

PAQUITEAM
Équipe 4

Phase de construction 1 : Paquito, easy packaging



Auteurs :

Lucas ROBIN
Kevin FLEURIOT
Alexandre NOIRET
Yassine AIT ELMOUDEN
Seynabou KA

Clients :

Michael FORTIER
Hugues LEPRIEUR

Encadrants :

Thierry HAMON
Joseph LE ROUX

17 mars 2016

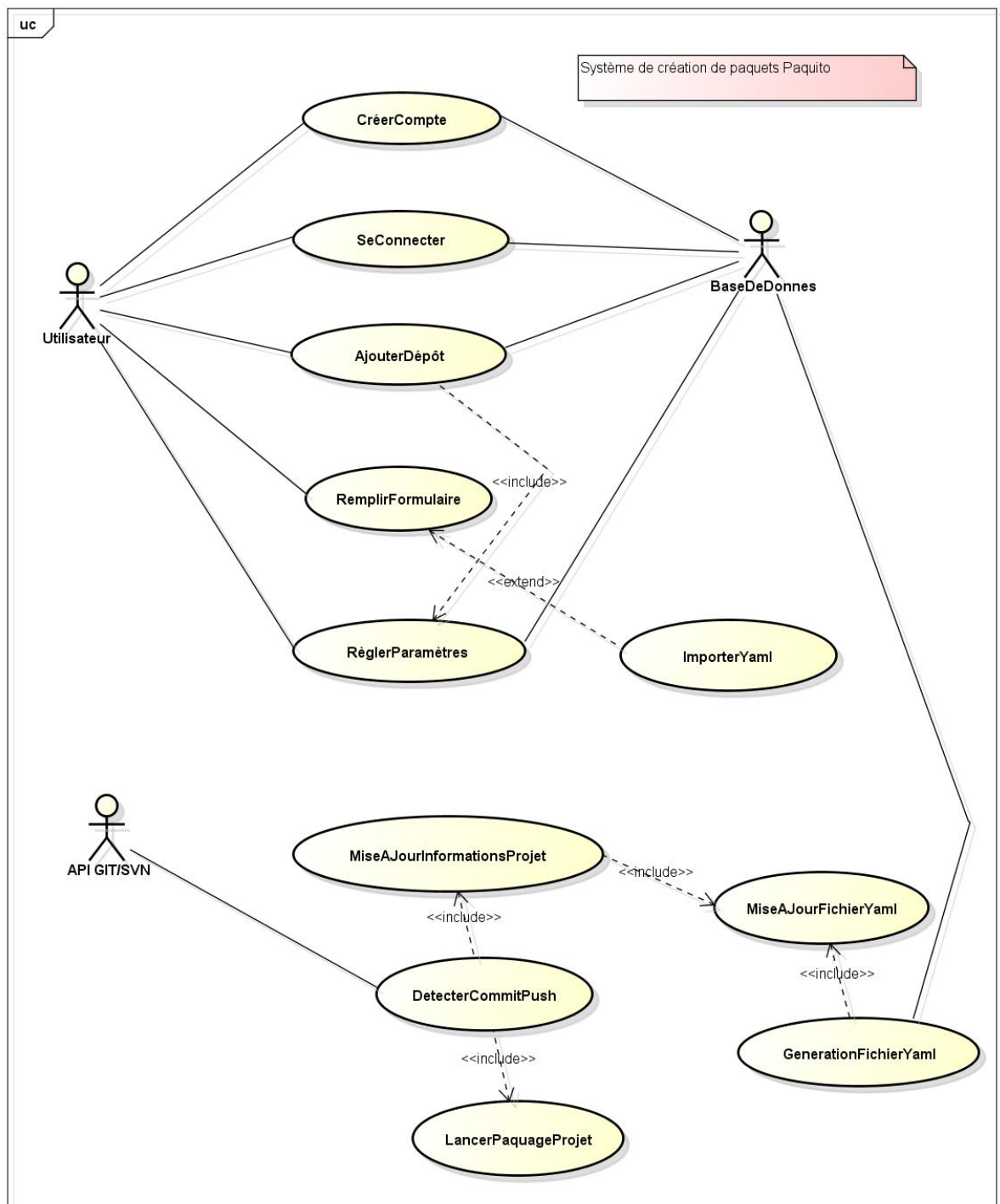


Introduction

Notre projet se porte sur l'outil Paquito, qui est un outil de génération de paquets pour un logiciel donné. Chaque paquet contient les fichiers de configuration nécessaires pour l'installation et l'utilisation du logiciel. Cet outil est utilisable pour plusieurs distributions de Linux. Le but de notre projet est donc d'améliorer l'utilisation de cet outil. Nous commencerons par transformer le programme en ligne de commande en un service web plus simple d'utilisation et plus complet qui permettra au développeur de rentrer toute ses informations dans un formulaire approprié ou en important directement un fichier YAML. Lors de nos précédents rapports, nous avons mentionné que nous devions adapter l'outil Paquito à MacOS mais à la demande du client, nous adapterons l'outil pour Ubuntu et non MacOS.

Dans cet écrit, nous allons vous exposer les différents cas d'utilisation de notre système. Puis nous présenterons la structure de notre base de données.

1 Diagramme des cas d'utilisation



2 Détails des cas d'utilisation

- **Créer un compte**

- **Identification**

- Niveau de l'action : Sous-fonction

- But : L'utilisateur (le développeur) crée un compte pour utiliser le service Web

- Acteur : Utilisateur

- **Scénario principal**

- 1. L'utilisateur renseigne ses coordonnées
 2. Le site web accepte l'inscription

- **Se connecter**

- **Identification**

- Niveau de l'action : Sous-fonction

- But : L'utilisateur (le développeur) se connecte pour utiliser le service Web

- Acteur : Utilisateur

- **Scénario principal**

- 1. L'utilisateur entre son login et son mot de passe
 2. Le site web accepte l'inscription

- **Echec**

- 1. L'utilisateur n'est pas inscrit
 2. Le site web notifie l'erreur et propose à l'utilisateur de s'inscrire
 3. L'utilisateur est inscrit mais n'a pas renseigné le bon mot de passe
 4. Le site web notifie l'erreur et propose à l'utilisateur de recommencer

- **Ajouter un dépôt**

- **Identification**

- Niveau de l'action : But utilisateur

- But : L'utilisateur ajoute un dépôt sur le service Web pour un projet.

- Acteur : Utilisateur

- **Scénario principal**

- 1. L'utilisateur indique le nom et l'URL du dépôt.
 2. Le serveur est contacté et lance la commande d'ajout de dépôt.

- **Mise à jour des informations du projet**

- **Identification**

- Niveau de l'action : Sous fonction

- But : Mettre à jour les informations du projet

Acteur : Utilisateur

— **Scénario principal**

1. L'utilisateur change les informations via l'application web
2. Les informations sont changées suite à un commit ou un push
3. Changement des informations dans la Base de Donnée
4. Génération du nouveau fichier YAML

- **Gérer paramètres de création de paquets** : On peut modifier les paramètres de création des paquets

- **Lancement du paquetage du projet**

— **Identification**

Niveau de l'action : Sous fonction

But : Créer les fichiers qui composent le paquetage et génère le fichier contenant les informations sur le paquetage

Acteur : API GIT/SVN

— **Scénario principal**

1. Les sources et le fichier YAML ainsi que d'autres informations sont envoyées, via ssh par exemple, aux Machines Virtuelles.
2. Lancement de l'opération de compilation.
3. On notifie le(s) développeur(s) de la réussite de la création du paquetage

— **Echec**

- En cas d'erreur lors de la création du paquetage on envoie une notification au(x) développeur(s) avec l'erreur retournée par Paquito

- **Détection de Commit ou de Push**

— **Identification**

Niveau de l'action : Sous fonction

But : L'API GIT/SVN détecte un changement, qui est notamment un commit ou un push.

Acteur : API GIT/ SVN

— **Scénario principal**

1. Un développeur fait un commit/push sur son dépôt.
2. Le serveur en est averti via l'API Git/Svn
3. Les informations du projet ont été mises à jour
4. Lancement de la compilation et paquetage sur les Machines Virtuelles du projet.

- **Remplir formulaire**

— **Identification**

Niveau de l'action : Sous fonction

But : L'utilisateur remplit les informations sur le projet qu'il veut faire suivre par paquito

Acteur : Utilisateur

— **Scénario principal**

1. L'utilisateur remplit tous les champs
2. L'utilisateur soumet le formulaire

— **Echec**

- Un ou plusieurs champs ne sont pas remplis

• **Importer un fichier YAML**

— **Identification**

Niveau de l'action : Sous-fonction

But : Importer un fichier YAML lorsque l'utilisateur l'a déjà en sa disposition

Acteur : Utilisateur

— **Scénario principal**

1. L'utilisateur va envoyer son fichier YAML via l'application Web
2. Le système va lire le fichier et remplir automatiquement le formulaire sur l'application Web.

3 Architecture

3.1 Description de l'architecture globale

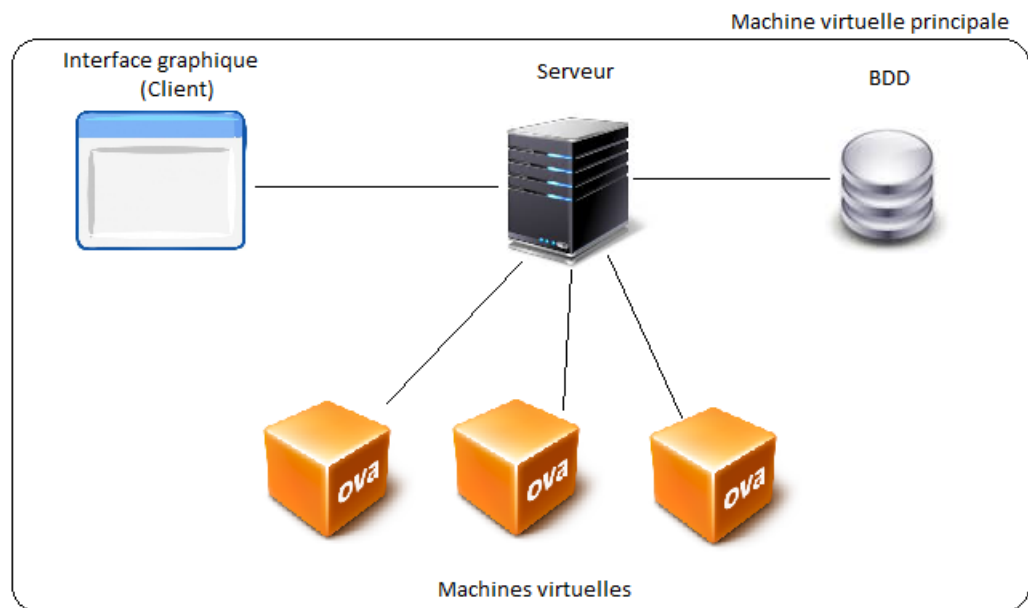
Nous avons convenu avec le client qu'un livrable possible serait sous la forme d'une machine virtuelle qui ferait fonctionner Paquito. Sur cette Machine Virtuelle il y aura :

Un service Web sous la forme d'un client/serveur

Le serveur sera relié à une base de données qui stockera les informations sur les utilisateurs mais également toutes les informations nécessaires pour pouvoir créer un fichier YAML qui réunit toutes les informations nécessaires pour que Paquito puisse créer les paquets.

3 ou 4 autre Machine Virtuelle en lien avec le service Web. Ces dernières permettrait de pouvoir créer les paquets, car pour que Paquito puisse créer les paquets d'une certaine distribution il faut qu'il soit lancé sous cette distribution

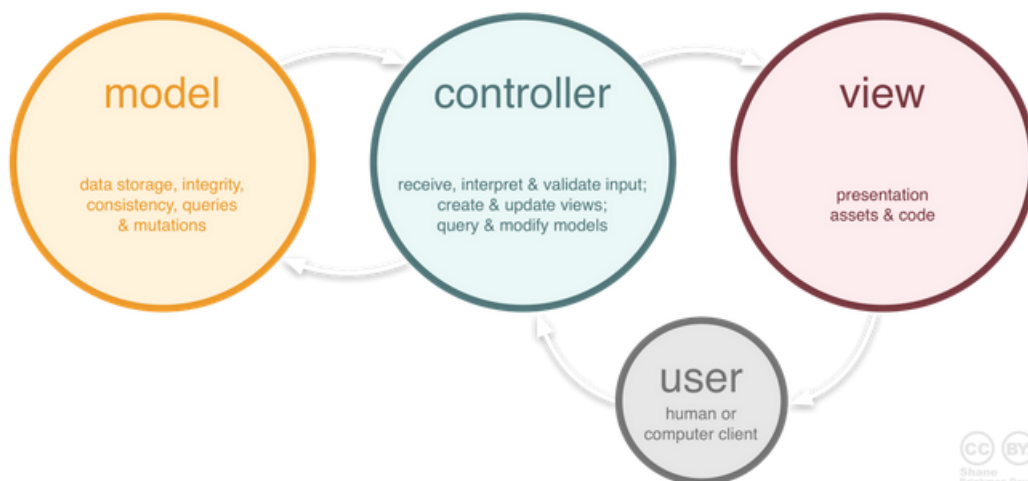
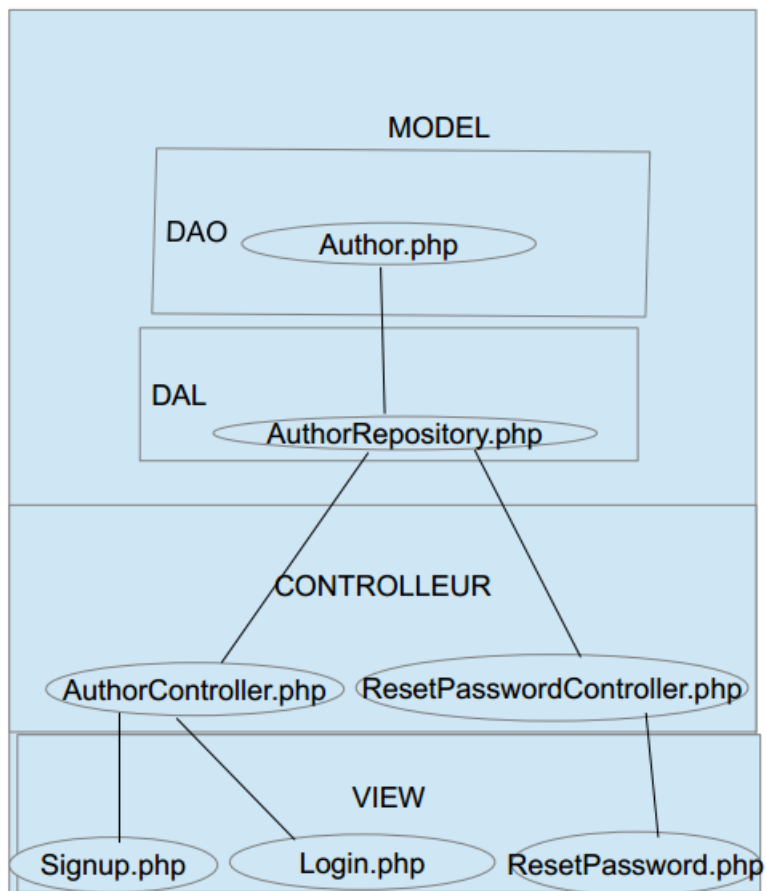
Voici un schéma permettant de visualiser l'architecture de notre projet.



3.2 Architecture côté serveur

L'utilisateur a la possibilité de s'inscrire et de se connecter sur l'application web. Il peut également ajouter un projet, en renseignant un certain nombre de champs. Nous avons décidé d'utiliser l'architecture Modèle-Vue-Contrôleur pour gérer la partie Serveur car pour nous il est important de bien organiser le travail dans le but de réaliser un projet comme le notre.

Ainsi, les schéma ci-dessous illustrent cette architecture appliquée à notre projet :



- Le Modèle contient le logique métier
- La vue regroupe tout ce qui a trait à la présentation (des données / comme

des interactions utilisateur)

- Le contrôleur répond à des interactions utilisateurs en provenance de la vue.

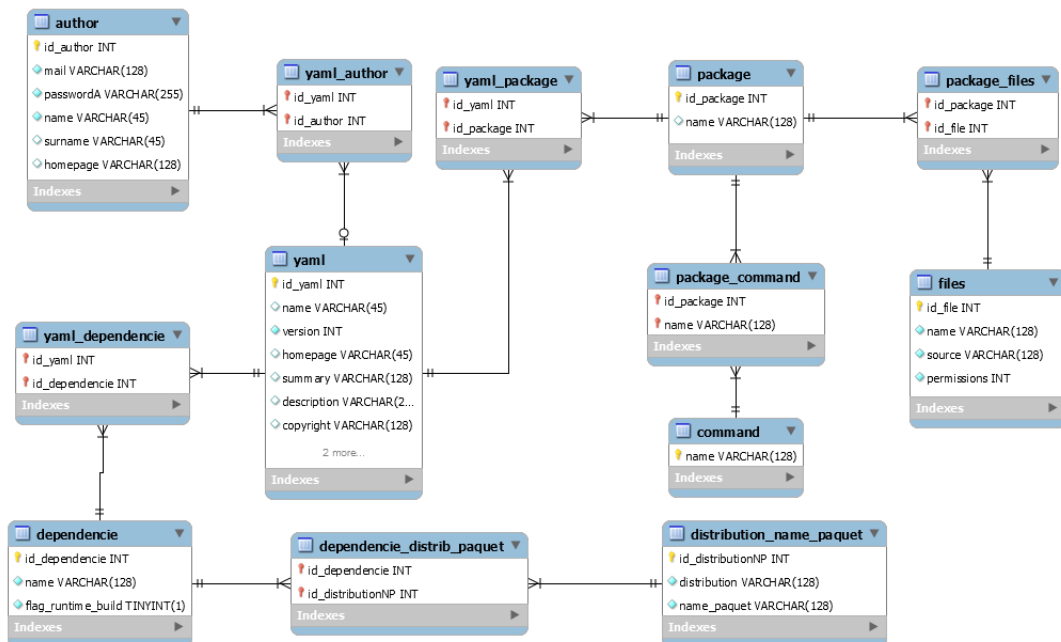
Nous avons utilisé le design pattern Data Access Object DAO pour découper la couche Modèle en deux autres couches qui sont :

- La DAL (Data Access Layer) : couche abstraction de données
- Le DAO (Data Access Object) : objet d'accès aux données.

Brève description des fichiers

- Author.php : présente les données dans une classe author. Nous y avons renseigné les différents attributs de la table qui gère les utilisateurs (author).
- AuthorRepository.php : Dispose d'un constructeur et fait appel à Author.php
- AuthorController.php : permet de gérer les actions de l'utilisateur, à savoir se connecter, s'inscrire, se déconnecter ; ainsi que certaines fonctions qui vérifie si l'utilisateur est connecté ou non. Nous avons ainsi dans ce fichier les fonctions : register(), login(), logout(), is_login().
- ResetPasswordControlleur.php : dispose de deux fonctions qui permettent à l'utilisateur de retrouver son mot de passe en cas d'oubli telles que resetPassword() et sendMail(). Cette dernière permet à l'utilisateur de recevoir un mail de réinitialisation de mot de passe.
- Les fonctions de la partie View font appel à ceux de la partie Contrôleur, et présenteront ainsi l'interface.

4 Base de données



Notre base de données sera liée au côté serveur de notre service Web et contiendra toutes les informations nécessaires pour pouvoir créer le fichier YAML d'un projet. Elle contient également les informations sur les utilisateurs afin de pouvoir gérer les connexions au service web.

La création d'un paquetage nécessite une grande quantité d'informations, celles-ci se trouvent dans le fichier `paquito.yaml` qui se divise en 2 parties :

- d'abord les "méta-informations" sur le paquetage, par exemple le nom du paquetage, sa version, une description de celui-ci, ses auteurs etc.
- ensuite la partie *Packages* qui regroupe les informations sur les dépendances et les opérations nécessaires à la construction du paquetage. Pour chaque dépendance, on liste pour chaque distribution les noms des paquetages donnant accès à la fonction voulue.

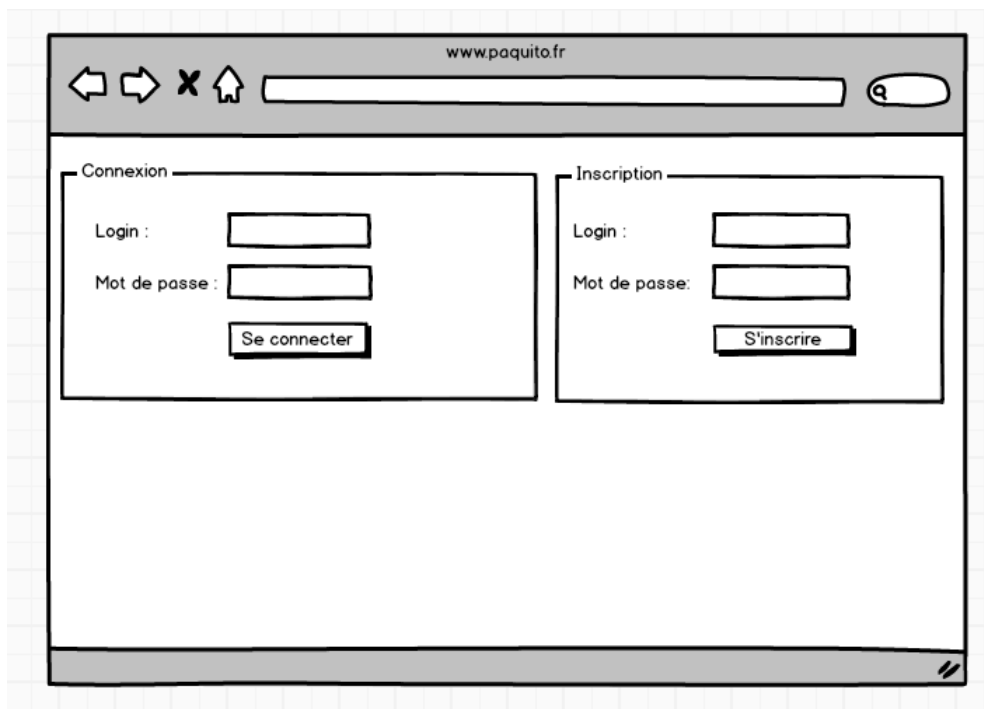
Nous sommes entrain de réfléchir sur la lecture des informations dans le fichier `yaml`.

Vous trouverez en annexe le script permettant de générer la base de données, ainsi que le fichier `YAML` utilisé par `Paquito`.

5 Maquette du site

Nous sommes en plein développement des pages Web, nous essayons grâce à Bootstrap de donner un design agréable à l'application. Nous avons préféré laisser les maquettes car le design peut être amené à changer souvent. Voici à nouveau un aperçu des maquettes de notre interface.

Nous avons dans un premier temps, la maquette de la page de connexion, cela reste classique avec un login et un mot de passe pour s'inscrire et se connecter.



La maquette illustre une interface web pour le site www.paquito.fr. Elle est présentée dans un navigateur avec des boutons de navigation (arrière, avant, fermer, accueil) et une barre d'adresse. Le contenu principal est divisé en deux colonnes : 'Connexion' à gauche et 'Inscription' à droite. La section 'Connexion' comprend des champs pour 'Login' et 'Mot de passe', suivis d'un bouton 'Se connecter'. La section 'Inscription' comprend des champs pour 'Login' et 'Mot de passe', suivis d'un bouton 'S'inscrire'. Une barre de statut grise est visible en bas de la page.

Nous avons ensuite l'interface d'ajout de projet à Paquito où il y a un onglet à remplir pour que par la suite Paquito puisse compiler et créer les paquets en fonction de la distribution si une personne voudrait l'installer sur sa machine.

Chargement Téléchargement

Nom du Projet :

Version :

Auteurs :

Mainteneur :

Description :

Dépendances :

Différentes dépendance du projet :

Exemple :

Runtime :

php-cli: Debian: php5-cli Archlinux: php-cgi

Build :

base-devel: Debian: build-essential Centos: "<none>"

Fichiers :

Fichiers associés au Projet

Exemple :

/usr/share/paquitophar: paquito

/usr/bin/paquito:

Source: paquito.sh

Permissions: 755

Importer fichier YAML existant

Valider informations

Puis nous avons la vue d'une page où l'on peut voir des détails sur le projet comme le nom du développeur, le nom du projet.

Projet

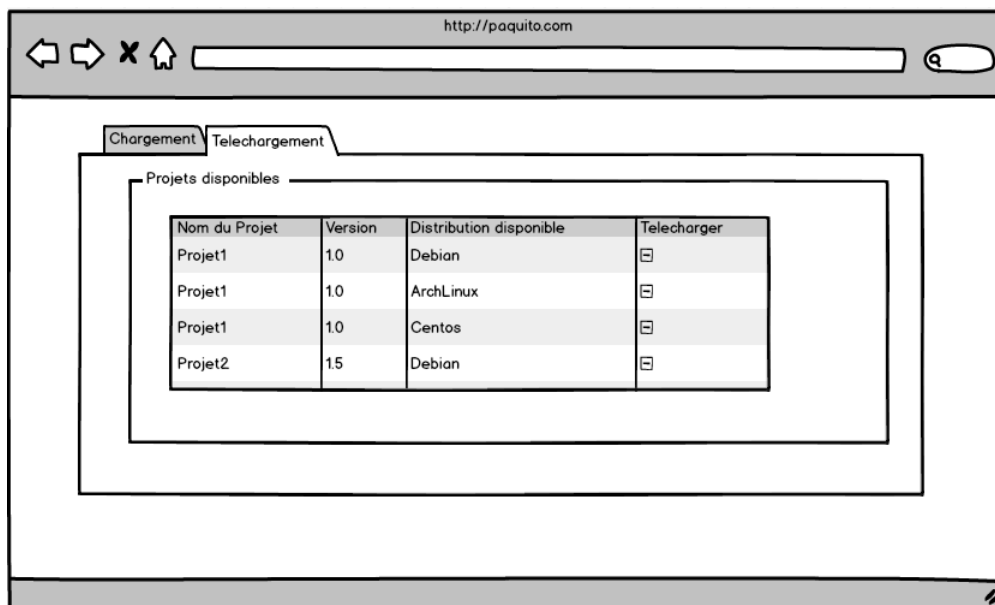
Nom :

Auteur :

Version:

Description :

Et enfin, la page pour télécharger les paquets générés.



6 Évolution du projet

Adaptation de paquito pour Ubuntu 14.04 Pour l'adaptation de Paquito à Ubuntu Trusty Tahr qui nous à été demandé à la place de Mac OS, le travail demandé devrait être un peu plus simple, en effet le système de paquets Ubuntu est le même que celui de Debian. Tout le travail que nous avons à faire c'est d'adapter le code de Paquito afin qu'il soit capable de prendre en compte le fait qu'il est sur un système sous Ubuntu. Pour le moment nous sommes dans une phase d'étude de l'existant afin de pouvoir l'adapter à Ubuntu, une fois cette phase finie nous réaliserons les modifications prévues et nous testerons la créations de paquetages sous Ubuntu.

Annexes

Script d'implémentation de la base de données

```
-- -----  
-- Schema mydb  
-- -----  
CREATE SCHEMA IF NOT EXISTS 'mydb' DEFAULT CHARACTER SET utf8 ;  
USE 'mydb' ;  
  
-- -----  
-- Table 'mydb'.'yaml'  
-- -----  
CREATE TABLE IF NOT EXISTS 'mydb'.'yaml' (  
    'id_yaml' INT NOT NULL,  
    'name' VARCHAR(45) NULL,  
    'version' INT NOT NULL,  
    'homepage' VARCHAR(45) NULL,  
    'summary' VARCHAR(128) NULL,  
    'description' VARCHAR(255) NULL,  
    'copyright' VARCHAR(128) NULL,  
    'maintainer' VARCHAR(128) NULL,  
    'depotGitSvn' VARCHAR(255) NULL,  
    PRIMARY KEY ('id_yaml'))  
ENGINE = InnoDB;  
  
-- -----  
-- Table 'mydb'.'author'  
-- -----  
CREATE TABLE IF NOT EXISTS 'mydb'.'author' (  
    'id_author' INT NOT NULL,  
    'mail' VARCHAR(128) NOT NULL,  
    'passwordA' VARCHAR(255) NOT NULL,  
    'name' VARCHAR(45) NOT NULL,  
    'surname' VARCHAR(45) NULL,  
    'homepage' VARCHAR(128) NULL,  
    PRIMARY KEY ('id_author'))  
ENGINE = InnoDB;
```

```

-- -----
-- Table 'mydb`.`yaml_author`
-- -----
CREATE TABLE IF NOT EXISTS 'mydb`.`yaml_author' (
  'id_yaml' INT NULL,
  'id_author' INT NULL,
  PRIMARY KEY ('id_yaml', 'id_author'),
  INDEX 'id_author_idx' ('id_author' ASC),
  CONSTRAINT 'fk_yaml_author_id_yaml'
    FOREIGN KEY ('id_yaml')
      REFERENCES 'mydb`.`yaml' ('id_yaml')
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT 'fk_yaml_author_id_author'
    FOREIGN KEY ('id_author')
      REFERENCES 'mydb`.`author' ('id_author')
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table 'mydb`.`package`
-- -----
CREATE TABLE IF NOT EXISTS 'mydb`.`package' (
  'id_package' INT NULL,
  'name' VARCHAR(128) NULL,
  PRIMARY KEY ('id_package'))
ENGINE = InnoDB;

-- -----
-- Table 'mydb`.`files`
-- -----
CREATE TABLE IF NOT EXISTS 'mydb`.`files' (
  'id_file' INT NULL,
  'name' VARCHAR(128) NOT NULL,
  'source' VARCHAR(128) NOT NULL,

```

```

    'permissions' INT NOT NULL,
    PRIMARY KEY ('id_file'))
ENGINE = InnoDB;

```

```

-- -----
-- Table 'mydb`.`dependencie`
-- -----

```

```

CREATE TABLE IF NOT EXISTS 'mydb`.`dependencie' (
    'id_dependencie' INT NULL,
    'name' VARCHAR(128) NOT NULL,
    'flag_runtime_build' TINYINT(1) NOT NULL,
    PRIMARY KEY ('id_dependencie'))
ENGINE = InnoDB;

```

```

-- -----
-- Table 'mydb`.`distribution_name_paquet`
-- -----

```

```

CREATE TABLE IF NOT EXISTS 'mydb`.`distribution_name_paquet' (
    'id_distributionNP' INT NULL,
    'distribution' VARCHAR(128) NOT NULL,
    'name_paquet' VARCHAR(128) NOT NULL,
    PRIMARY KEY ('id_distributionNP'))
ENGINE = InnoDB;

```

```

-- -----
-- Table 'mydb`.`command`
-- -----

```

```

CREATE TABLE IF NOT EXISTS 'mydb`.`command' (
    'name' VARCHAR(128) NULL,
    PRIMARY KEY ('name'))
ENGINE = InnoDB;

```

```

-- -----
-- Table 'mydb`.`yaml_package`
-- -----

```

```

CREATE TABLE IF NOT EXISTS 'mydb`.`yaml_package' (

```



```

'id_yaml' INT NULL,
'id_package' INT NULL,
PRIMARY KEY ('id_yaml', 'id_package'),
INDEX 'fk_yaml_package_id_package_idx' ('id_package' ASC),
CONSTRAINT 'fk_yaml_package_id_yaml'
    FOREIGN KEY ('id_yaml')
    REFERENCES 'mydb`.`yaml' ('id_yaml')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT 'fk_yaml_package_id_package'
    FOREIGN KEY ('id_package')
    REFERENCES 'mydb`.`package' ('id_package')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table 'mydb`.`package_files'
-- -----

```

```

CREATE TABLE IF NOT EXISTS 'mydb`.`package_files' (
    'id_package' INT NULL,
    'id_file' INT NULL,
    PRIMARY KEY ('id_package', 'id_file'),
    INDEX 'fk_package_files_id_file_idx' ('id_file' ASC),
    CONSTRAINT 'fk_package_files_id_package'
        FOREIGN KEY ('id_package')
        REFERENCES 'mydb`.`package' ('id_package')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT 'fk_package_files_id_file'
        FOREIGN KEY ('id_file')
        REFERENCES 'mydb`.`files' ('id_file')
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table 'mydb`.`dependencie_distrib_paquet'

```

```

-----
CREATE TABLE IF NOT EXISTS 'mydb'.'dependencie_distrib_paquet' (
  'id_dependencie' INT NULL,
  'id_distributionNP' INT NULL,
  PRIMARY KEY ('id_dependencie', 'id_distributionNP'),
  INDEX 'fk_depend_distrib_id_distribNP_idx' ('id_distributionNP' ASC),
  CONSTRAINT 'fk_depend_distrib_id_depend'
    FOREIGN KEY ('id_dependencie')
    REFERENCES 'mydb'.'dependencie' ('id_dependencie')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT 'fk_depend_distrib_id_distribNP'
    FOREIGN KEY ('id_distributionNP')
    REFERENCES 'mydb'.'distribution_name_paquet' ('id_distributionNP')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table 'mydb'.'package_command'
-----
CREATE TABLE IF NOT EXISTS 'mydb'.'package_command' (
  'id_package' INT NULL,
  'name' VARCHAR(128) NULL,
  PRIMARY KEY ('id_package', 'name'),
  INDEX 'fk_package_command_name_idx' ('name' ASC),
  CONSTRAINT 'fk_package_command_id_package'
    FOREIGN KEY ('id_package')
    REFERENCES 'mydb'.'package' ('id_package')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT 'fk_package_command_name'
    FOREIGN KEY ('name')
    REFERENCES 'mydb'.'command' ('name')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table 'mydb`.`yaml_dependencie`
-----
CREATE TABLE IF NOT EXISTS 'mydb`.`yaml_dependencie' (
  'id_yaml' INT NULL,
  'id_dependencie' INT NULL,
  PRIMARY KEY ('id_yaml', 'id_dependencie'),
  INDEX 'fk_yaml_depend_id_depend_idx' ('id_dependencie' ASC),
  CONSTRAINT 'fk_yaml_depend_id_depend'
    FOREIGN KEY ('id_dependencie')
      REFERENCES 'mydb`.`dependencie' ('id_dependencie')
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT 'fk_yaml_depend_id_yaml'
    FOREIGN KEY ('id_yaml')
      REFERENCES 'mydb`.`yaml' ('id_yaml')
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Exemple-type d'un fichier YAML : fichier YAML de Paquito

Name: Paquito

Version: 0.3

Homepage: <https://www.paquito.fr>

Summary: Programme de packaging Paquito

Description: Création automatique de paquet Debian (DEB), Archlinux (PKG) et CentOS (RPM) à partir de sources

Copyright: La licence

Maintainer: Corentin <coco@gmail.com>

Authors:

- sarah <sarah@gmail.com>
- corentin <corentin@gmail.com>

Packages:

paquito:

Type: binary

Files:

/usr/share/paquito.phar: paquito

/usr/bin/paquito:

Source: paquito.sh

Permissions: 755

Runtime:

Dependencies:

php-cli:

Debian: php5-cli

Archlinux: php-cgi

php-process:

Debian: "<none>"

Archlinux: "<none>"

rpm-build:

Debian: dh-make

Archlinux: "<none>"

```

        rpmdevtools:
            Debian: "<none>"
            Archlinux: "<none>"
        base-devel:
            Debian: build-essential
            Centos: "<none>"

Build:
    Dependencies:
        base-devel:
            Debian: build-essential
            Centos: "<none>"
        php-xml:
            Debian: "<none>"
            Archlinux: "<none>"
        php-process:
            Debian: "<none>"
            Archlinux: "<none>"
        rpmdevtools:
            Debian: "<none>"
            Archlinux: "<none>"
        rpm-build:
            Debian: dh-make
            Archlinux: "<none>"
    Commands:
        - bin/create-phar.sh

```