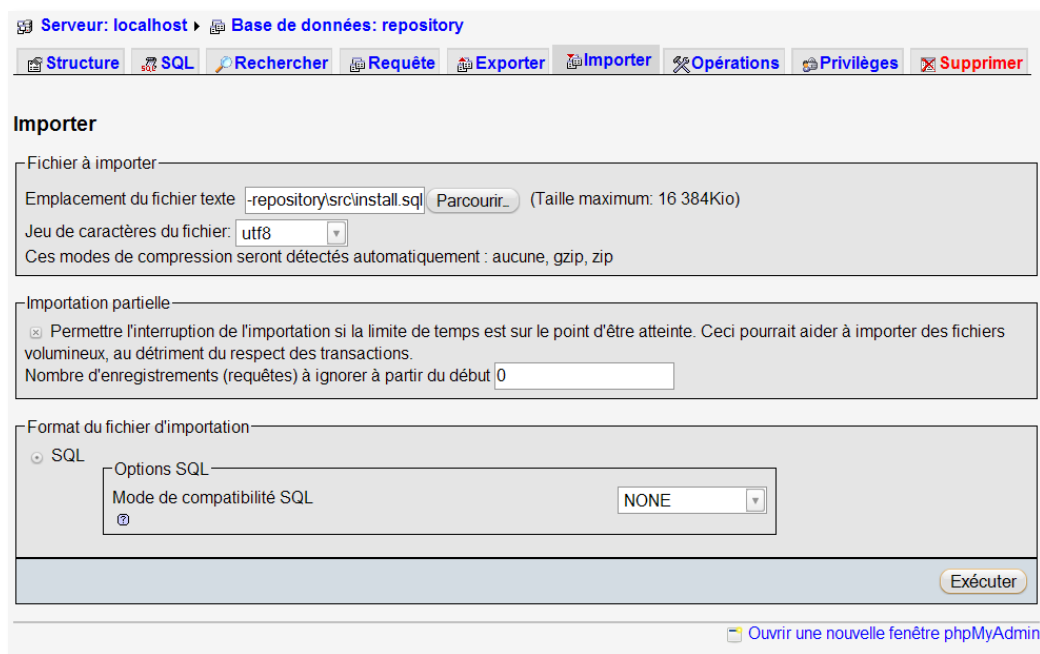
## INSTALLATION DU DEPOT

Cero Repository nécessite la configuration matérielle et logicielle suivante :

- Un serveur supportant le PHP5
- Une base de données MySQL
- Un serveur FTP pour mettre le site en ligne (optionnel, on peut utiliser SSH par exemple)

L'installation de Cero Repository a été simplifiée au maximum. En effet, celle-ci est composée de 3 étapes :

- La modification du fichier de configuration
- La copie des fichiers sur le serveur
- L'installation de la base de données



Tout d'abord, il vous faut éditer le fichier de configuration du dépôt qui se trouve à l'emplacement `\src\repository\application\config\config.php` avec les informations relatives à votre nouveau dépôt, en particulier la ligne commençant par `$config['base_url']`. Vous devez ensuite éditer le fichier `\src\repository\application\config\database.php` en y précisant les paramètres de votre base de données comme cela y est indiqué.

Une fois la partie configuration effectuée, copiez par l'intermédiaire d'un client FTP l'intégralité des fichiers de la distribution sur le serveur WEB.

Pour finir, dans phpMyAdmin, commencez tout d'abord par créer une nouvelle base de données appelée « repository »<sup>1</sup>. Une fois sur la page de la base « repository », cliquez sur l'onglet « Importer » puis sélectionnez le fichier « install.sql » situé à la racine de la distribution et cliquez sur « Exécuter ». Cela créera les différentes tables nécessaires au dépôt ainsi qu'un premier utilisateur, admin ayant comme mot de passe admin, et ayant tous les droits sur le dépôt.

Voilà, vous venez de terminer l'installation du dépôt.

<sup>1</sup> Si le nom est différent, pensez à le préciser dans le fichier de configuration « database.php ».

### SITE WEB

#### UTILISATEURS :

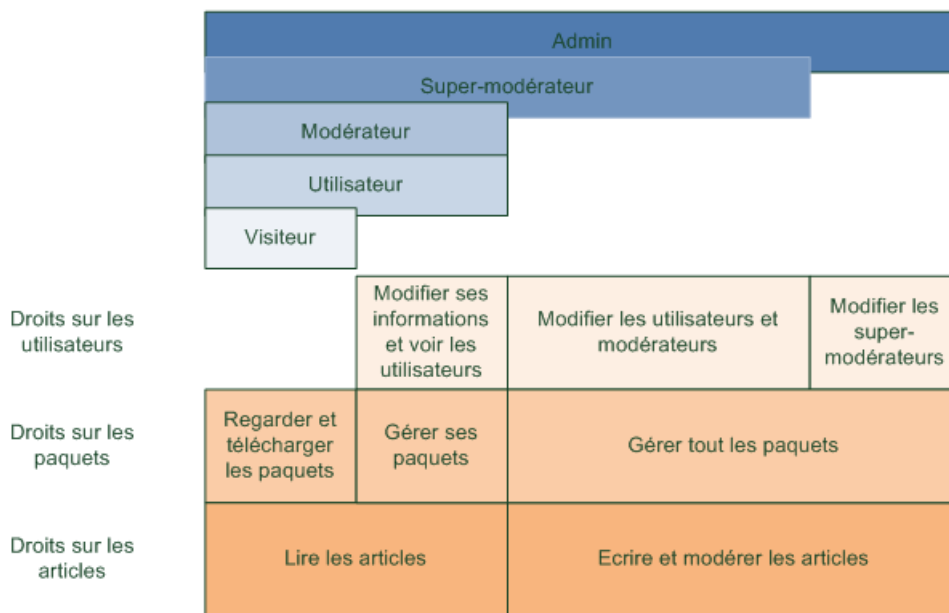
A l'installation du site, vous disposez d'un compte « admin » ayant pour mot de passe « admin ». Cet utilisateur a tous les droits sur le dépôt, et peut donc attribuer des droits aux autres utilisateurs.

Il y a 5 niveaux d'utilisateurs :

- **Admin** : n'a aucune restriction
- **Super-modérateur** : a pour seule restriction de ne pouvoir désigner de nouveaux super-modérateurs et de ne pouvoir modifier l'administrateur. Ce rôle a été initialement créé pour isoler le compte administrateur qui pourra, dans une prochaine version, configurer le site à travers quelques pages spécifiques. Le super-modérateur est donc un administrateur « restreint ».
- **Modérateur** : la fonctionnalité de modération n'ayant pas encore été développée, il ne dispose actuellement d'aucun droit particulier.
- **Utilisateur** : il peut créer des paquets et modifier ses informations. Il peut de plus voir les informations des autres utilisateurs.
- **Visiteur** : il ne peut que regarder et télécharger les paquets, et lire les articles.

En plus de ces droits spécifiques, chaque niveau d'utilisation a les droits du niveau d'utilisation inférieur : par exemple, un utilisateur peut lire les articles, un super-modérateur peut créer des packages.

### Niveaux d'utilisateurs



## CREATION DE COMPTE

En cliquant sur « Register » et en entrant des informations valides, un nouvel utilisateur pourra s'enregistrer. A partir de ce moment, il pourra se connecter au site.

## CONNEXION

**NB : cette fonctionnalité nécessite l'activation des cookies sur le browser de l'utilisateur.**

En entrant son pseudonyme (login) et son mot de passe (password), un utilisateur se connecte au site. Avant d'envoyer ses informations de connexion, il peut choisir de cocher la case « Remember Me », qui permettra au site de le reconnaître pendant une semaine dès qu'il reviendra dessus. Sans cocher cette case, il ne sera reconnu par le site que pendant les 2 heures suivant sa dernière activité.

Une fois connecté, il est redirigé sur sa page d'accueil, où sont listés les articles qu'il a rédigé s'il en a, et de même pour les paquets qu'il possède s'il en a. C'est sur cette page que devra donc s'afficher toutes les informations dont l'utilisateur a spécifiquement besoin en cas d'extensions (par exemple la modération). On pourrait aussi afficher les nouvelles liées à certains paquets auxquels l'utilisateur peut « s'abonner ».

## PAQUETS

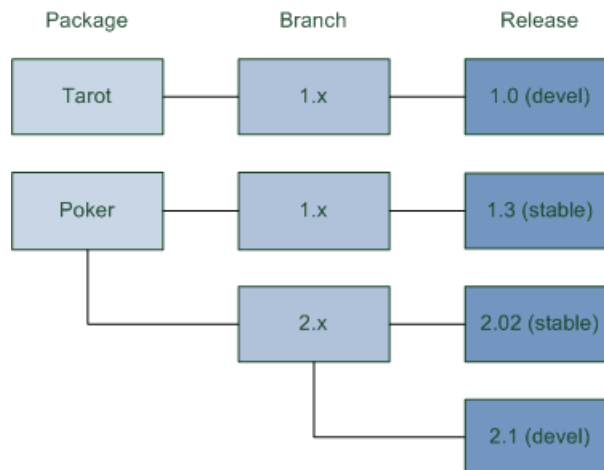
Un paquet (package) correspond à un logiciel. Celui-ci se déclinera en plusieurs branches selon les différentes implémentations de ce logiciel. Enfin, chaque branche se décomposera en versions (releases) qui seront des états différents du logiciel au cours de son développement.

Les branches sont destinées à regrouper les versions du logiciel qui sont rétro-compatibles : une version X du logiciel est rétro-compatible avec la version Y si elle remplit les mêmes interfaces que la version Y. Ceci signifie

donc que les versions successives d'une branche peuvent ajouter des fonctionnalités au logiciel, corriger des bugs, mais ne peuvent pas modifier le comportement externe du logiciel. Si les développeurs décident d'en faire autrement, il est alors de leur responsabilité de veiller à ce que tous les programmes dépendants dudit logiciel soient convenablement mis à jour.

Pour chaque branche, seulement deux versions peuvent être stockées simultanément : une version stable et une version en développement.

### Exemple d'arborescence de packages



## TELECHARGEMENT

Lorsqu'un visiteur veut télécharger un logiciel, il commence donc par sélectionner le paquet, puis la branche qu'il désire, et enfin la version qu'il préfère (stable ou en développement). Il obtient alors un fichier d'extension cpkg, qui est un fichier de type zip contenant la distribution du logiciel. Ce fichier cpkg est aussi destiné à contenir à terme des informations supplémentaires (Métadonnées), afin de faciliter l'installation par le client Java ou un autre client adapté.

## RELATIONS INTER-PAQUETS

Il est possible de définir des relations entre les paquets, par exemple si un logiciel dépend d'un autre. Les dépendances et la plupart des relations étant des contraintes liées à la version du logiciel, les dépendances sont donc situées entre les branches. On n'est par exemple pas sûr que la version 2 d'un logiciel utilise les mêmes bibliothèques (et donc les mêmes dépendances) que la version 1.

Les types de relation actuellement proposées sont :

X 'depends on' Y version Z : Indique que la branche X nécessite la version Z ou supérieure de la branche Y pour fonctionner.

X 'conflicts' Y : Indique que la branche X ne peut être installée en même temps que la branche Y.

X 'recommends' Y : Indique que la branche X recommande la branche Y. Ceci peut être utilisé pour se promouvoir mutuellement, pour qu'une ancienne branche d'un logiciel (1.x) puisse indiquer l'existence d'une branche améliorée (2.x).

Cependant, il n'existe actuellement qu'une vérification minimale de la cohérence des dépendances : un paquet ne peut se mettre en relation avec lui-même, mais il est possible de définir des dépendances cycliques, des conflits avec les paquets dont on dépend, etc. On laissera incomber au programme d'installation la vérification des conflits éventuels, le dépôt se chargeant uniquement de la distribution.

## NAVIGATION DU SITE

---

Afin de dynamiser le site, le parcours du site se fait à l'aide de la technologie Ajax : celle-ci fait que l'on ne change jamais de page. Les visiteurs devront donc prendre conscience qu'il n'est pas possible d'utiliser les boutons « précédent » et « suivant » du site pour le naviguer. Cependant, le(s) menu(s) propose(nt) un accès rapide aux pages importantes, ce qui permet un parcours aisé.

## SERVICE DE MISE A JOUR

---

Cette fonctionnalité permet d'obtenir la base de données des paquets, des branches et des releases sous forme XML. C'est grâce à celle-ci que le client Java se mettra à jour. En appelant la page « update/X », on obtient le fichier XML contenant les nouveautés de la base de données depuis la version X. Pour obtenir la totalité des informations, on doit passer X = -1.

## CLIENT JAVA

### CONNEXION AU DEPOT

---

Le client Java offre la possibilité de mettre à jour et installer des logiciels provenant du dépôt sans passer par l'interface du site web. Celui-ci se connecte au service de mise à jour du dépôt pour obtenir les informations lui manquants, et complète ainsi une base de données locale.

### INTERFACE GRAPHIQUE

---

C'est à partir du client que l'utilisateur pourra rechercher les paquets qu'il désire et les installer sur sa machine. Tout ceci se fait à partir d'une interface graphique complète, permettant de filtrer la liste des paquets selon différents critères : type de release, recherche par nom, paquets déjà installés ou pas, etc.

### INSTALLATION DES PAQUETS

---

L'installation ou la mise à jour consiste actuellement à dézipper le contenu du fichier cpkg téléchargé dans le dossier *install*. Cependant, cette méthode devra être adaptée pour chaque programme, par exemple en exploitant les futures Métadonnées. Le client ne gère actuellement pas les dépendances, mais il s'agit d'une des priorités dans la continuité du projet.

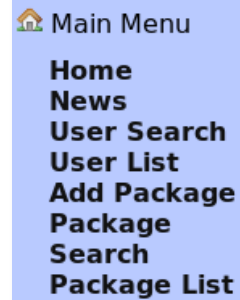
## SCENARIOS D'UTILISATION

### APPLICATION WEB

#### NAVIGATION EN TANT QUE VISITEUR

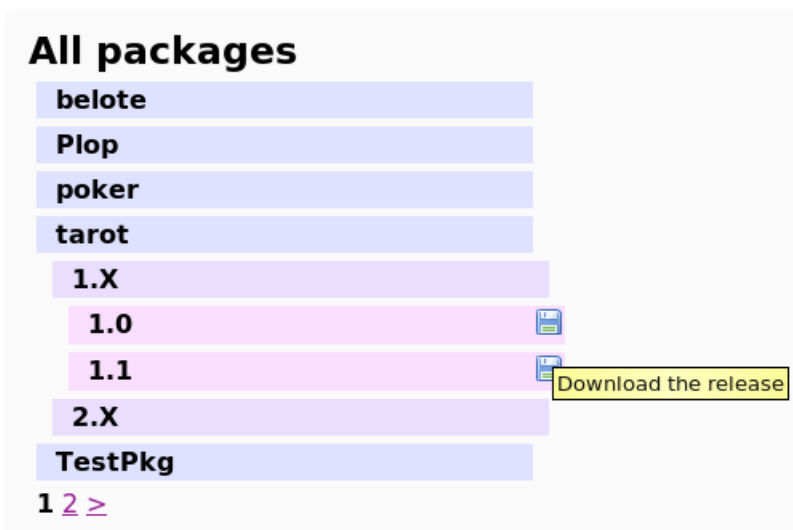
##### OPERATIONS BASIQUES

- Accès à l'accueil (Home)
- Accès aux nouvelles (news)
- Recherche de paquets (Package Search)
- Accès à la liste des paquets (Package List)



##### TELECHARGEMENT D'UNE RELEASE

- Accès au paquet voulu, soit par une recherche, soit à partir de la liste.
- Un clic sur le paquet souhaité donne accès aux branches de ce dernier.
- Un clic sur la branche souhaitée donne accès aux versions de cette branche (une stable et une en développement)
- Un clic sur l'icône en forme de disquette permet de télécharger la release.



##### CREER UN UTILISATEUR

- Un clic sur "Register" donne accès au formulaire d'inscription.
- Remplir et valider le formulaire (un mail vous sera adressé pour vous confirmer votre inscription).

<b>Nickname :</b>	<input type="text" value="linus"/>
<b>First Name :</b>	<input type="text" value="linus"/>
<b>Last Name :</b>	<input type="text" value="torvald"/>
<b>E-mail :</b>	<input type="text" value="linus.torvald@example.org"/>
<b>Password :</b>	<input type="password" value="*****"/>
<b>Password (check) :</b>	<input type="password" value="*****"/>

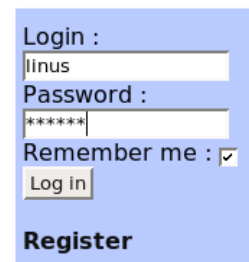
---

## NAVIGATION EN TANT QU'UTILISATEUR

### SE CONNECTER

---

- Tapez votre pseudo et votre mot de passe dans la vue d'authentification, puis validez. (Vous pouvez préciser si vous souhaitez une connexion automatique à la prochaine visite)
- Une fois connecté, vous pouvez éditer votre profil si vous le désirez.



Login :  
  
Password :  
  
Remember me :   
  
**Register**

### OPERATIONS BASIQUES

---

- Accès à la liste des utilisateurs (User List)
- Recherche d'utilisateurs (User Search)
- Accès à sa page personnelle (My Homepage)
- Accès à son profil (My Profile)
- Se déconnecter (Logout)



 **User Menu**  
**My Homepage**  
**My Profile**  
**Logout**

### SOUMETTRE UNE RELEASE


---

#### CREATION D'UN PAQUET

- Un clic sur "Add Package" donne accès au formulaire de création de paquet.
- Remplissez puis validez le formulaire.
- Le paquet créé est rajouté dans la liste générale des paquets et dans la liste de vos paquets sur votre page personnelle (My Homepage).


**Name :**   
**Website :**   
**Licence :**   
**Category :**    
  
**Short description :**

## CREATION D'UNE BRANCHE




- Un clic sur l'icône  présente sur le paquet créé donne accès au formulaire de création de branche.
- Remplissez puis validez le formulaire.
- La branche créée est affichée sous le paquet auquel elle appartient. (Après avoir cliqué sur le paquet).

**Welcome to Cero Repository, linus !**

You are an User

 [Edit your profile](#)


**Your packages**

**Bezique**   



**Add Branch**

**Name :**   
  
**Description :**

## CREATION D'UNE RELEASE

- Un clic sur l'icône  présente sur la branche créée donne accès au formulaire de création de release.
- Remplissez puis validez le formulaire.
- La release a été soumise, elle est maintenant présente sous la branche à laquelle elle appartient (après avoir cliqué sur la branche).

**Your packages**

**Bezique**   


**1.X**   

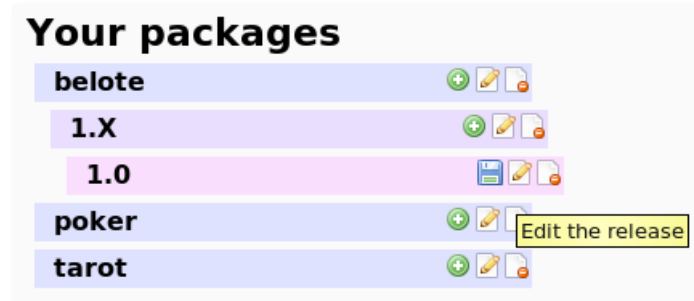
**No releases for this branch**

**Add Release**


**Version :**   
  
**Release Note :**   
**Type :**   
**Release :**

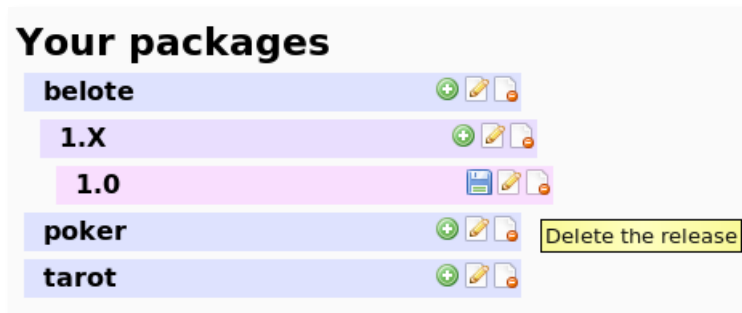
## EDITION DE PACKAGE/BRANCH/RELEASE

- Un clic sur l'icône  présente sur le paquet, la branche ou la release permet d'éditer l'élément correspondant. (On ne peut éditer que les éléments qui nous appartiennent).



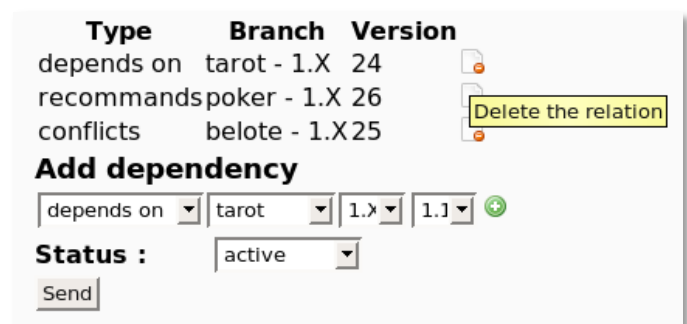
## SUPPRESSION DE PACKAGE/BRANCH/RELEASE

- Un simple clic sur l'icône  présente sur le paquet, la branche ou la release permet de supprimer l'élément correspondant. (On ne peut supprimer que les éléments qui nous appartiennent)



## GESTION DES RELATIONS

- Editez la branche qui est destinée à avoir une relation avec d'autres releases.
- Une section "Add dependency" est présente sur le formulaire, choisissez un type de relation, puis un paquet dans la liste.
- Sélectionnez une branche puis une version dans le champ qui apparait.
- Validez le formulaire.



Vous pouvez supprimer des relations de la liste en cliquant sur l'icône "avec un point rouge sur une feuille" correspondant à l'élément à supprimer.

---

## NAVIGATION EN TANT QUE MODERATEUR

La partie modération n'étant pas encore implémentée dans la version actuelle du projet, il ne nous est pas possible de détailler plus précisément les fonctionnalités proposées à ce type d'utilisateur.

---

## NAVIGATION EN TANT QUE SUPER-MODERATEUR/ADMINISTRATEUR

---

### OPERATIONS BASIQUES

---

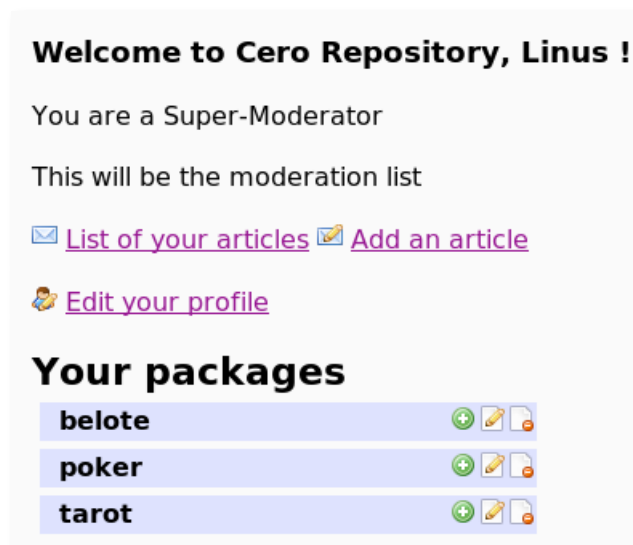
- Editer les droits des utilisateurs ayant des droits inférieurs
- Afficher la liste des articles qu'il a soumis (List of your articles)
- Editer les articles

---


### AJOUTER UN ARTICLE

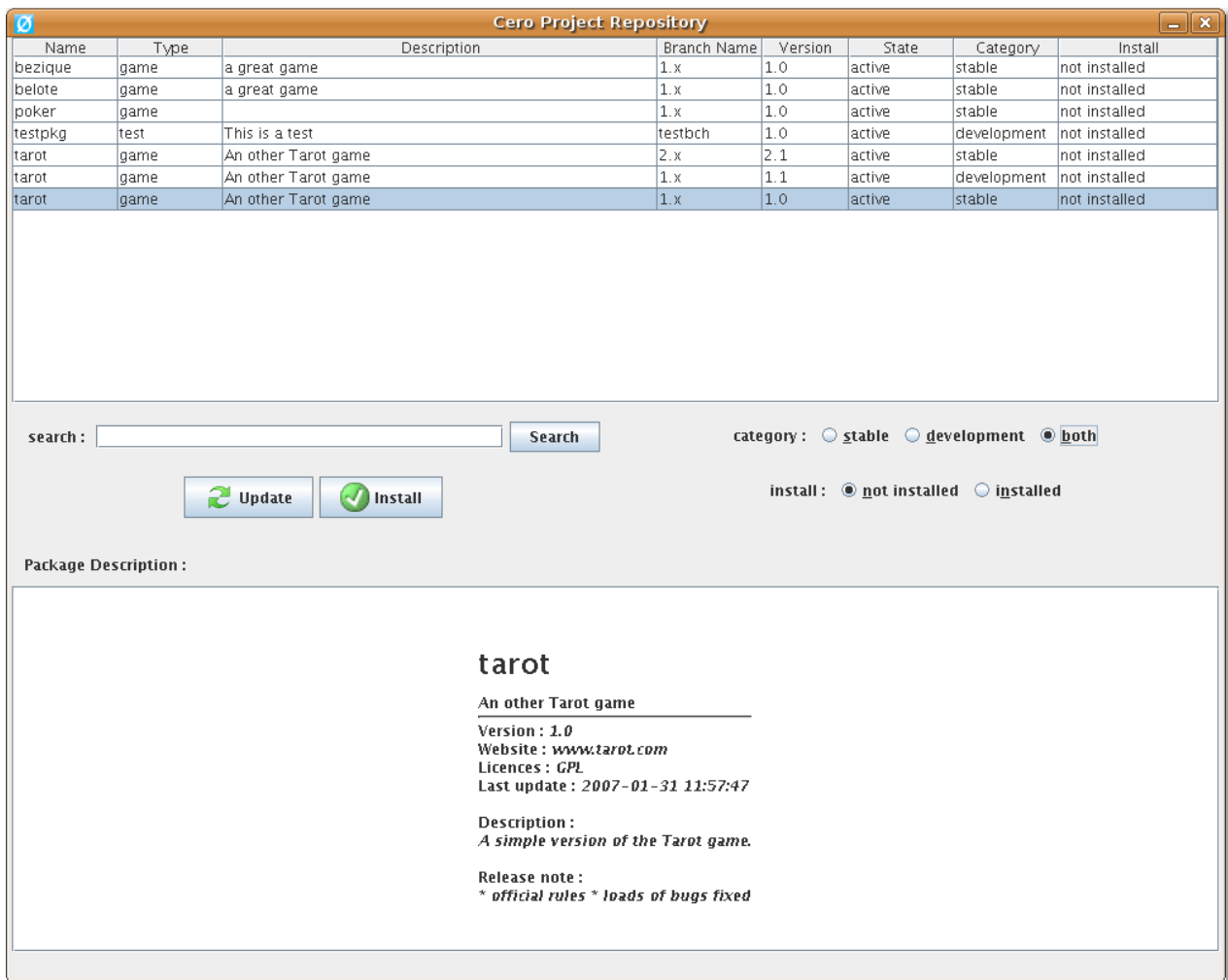
---

- Un clic sur ajouter un article (Add an article) donne accès au formulaire de création d'article.
- Remplissez puis validez le formulaire.



## MISE A JOUR DE LA LISTE DES RELEASES DISPONIBLES

- Un simple clic sur le bouton  suffit.





The screenshot shows the 'Cero Project Repository' window. At the top, there is a table listing several packages. Below the table, there is a search bar and several control buttons. The 'tarot' package is selected, and its details are shown in a separate pane below.

Name	Type	Description	Branch Name	Version	State	Category	Install
bezique	game	a great game	1.x	1.0	active	stable	not installed
belote	game	a great game	1.x	1.0	active	stable	not installed
poker	game		1.x	1.0	active	stable	not installed
testpkg	test	This is a test	testbch	1.0	active	development	not installed
tarot	game	An other Tarot game	2.x	2.1	active	stable	not installed
tarot	game	An other Tarot game	1.x	1.1	active	development	not installed
tarot	game	An other Tarot game	1.x	1.0	active	stable	not installed

search :  Search

category :  stable  development  both

install :  not installed  installed

Package Description :

**tarot**  
 An other Tarot game  
 Version : 1.0  
 Website : [www.tarot.com](http://www.tarot.com)  
 Licences : GPL  
 Last update : 2007-01-31 11:57:47

Description :  
*A simple version of the Tarot game.*

Release note :  
*\* official rules \* loads of bugs fixed*

## RECHERCHER ET FILTRER LES RELEASES

- Tapez le début du nom du package dans la barre de recherche. La liste des releases est filtrée et n'affiche que les releases dont le nom du paquet auxquelles elles appartiennent contient l'expression tapée.

search :  Search

- Sélectionnez ensuite la catégorie et l'état des releases en cliquant sur les boutons radio correspondants pour n'afficher que les releases qui vous intéressent.

category :  stable  development  both

install :  not installed  installed

---

#### AFFICHAGE DE DESCRIPTION

- Cliquez sur la ligne du tableau représentant la release pour afficher sa description.

**tarot**

**An other Tarot game**

---


Version : 1.0  
Website : [www.tarot.com](http://www.tarot.com)  
Licences : GPL  
Last update : 2007-01-31 11:57:47

Description :  
*A simple version of the Tarot game.*

Release note :  
*\* official rules \* loads of bugs fixed*

---

#### INSTALLATION D'UNE RELEASE

- Sélectionnez la release souhaitée.
- Cliquez sur le bouton  pour installer la release sélectionnée.

### 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.

### BRANCHE

---

Une **branche** est un regroupement de versions d'un logiciel où toutes les releases sont rétro-compatibles avec les versions précédentes.

### DEPOT

---

En informatique, un **dépôt** (de l'anglais « *repository* »), est un stockage centralisé et organisé de données. Ce peut être une ou plusieurs bases de données où les fichiers sont localisés en vue de leur distribution sur le réseau, ou bien un endroit directement accessible aux utilisateurs.

### 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.

## PAQUET

---

En informatique, on désigne par le terme **paquet** (en anglais *package*) une archive (parfois compressée) comprenant des fichiers informatiques, ainsi que les informations et procédures nécessaires à son installation.

## PHP

---

Le PHP, ou **Hypertext Preprocessor**, est un langage de programmation web principalement utilisé pour être exécuté par un serveur HTTP. Il possède aussi une structure d'objets.

## 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.

## REST

---

**REST (Representational state transfer)** est une manière de construire une application pour les systèmes distribués comme le World Wide Web. REST n'est pas un protocole ou un format, c'est une architecture, c'est l'architecture originale du Web, bâtie sur quelques principes simples :

- l'URI est important : connaître l'URI doit suffire pour accéder à la ressource
- HTTP fournit toutes les opérations nécessaires (GET, POST, PUT et DELETE, essentiellement)
- chaque opération est auto-suffisante : il n'y a pas d'état
- utilisation des standards hypermédia : HTML ou XML

C'est donc une alternative à RPC et SOAP.

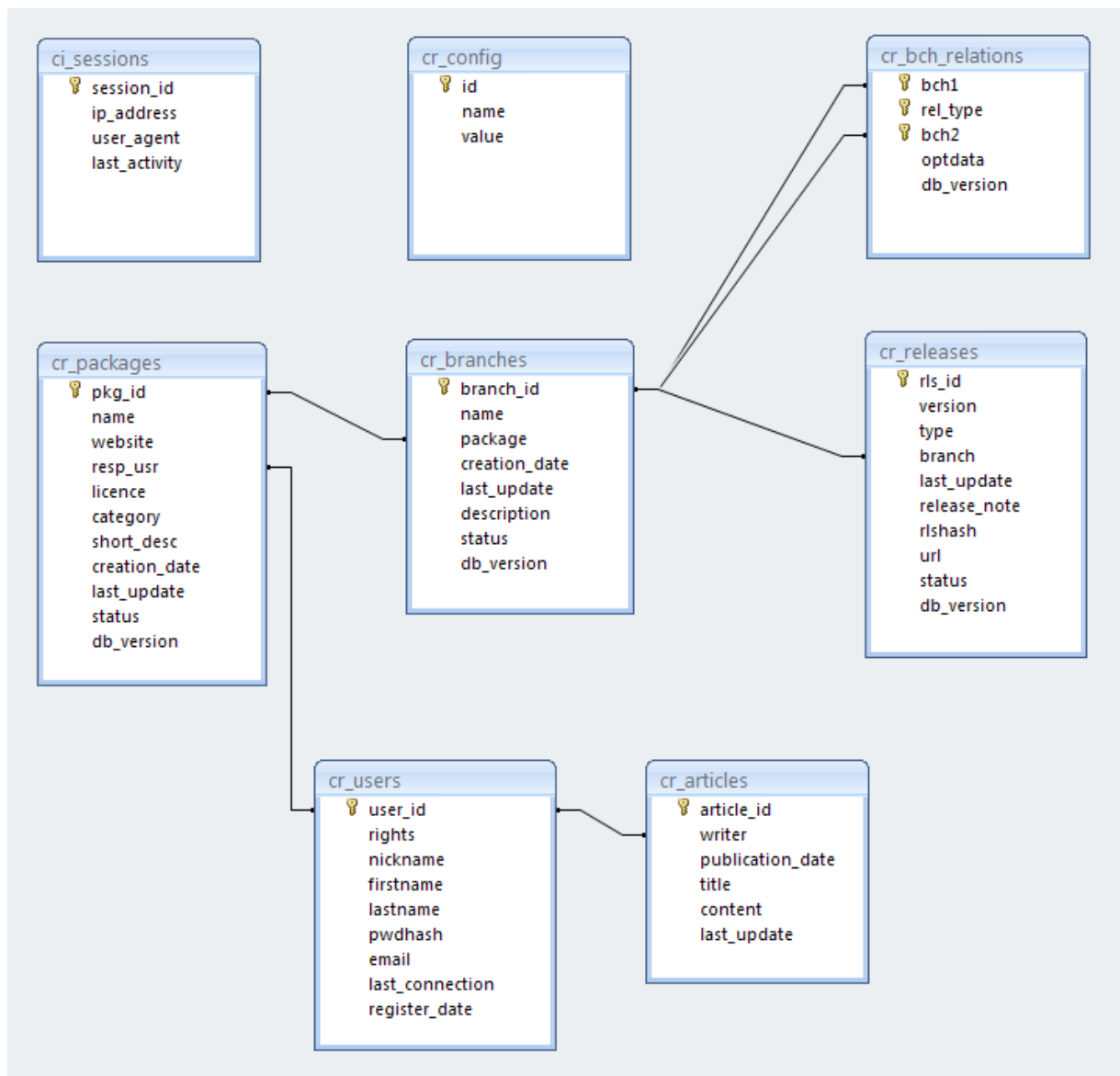
## SERIALISATION

---

En informatique la **sérialisation** (ou *marshalling* en anglais) est un processus visant à encoder l'état d'un objet qui est en mémoire sous la forme d'une chaîne d'octets dans un flux de données. Cette chaîne d'octet pourra par exemple être utilisée pour la sauvegarde sur disque (persistance) ou le transport sur le réseau. L'activité inverse, visant à décoder la suite d'octets pour créer une copie conforme des objets d'origine, s'appelle la désérialisation (ou *unmarshalling*).

## STRUCTURE DE LA BASE DE DONNEES

Voici la structure de la base de données utilisée dans Cero Repository.



```

- <database>
  <version>152</version>
- <packages>
  - <entry>
    <string>22</string>
    - <package>
      <pkgid>22</pkgid>
      <name>test</name>
      <website>test</website>
      <respusr>13</respusr>
      <licence>test</licence>
      <category>Game</category>
      <shortdesc>test</shortdesc>
      <creationdate>2007-02-01</creationdate>
      <lastupdate>2007-02-01 20:26:03</lastupdate>
      <status>active</status>
    - <branches>
      - <entry>
        <string>22</string>
        - <branch>
          <branchid>22</branchid>
          <name>3.X</name>
          <pck>22</pck>
          <creationdate>2007-02-01</creationdate>
          <lastupdate>2007-02-01 20:29:53</lastupdate>
          <description>test</description>
          <status>deleted</status>
        - <releases>
          - <entry>
            <string>35</string>
            - <release>
              <rlsid>35</rlsid>
              <version>3.1</version>
              <branch>22</branch>
              <lastupdate>2007-02-01 20:29:45</lastupdate>
              <releasenote>test</releasenote>
              <rlshash>81ef093be333cf995d827ae8abac219b</rlshash>
              <status>deleted</status>
              <state>devel</state>
            </release>
          </entry>
          + <entry></entry>
        </releases>
      </branch>
    </entry>
  </branches>
</package>
</entry>
</packages>
</database>

```

### MOOTOOLS

---

Site officiel : <http://mootools.net/>  
Documentation : <http://docs.mootools.net/files/Moo-js.html>

**Taille** : 5 à 27ko (javascript compressé)

**Licence** : Open Source MIT License

### AVANTAGES

---

- “Fonctions-raccourci” (ex : `$()`)
- Documentation claire
- Léger

### INCONVENIENTS

---

- Peu d'exemples
- Encore peu utilisé

### PROTOTYPE

---

Site officiel : <http://prototype.conio.net/>  
Documentation : <http://wiki.script.aculo.us/scriptaculous/show/Prototype>

**Taille** : 46ko (possibilité de télécharger les modules séparément à partir du source)

**Licence** : Open Source MIT License

### AVANTAGES

---

- “Fonctions-raccourci” (ex : `$()`, `$$()`,...)
- Utilisé par Scriptaculous

### INCONVENIENTS

---

- Peu de fonctionnalités
- Rarement utilisé seul

---

## SCRIPTACULOUS (+ PROTOTYPE)

Site officiel : <http://script.aculo.us/>  
Documentation : <http://wiki.script.aculo.us/scriptaculous/>

**Taille** : 19 ko (nécessite Prototype)

**Licence** : Open Source MIT License

### AVANTAGES

---

- Très utilisé
- Utilisé par Digg, Apple, FeebBurner, Ruby on Rails, ...
- Permet d'utiliser très facilement des effets visuels JavaScript

### INCONVENIENTS

---

- Documentation parfois incomplète

---

## DOJO

Site officiel : <http://dojotoolkit.org/>  
Documentation : <http://dojotoolkit.org/api/>  
Dojo Book : <http://manual.dojotoolkit.org/WikiHome/DojoDotBook>

**Taille** : 146 ko (nécessite Prototype)

**Licence** : Open source BSD License

### AVANTAGES

---

- Widgets
- Contribution d'IBM
- Nombreuses fonctionnalités

### INCONVENIENTS

---

- Assez lourd (cf. taille)
- Du code contenant des warning est nécessaire pour l'utilisation des widgets (arguments propriétaires dans les balises)
- Documentation parfois incomplète

## REPARTITION TEMPORELLE DES TACHES

	Mois précédents	Semaine 1	Semaine 2	Semaine 3
Michael	Recherche Framework PHP & AJAX	Vues, contrôleurs et JavaScript	+ débogage + modèle	Rapport + Préparation soutenance
Camille				
Mathieu		Client Java		
Matthieu				