

|  |
| --- |
|  |

**Equipe**

Amirdine MDJASSIRI Charly CELEREAU

**Master 2 informatique FSIL**

**Architecture JEE**

**Année 2015-2016**

(Document de pages)

**Rapport de projet**

**Gestion d’un annuaire – Devoir**

L'objectif de ce projet est est de gérer à l'aide de la technologie JEE un annuaire de personnes**.**

Université d’Aix-Marseille  
**Faculté des Sciences**   
Site de Luminy 163 Avenue de Luminy 13009 MARSEILLE

Ce document est la propriété de l'Université de Provence.   
Toute reproduction même partielle ne peut se faire sans leur approbation préalable.

Section des Rédacteurs

|  |  |  |
| --- | --- | --- |
| **Nom** | **Prénom** | **Contribution** |
| MDJASSIRI | Amirdine | Implémentation de la couche DAO, Services et rédactions du rapport. |
| CELEREAU | Charly | Implémentation de la couche présentation (Servlet et JSP) et rédaction du cahier des charges. |

Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Prénom** | **Email** | **Fonction** |
| MDJASSIRI | Amirdine | mdjassiri.amirdine@yahoo.fr | Codeur, testeur et rédacteur |
| CELERAU | Charly | charly.celereau@gmail.com | Codeur, testeur et rédacteur |

Table des Matières

[1. Introduction 3](#_Toc210393436)

[2. Architecture 3](#_Toc210393440)

[3. Diagramme de Gantt 5](#_Toc210393440)

[4. Conception 5](#_Toc210393440)

1. Introduction

L’objectif de ce projet est de réaliser une application web en Java/JEE permettant :

* De consulter une liste de personne avec une vue détaillée pour chaque personne
* De s’authentifier et modifier sa description. Une personne connectée pourra voir des des informations qu’une personne non-connectée ne pourra pas voir.
* Et de récupérer son mot de passe.

1. Architecture

Pour notre projet, nous avons utilisé l’architecture 3-tiers. L’application est composé de trois couches. Une couche DAO (Data Acces Object) basée sur OpenJPA dont le but est de gérer l’accès aux données. Une couche métier (business) basée qui correspond à la partie fonctionnelle de l’application. Et une couche présentation basée sur la technologie Spring/JSP/Servlet qui est la partie visible de l’application qui fait office d’interface homme machine.

* 1. Modèle

**Package webdir.main.model:**

* Group
* Person
* User

Group représente l’objet groupe qui lui-même comporte des personnes. Person représente l’objet personne qui comporte un ensemble d’informations: nom, prénom, adresse électronique, site web, date de naissance et mot de passe. Chaque personne est placée dans un groupe. User représente l’utilisateur de l’application. Il permet aussi l’authentification d’une personne

* 1. DAO

**Package : webdir.main.business.dao :**

* GroupDao
* PersonDao

L’objet GroupDao gère essentiellement la table Group. Il permet de récupérer des informations, par exemple la liste des tables. PersonDao gère essentiellement la table Person. Il permet de récupérer des informations, par exemple la liste de toutes les personnes.

* 1. Services

**Package : webdir.main.business.services**

* GroupService
* PersonService
* EmailService
* UserService

GroupService gère des services tels que la récupération de la liste de tous les groupes contenus dans l’annuaire. PersonService gère des services tels que la récupération de liste de tous les personnes présente dans l’annuaire. EmailService permet d’envoyer des e-mails et UserService permet de se connecter et de récupérer son mot de passe via l’envoi d’un e-mail.

* 1. Présentation

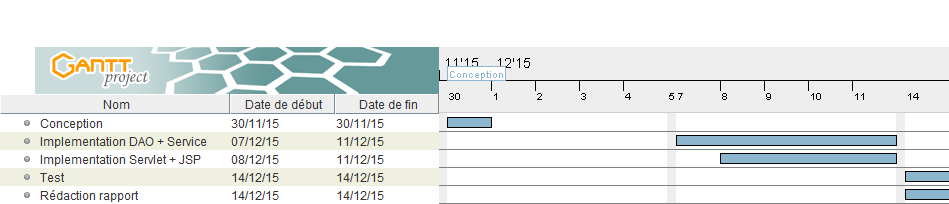
**Package : webdir.main.web (Servlet):**

* IHomeController

**Chemin: WebContent/WEB-INF/jsp (JSTL) :**

* home.jsp
* errors.jsp
* template.jsp
* includes/footer.jsp
* includes/header.jsp
* includes/footer.jsp
* includes/header.jsp
* includes/side\_pannel.jsp
* includes/stylesheets.jsp

1. Diagramme de Gantt



1. Conception
   1. Schéma Merise

|  |
| --- |
| **Person**  **(1 , 1) (0 , n)** |
| * id (int) * firstname (varchar) * lastname (varchar) * birthdate (date) * email (varchar) * website (varchar) * password (varchar) * group\_id (varchar) |

|  |
| --- |
| **Group\_** |
| * groupID (int) * name (varchar) |

La base de données comportera deux tables, une table PERSON permettant de décrire une personne et une table GROUP\_ permettant de décrire un groupe. Une personne appartient à un et un seule groupe et un groupe peut appartenir à 0 ou plusieurs personnes.

* 1. Diagramme de classes

|  |
| --- |
| @Entity(name = "Group\_")  **Group** |
| - groupID  - name |
| \* Getter et setters |

**Modèle :**

|  |
| --- |
| @Entity(name = "Person")  **Person** |
| - id : long  - firstname : String  - lastname : String  - birthdate : Date  - email : String  - website : String  - password : String  - group\_id : Group |
| \* Getter et setters |

|  |
| --- |
| **User** |
| +id  + password |
| \* getter et setters |

|  |
| --- |
| **<< Interface>>**  **IGroupDao** |
|  |
| + init(): void  + getAllGroups():List<Group>  + getContentGroup(long id): Collections<Person>  + getGroup(long id): Group  + groupIDEexist(long id): boolean  + close(): void |

**DAO :**

|  |
| --- |
| **<< Interface>>**  **IPersonDao** |
|  |
| + init(): void  + getPerson: Person  + getAllPersons(): List<Person>  + personIDExists(long id): boolean  + updatePerson(Person p): void  + close(): void |

|  |
| --- |
| **GroupDao** |
| - em : EntityManager  - factory : EntityManagerFactory |
| + init(): void  + getAllGroups():List<Group>  + getContentGroup(long id): Collections<Person>  + getGroup(long id): Group  + groupIDEexist(long id): boolean  + close(): void |

|  |
| --- |
| **PersonDao** |
| - em : EntityManager  - factory : EntityManagerFactory |
| + init(): void  + getPerson: Person  + getAllPersons(): List<Person>  + personIDExists(long id): boolean  + updatePerson(Person p): void  + close(): void |

**Service:**

|  |
| --- |
| **<< Interface>>**  **IPersonService** |
|  |
| + init(): void  + getPerson(): Person  + getAllPersons(): List<Person>  + personIDExists(long id): boolean  + updatePerson(Person p): void  + close(): void |

|  |
| --- |
| **<< Interface>>**  **IGroupService** |
|  |
| +init():void  +getAllGroups():Collections<Group>  +getContentGroup(long id):Collections<Person>  +getGroup(long id): Group  +groupIDEexist(long id):boolean  + close(): void |

|  |
| --- |
| **GroupService** |
| - groupDao: IGroupDao |
| +init():void  +getAllGroups():Collections<Group>  +getContentGroup(long id):Collections<Person>  +getGroup(long id): Group  +groupIDEexist(long id):boolean  +close():void |

|  |
| --- |
| **PersonService** |
| - personDao: IPersonDao |
| + init(): void  + getPerson: Person  + getAllPersons(): List<Person>  + personIDExists(long id): boolean  + updatePerson(Person p): void  + close(): void |

|  |
| --- |
| **<< Interface>>**  **IEmailService** |
|  |
| + sendEmail(String recipient, String subject, String content): void |

|  |
| --- |
| **EmailService** |
| - username: String // Adresse e-mail de l'expéditeur  - password: String // Mot de passe l'adresse e-mail de l'expéditeur  - prop: Properties // Configuration serveur smtp Gmail  - session: Session |
| + sendEmail(String recipient, String subject, String content): void  + setUsername(String username) : void  + setPassword(String password) : void  + setProp(Properties prop) : void |

|  |
| --- |
| **<< Interface>>**  **IUserService** |
|  |
| +connect(long login, String password):boolean  +sendUserPassword(long id):void //id de l'utilisateur |

|  |
| --- |
| **IUserService** |
| - personDao: IPersonDao  - emailService: IEmailService |
| +connect(long login, String password):boolean  +sendUserPassword(long id):void //id de l'utilisateur |

1. Bilan

Ce projet nous a permis de mettre en pratique ce qu’on a appris en cours. Maintenant nous sommes capables de faire une application web complète en Java/JEE. Au cours de ce projet nous avons rencontré de nombreux problèmes comme les conflits de dépendances, eu des difficultés dans l’installation et la configuration de notre environnement de développement, savoir où placer les fichiers CSS.