LogiSoft

I18n Project Manager
Document architecture logiciel
Version 1.1

Historique des révisions

Date	Version	Description	Auteur
<23/05/2016>	<1.0>	Création du document	MAARAF Hafida
<04/06/2016>	<1.1>	Modification du document	MABROUK Med Yahya

Table des matières

1. Introduction 4

2. Objectif du logiciel 4
   - 2.1 Contexte 4
   - 2.2 Besoins fonctionnels 4
   - 2.3 Besoins non fonctionnels 4

3. Structure 4
   - 3.1 Vue des couches 4
   - 3.2 Sous-systèmes et paquetages 4
   - 3.3 Interfaces 5

4. Comportement 5
   - 4.1 Réalisation des cas d'utilisation 5
   - 4.2 Mécanismes 5

5. Autres vues 5
   - 5.1 Vue processus (optionnel) 5
   - 5.2 Vue implémentation (optionnel) 5
   - 5.3 Vue déploiement (optionnel) 6
   - 5.4 Vue données (optionnel) 6

6. Concepts du domaine 6

7. Qualités de l'architecture 6

8. Points ouverts 6

1. Introduction

Ce document permet d'avoir une vision claire et globale sur l'architecture du système cela grâce aux principaux modules qui détaillent les différents aspects du système .Ces derniers mettent en évidence nos choix concernant l'architecture qui vont influencer la conception.

Le document architecture logicielle est destiné à l'architecte et aux concepteurs du projet puisque toute modification apporté à ce document impliquera une révision du modèle de cas d'utilisation.

Références

- Document Vision 1.2
- Modèle des cas d'utilisation 1.0
- Glossaire 1.0

2. Objectif du logiciel

Ce logiciel propose la gestion de projets adapté au contexte multinational des organismes tels que les groupes industriels ou les ONG internationaux. Il permet la gestion et le suivi de l'avancement des activités des projets internationaux tout en prenant en considération les préférences culturelles de ces utilisateurs.

2.1 Contexte

Ce logiciel vient pour réduire les nuisibilités dus aux différents référentiels utilisés par les membres de la même équipe. Ainsi, il propose d'affecter les ressources, arranger des meetings, effectuer des plannings tout en respectant les préférences culturelles des acteurs d'un projet.

2.2 Besoins fonctionnels

Dans cette partie, on présente les cas d'utilisation les plus importants pour l'architecture du logiciel. Les cas d'utilisation que nous avons retenus sont les suivants :

- **CU1 :** Mettre à jour et consulter planning.
- **CU2 :** Etablir les plannings et Arranger les meetings.
- **CU3 :** Configurer projet.

2.3 Besoins non fonctionnels

Pour assurer le bon fonctionnent de l'application plusieurs contraintes sont à prendre en considération lors de l'établissement de l'architecture du système à savoir :

- L'application doit être fonctionnelle, performante et le temps de réponse aux requêtes clients plutôt court.

- Les activités liées aux projets, ainsi que les informations personnelles des utilisateurs doivent être enregistrées d'où la nécessite de gérer, stocker les données persistances

- L'application doit être compatible avec les différents systèmes d'exploitation et navigateurs.

-L'application doit permettre à plusieurs utilisateurs de se connecter à la fois sans que cela engendre une altération de ces qualités.

-L'environnement de développement exigé est Mega.

3. Structure

On a établi un style d'architecture qui répond au mieux aux exigences de nos clients à savoir :

- LesFramework qu'on opté pour sont :

  - Hibernate qui va nous faciliter le développement de la couche persistance.

  -JSF et primefaces .

- Les design Patterns choisies sont :

  -le pattern MVC pour bien organiser le code source et séparer les différentes parties de l'application.

  -DAO

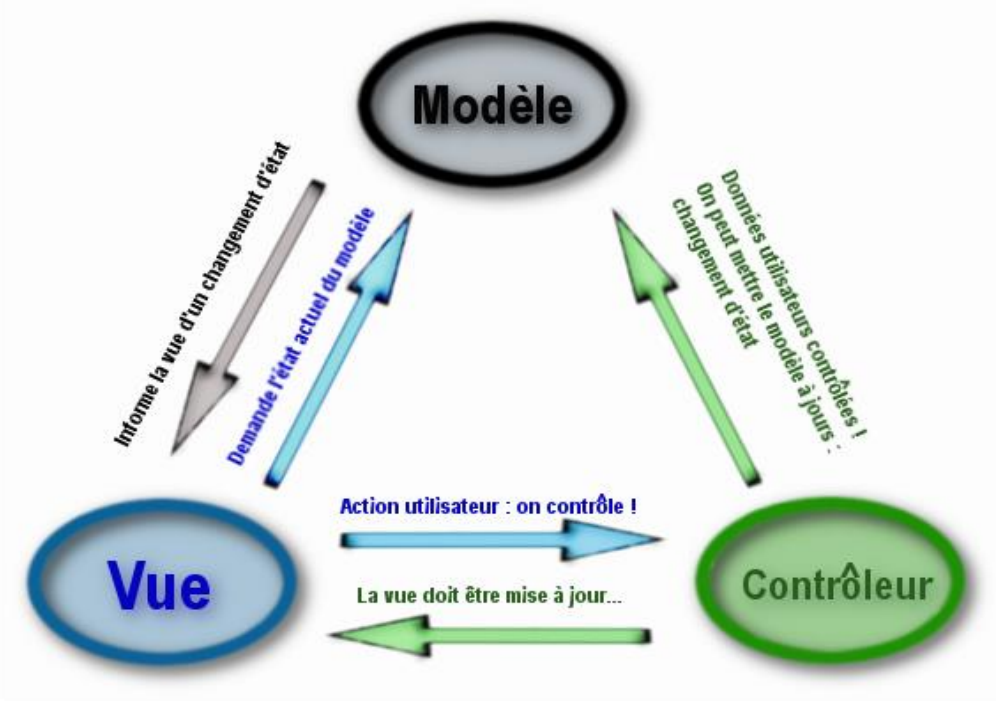
  -State

  -Observer

Dans la suite de cette section, on va détailler la structure statique de la vue logique de l'architecture tout en montrant les composants, leurs interconnexions et les interfaces offertes par ces composants.

3.1 Vue des couches

Le diagramme suivant représente les couches du logiciel. L'explication de chaque couche sera donnée par la suite :


Modèle
Informe la vue d'un changement d'état
Demande l'état actuel du modèle
Données utilisateurs contrôlées !
On peut mettre le modèle à jours :
changement d'état
Action utilisateur : on contrôle !
La vue doit être mise à jour...
Vue
Contrôleur

La couche Vue

La couche vue permet de fournir les interfaces à l'utilisateur .Elle interagit avec la couche contrôleur et lui transmet les requêtes saisies par l'utilisateur.

La couche vue ne gère que les aspects graphiques de l'application et c'est la seule couche qui va interagir directement avec l'utilisateur .

La couche Contrôleur

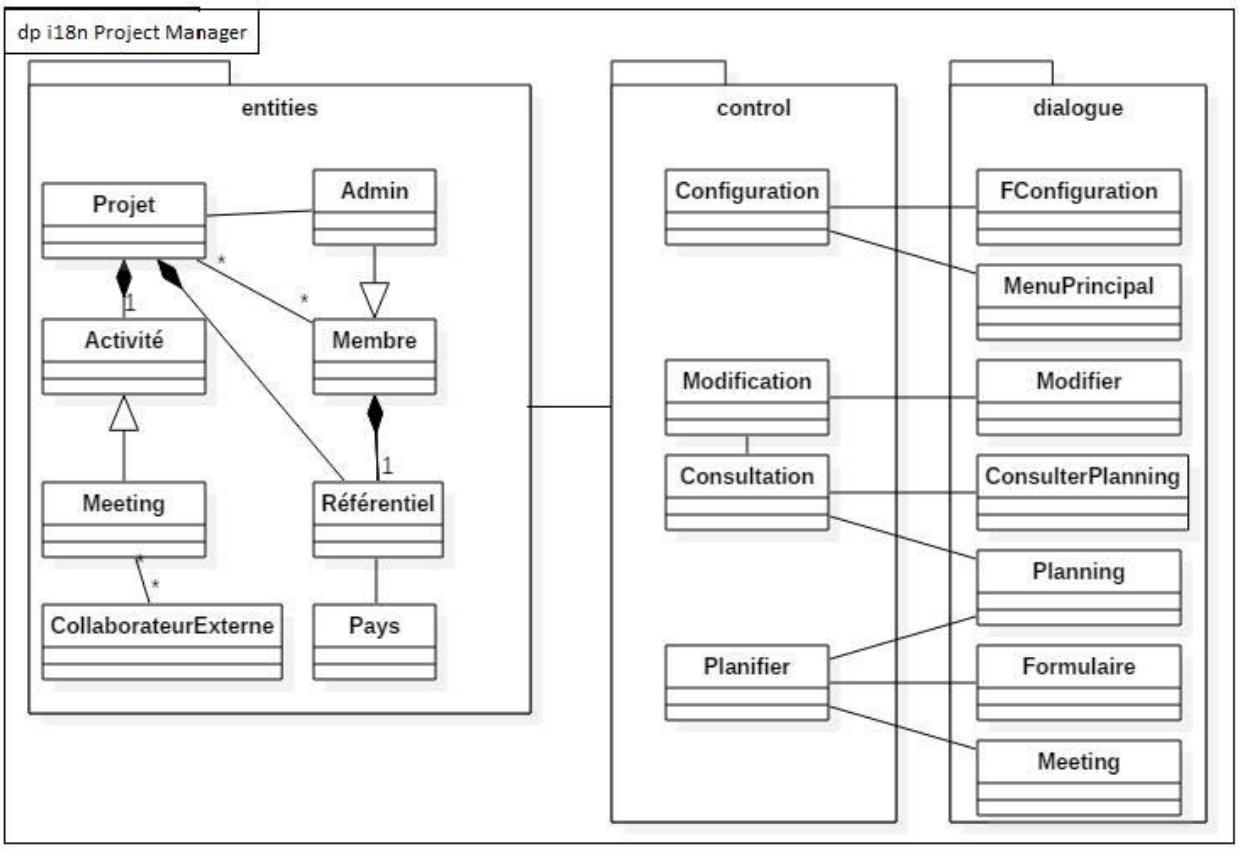
La couche Contrôleur gère les requêtes des utilisateurs. Elle est responsable de retourner une réponse en agissant sur deux niveaux de l'architecture :

- Avec la couche Vue, afin de gérer les Interfaces Homme-Machine
- Avec la couche Modèle, afin d'effectuer les traitements sur les objets métier de l'application, c'est-à-dire au niveau interne.

La couche Modèle

La couche Modèle représente la partie de l'application qui exécute la logique métier. Cela signifie qu'elle est responsable de récupérer les données, de les convertir selon des concepts chargés de sens pour l'application. Chaque composant de cette couche sera responsable d'un élément interne de l'application et possèdera ses propres attributs et méthodes.

3.2 Sous-systèmes et paquetages



3.3 Interfaces

[optionnel. Description de chaque interface avec ses opérations.]

4. Comportement

[Cette section montre comment la collaboration des composants répond aux besoins fonctionnels et non fonctionnels.]

4.1 Réalisation des cas d'utilisation

[pour chaque cas d'utilisation identifié comme important pour l'architecture, présenter sa réalisation. On prendra un ou plusieurs scénarios du cas d'utilisation et

on le décrira par un diagramme d'interaction. Les objets apparaissant dans les diagrammes doivent être clairement identifiés dans la partie structure. Ajouter si nécessaire des explications pour qu'on comprenne bien comment les éléments structurels contribuent au scénario.]

4.2 Mécanismes

[Présenter les mécanismes et patterns du logiciel.

Pour chacun décrire son mode d'emploi pour qu'un concepteur puisse l'utiliser facilement. Ajouter éventuellement des diagrammes d'interaction. Tracer avec les besoins non fonctionnels.]

5. Autres vues

5.1 Vue processus (optionnel)

[Cette section présente les classes actives (processus et "threads") du logiciel, et montre le "mapping" avec les composants de la vue logique. Décrire les principaux mécanismes de communication : envoi de messages, interruptions, signaux, files d'attente …]

5.2 Vue implémentation (optionnel)

[Cette section décrit l'organisation des composants physiques (code) et le mapping avec les composants logiques.]

5.3 Vue déploiement (optionnel)

[Cette section décrit la partie matérielle de l'infrastructure et montre le mapping des composants physiques ou logiques ou des processus sur les nœuds. Chaque nœud et les interconnexions entre eux sont décrits.]

5.4 Vue données (optionnel)

[Cette section présente les données persistantes.]

6. Concepts du domaine

[Une description des concepts spécifiques du domaine et de leurs relations. Préciser comment les choix d'architecture préservent l'indépendance de ces concepts vis à vis de la technologie.]

7. Qualités de l'architecture

[Avantages et inconvénients de l'architecture. Limitations en vue d'une extension.]

8. Points ouverts