Projet : Gestion de tontine

Membres : AGBETONYO Joël – KAYI Yao Maxime

Encadreur : BATANA Ferdinand

**Architecture logicielle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Description / Changements apportés** | **Intervenant** | **Date** |
| 1.0 | Initialisation | Yao Maxime KAYI | 23/11/20 |

# 1. Introduction

Le présent document donne présente en générale l’architecture de la plateforme de Gestion des groupes de tontine tout en fournissant des explications détaillées.

Il définit l’objectif de l’architecture proposée, les cas d’utilisation supportés par le système, les styles et les composants qui ont été sélectionné. Le document fournit une justification de l’architecture et les décisions de conception obtenues à partir de l’idée conceptuelle jusqu’à son implémentation.

## 1.1 Objectifs

Le document d’architecture logicielle (DAL ou SAD en anglais) fournit un aperçu compréhensif de l’architecture du système de Gestion de Tontine. Il présente un certain nombre des différentes vues architecturales menant vers le déploiement du système.

Dans le but de décrire le système aussi précisément que possible, la structure du document est basé sur le « 4+1 » vues du modelé architectural de Philipe Kruchten.

Vue Conception

Vue Implémentation

Vue Logique

Vue Déploiement

Vue Cas d’utilisation

## 1.2 Portée

La portée de ce document est d’expliquer l’architecture du système de Gestion de Tontine.

Le document décrit plusieurs aspects de conception du système de Gestion de Tontine, qui sont importants.

## 1.3 Définitions, Acronymes et Abréviations

* SAD : Software Architecture Document
* DAL : Document Architecture Logicielle
* Apache : web server
* UML : Unified Modeling Language (Langage de Modélisation Unifié)
* SGT : Système de Gestion de Tontine

## 1.4 Références

[Kruchten]: The “4+1” view model of software architecture, Philippe Kruchten, November 1995, <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf>

Le SAD sera présenté dans l(ordre suivant :

1. Décrire chaque cas d’utilisation
2. Décrire l’objectif et les contraintes du système
3. Décrire les cas d’utilisations les plus importants
4. Décrire la vue logique du système avec les interfaces et définir les méthodes
5. Insister sur les éléments les plus importants
6. Décrire le processus déploiement

# 2. Représentation architecturale

Le document donne en détail l’architecture du système en utilisant les vues définies dans le « 4 + 1 » model de vues de Kruchten. Les vue utilisées pour documenter le SGT sont :

**Vue Cas d’utilisation** :

**Cibles** : toute partie prenante du système même les utilisateurs finaux

**Objectifs** : décrire la série des scénarios et/ou des cas d’utilisation du système. Cette vue s’occupe de présenter les besoins de l’utilisateur

**Données connexes** : les modèles de cas d’utilisation, la document des cas d’utilisation

**Vue Logique** :

**Cibles** : toute partie prenante du système même les utilisateurs finaux

**Objectifs** : décrire la série des scénarios et/ou des cas d’utilisation du système. Cette vue s’occupe de présenter les besoins de l’utilisateur

**Données connexes** : les modèles de cas d’utilisation, la document des cas d’utilisation

**Vue de Donnée** :

**Cibles** : toute partie prenante du système même les utilisateurs finaux

**Objectifs** : décrire la série des scénarios et/ou des cas d’utilisation du système. Cette vue s’occupe de présenter les besoins de l’utilisateur

**Données connexes** : les modèles de cas d’utilisation, la document des cas d’utilisation

**Vue de Déploiement** :

**Cibles** : toute partie prenante du système même les utilisateurs finaux

**Objectifs** : décrire la série des scénarios et/ou des cas d’utilisation du système. Cette vue s’occupe de présenter les besoins de l’utilisateur

**Données connexes** : les modèles de cas d’utilisation, la document des cas d’utilisation

# 3. Objectifs et Contraintes Architecturales

Il existe quelques exigences clés des contraintes su le système qui ont un impact important sur l’architecture. Ceux sont :

1. Le système sera développé en utilisant les technologies JavaScript. Principalement le Type Script avec Angular pour le frontEnd et Express de Node js ave les dépendances pour les API et côté backEnd. On utilisera un SGBDR comme MySQL pour assurer la persistance de nos données et le système complet sera déployé sur Heroku. Il y a d’autres aspects qui feront surface lors du développement et qui seront prises en considération.
2. Le système doit échanger des données avec des APIs Tierces. Ceci pour pouvoir greffer de nouveaux services à notre application dans les versions ultérieures.

# 4. La vue cas d’utilisation

L’objectif de vue est de présenter les détails qui émanent de l’utilisation du système et les interactions entre ses composants. La section 4.1 liste tous les acteurs avec une description de chacun dans toutes les parties du système. La section qui suit, les cas d’utilisation les plus importants sont illustrés et modélisés avec le langage UML, diagramme de cas d’utilisation et le diagramme de séquence pour plus de clarification.

## 4.1 Acteurs