**INM5151 – Projet d'analyse et de modélisation**

**Concept of operations (ConOps)**

**Application Web d’organisation intuitive de l’agenda (WIZENDA)**

**Travail remis par**

**Bonnaire Benjamin**

**Travail remis à**

**Hafedh Mili dans le cadre du cours INM5151groupe 50**

**le 11 Mars 2019**

Table des matières

1. Introduction

Le système en développement est un agenda électronique intelligent qui rajoute des fonctionnalités intuitives par rapport aux autres systèmes similaires que l’on peut trouver sur le marché.

1.1 Objectifs

Ce document a pour but de présenter et de décrire les caractéristiques du système développées au cours du sprint courant. Dans ce documents se retrouveront en détails chacune des éléments qui ont été touchés au cours de ce sprint

1.2 Vue d’ensemble du produit

Ce document a pour but de décrire l’ensemble des attentes et des ressources du système afin de permettre à tous les acteurs prenant parti à son développement puisse avoir toutes les informations nécessaires. Grâce à ce document il sera aussi possible de mettre en lumière les nouveautés apportées par ce système par rapport à ce qu’il se fait déjà.

**Usagers** : Les usagers de ce système seront toutes les personnes souhaitant se donner des objectifs et ayant du mal à s’organiser ou toute personne qui souhaite bénéficier d’un agenda proposant des fonctionnalités intuitives et nouvelles.

**Développeurs** : Les développeurs sont chargés du développement du système et on besoin de toutes les informations relatives au produit, ce document leur sera donc indispensable.

**Administrateur** : L’administrateur est l’entité qui aura pour charge de veiller au bon fonctionnement du serveur.

1.3 Définitions, acronymes et abréviations

1.4 Documents de références

IEEE Guide for Information Technology-System Definition-Concept of Operations (ConOps) - IEEE, [EN LIGNE], le 2 octobre 2010, http://intra.info.uqam.ca/Members/mili\_h/ Enseignement / inm 5151- aut 10/ PresentationConcept. doc

Guide d’intégration Google Calendar:

<https://support.google.com/calendar/answer/41207?hl=fr>

Image architecture client Serveur :

https://www.google.com/search?q=client+serveur&source=ln ms&tbm=isch&sa=X&ved=0ahUKEwj12cOltLfgAhUlqlkKHW7vCzwQ\_AUIDigB#imgrc=e8HX7eTnUCUZ7M:

1.5 Aperçu du document

En première partie, dans ce document sera présentée un description générale du logiciel, comprenant la planification initiale des sprint. Ensuite sera développé le contenu du sprint actuel, avec ses cas d’utilisations détaillés, ainsi que les classes présentes.

Enfin sera énoncée la planification des prochains Sprint.

2. Description générale du logiciel

2.1 Vue d'ensemble des fonctions du produit

L’application WIZENDA, à partir des données entrées par l’utilisateur génère plusieurs propositions d’organisations de leur semaines. L’utilisateur entre premièrement ses obligations, s’il s’agit d’un étudiant, il rentrera ses heures de cours et/ou de travail en tant qu’obligation, il rentrera ensuite ses objectifs (heures souhaitées de sommeil, heures de sport, heures d’études souhaitées par semaine), et l’application se chargera de générer plusieurs propositions d’agencement de ses objectifs dans la semaine en fonctions du temps qu’il lui reste.

Si des évènements ponctuels (par exemple un rendez-vous) sont entrés pour une semaine particulière, l’application générera alors d’autres propositions.

L’utilisateur se crée un profil sur l’application, auquel il aura accès à partir de n’importe quel fureteur et y retrouvera l’ensemble de ses informations.

En accédant au site web, l’utilisateur arrive sur une interface lui présentant l’application et ses fonctionnalités ainsi qu’une invitation à se créer un compte ou à se connecter s’il en possède déjà un. Dans le cas où il ne posséderait pas de compte, il suivra une étape de création de compte ou il entrera ses informations personnelles dans un petit formulaire. La possibilité lui sera donnée de, s’il possède un compte Google, d’utiliser le processus de connexion rapide en entrant ses identifiants et mots de passe Google. Sinon le processus d’inscription sera plus long et un mail de confirmation lui sera envoyé.

Après s’être connecté, l’utilisateur accède à la page principale lui présentant son agenda. S’il s’agit de sa première connexion, une visite guidée lui sera proposée, qu’il aura le choix de suivre ou de décliner. L’utilisateur à partir de cette page a accès à toutes les fonctionnalités de l’application. Les différents concepts auxquels l’utilisateur pourra avoir accès sont détaillé ici :

* **Événement :** Plage horaire définie par l’utilisateur dans laquelle il ajoute un titre, une description et/ou une catégorie. Par défaut tout événement ajouté par un utilisateur manuellement est considéré comme une obligation
* **Obligation :** Évènement dont la date et l’heure ne seront pas changés par l’algorithme lors de son exécution.
* **Objectif :** Événement récurrent souhaité par l’utilisateur pour lequel il n’a donné que la durée souhaitée par semaine ou par jour.
* **Priorité :** Attribut donné à un objectif qui lui donne un ordre d’importance par rapport à un autre
* **Rappel :** Fonctionnalité qui permet à l’utilisateur d’être notifié à l’approche d’un évènement
* **TimeWizard :** Algorithme qui générera un agenda en fonction des obligations, des objectifs et de leur priorité

Afin d’assurer l’intégrité des informations des utilisateurs, après chaque insertion, une sauvegarde est effectuée au niveau du serveur.

2.2 Contenu des sprints

Lors de la planification initiale, le plan prévu pour les sprint était le suivant :

**Sprint 1:**

* Une application d’agenda fonctionnel à laquelle on peut se connecter, ajouter des évènements et un affichage en jour et semaines.
* Base de données fonctionnelles.

**Sprint 2 :**

* Implémentations des distinction (Obligations, objectifs )
* Implémentation de catégories (Travail, étude, sommeil, sport, loisir, Rendez-vous etc.)

**Sprint 3 :**

* Implémentation de l’algorithme (TimeWizard) qui est chargé d’agencer le nouvel emploi du temps en fonction des objectifs et des obligations
* Implémentation des priorités dans les objectifs
* Amélioration de la convivialité de l’interface utilisateur

**3. Contenu du sprint actuel (6 à 10 pages. max)**

3.1 Cas d’utilisation implantés dans le sprint

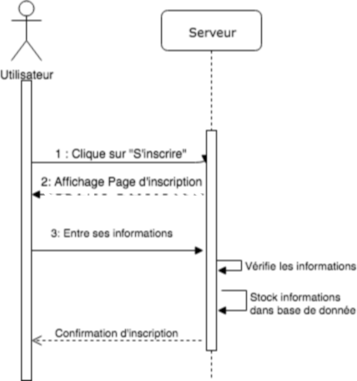
Pour le sprint actuel, il a été question d’implémenter la base de données, la page d’accueil, la page d’inscription, la page personnelle de l’utilisateur ainsi que le fonctions pertinentes pour leur fonctionnement.

|  |  |
| --- | --- |
| **Nom** | Créer un profil |
| **Niveau** | But utilisateur |
| **Acteurs** | utilisateur |
| **Préconditions** |  |
| **Post-Conditions** | Profil enregistré dans la base de donnée du serveur |
| **Scénario Principal** | 1. L’usager accède à la page d’accueil du site 2. La page d’accueil s’affiche 3. L’usager clique sur « Pas encore inscrit? » 4. Le système renvois l page de création de compte 5. L’usager entre son nom, son email et son mot de passe et confirme son inscription 6. Le système vérifie les informations 7. Le système stocke les informations dans la base de donnée 8. Le système donne une confirmation |
| **Scénario Alternatif** | 6.b Les informations entrées sont invalides  6.b.1Le système retourne un message d’erreur  6.b.2 Go to 5 |
| **Autres Commentaires/exigences** | Si les informations sont déjà inscrites dans la base de donnée, le mot de passe est trop court ou le nom contient des caractères spéciaux, l’inscription ne se fait pas |

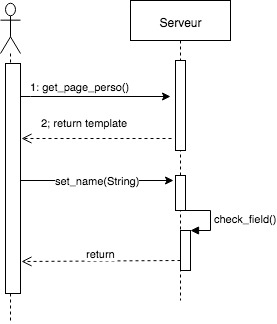
Ces implémentations répondent à un certains nombres de cas d’utilisations qui seront énumérées ci dessous.

1. Créer un profil
2. Se connecter à son profil
3. Modifier son nom d’utilisateur
4. Modifier son mot de passe
5. Modifier son adresse email

**Cas d’utilisation 1 : Créer un profil**

**Cas d’utilisatio****n 2 : Modifier son nom d’utilisateur**

|  |  |
| --- | --- |
| **Nom** | Modifier nom d’utilisateur |
| **Niveau** | But utilisateur |
| **Acteurs** | utilisateur |
| **Préconditions** | Avoir un compte et être connecté |
| **Post-Conditions** | Nouveau nom d’utilisateur sauvegardé dans la base de donnée |
| **Scénario Principal** | 1. L’usager clique sur « mon compte » 2. Le serveur lui retourne la page avec ses informations personnelles 3. L’usager clique sur modifier mes informations 4. Le système rends modifiables les champs 5. L’usager entre son nouveau nom et clique sur le bouton « Modifier » 6. Le système vérifie si le nom est valide et est différent du précédent 7. Le système stocke les informations dans la base de donnée 8. Le système donne une confirmation |
| **Scénario Alternatif** | 6.b Les informations entrées sont invalides  6.b.1Le système retourne un message d’erreur  6.b.2 Go to 5  6.c L’utilisateur décide d’annuler la modification et clique sur le bouton « Annuler »  6.c.1 La modification est annulée |
| **Autres Commentaires/exigences** | Si le nom contient des caractères spéciaux, la modification est invalide |



|  |  |
| --- | --- |
| **Signature** | <boolean> <verification\_mail\_existant>(<utilisateur.email>) |
| **Description** | Vérifie si le système possède déjà cet email dans sa base de donnée |
| **Références** | Créer un compte utilisateur |
| **Entrées** | 1. utilisateur.email : String - Attribut de la classe utilisateur dans lequel se trouve l’adresse mail |
| **Sorties** | Boolean indiquant si l’email se trouve déjà dans la base du système.  True : email déjà connu  False : email inexistant |
| **Exceptions** | Aucune |
| **Pré-conditions** | Aucune |
| **Post-conditions** | Aucune |
| **Notes** |  |

Afin d’effectuer ce cas d’utilisation, les méthodes suivantes sont appelées :

* create\_user(utilisateur) : Insère les informations d’un nouvel utilisateurs dans la base de donnée
* verification\_mail\_existant(email) : Vérifie si le mail entré existe déjà dans la base de données

|  |  |
| --- | --- |
| **Signature** | <boolean> <create\_user>(<utilisateur>) |
| **Description** | Insère les valeurs du nouvel utilisateur dans la base de donnée et crypte son mot de passe |
| **Références** | Créer un compte utilisateur |
| **Entrées** | 1. user : Classe de l’utilisateur contenant les attributs nom, prénom et mots de passes entrés lors du remplissage du formulaire de création de compte |
| **Sorties** | Boolean indiquant si l’entrée a bien été effectuée |
| **Exceptions** | une chaine de caractère trop longue dans les attributs pourrait ne pas rentrer dans la base de donnée (max 50 crac par attributs) |
| **Pré-conditions** | Les données de cet utilisateurs ne doivent pas être déjà dans le système, ou une erreur sera retournée |
| **Post-conditions** | La donnée se retrouve dans le système |
| **Notes** |  |

* get\_db() : Récupère l’accès à la base de donnée
* inscription() : Récupère les champs du formulaire d’inscription, les valide et retourne la page de confirmation d’inscription ou les erreurs.

3.2 Modèle de classe

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nom** | Database | | | |
| **Description** | Classe s’assurant de la connexion et du management de la base de donnée | | | |
| **Attributs** |  | | | |
|  | **Nom** | <connexion> | **Type** | <NoneType> |
|  | **Description** | <Lien de connexion à la base de données> | | |

2) sans attributs :

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | <Possede> | | |
| **Description** | <Un utilisateur possède un a plusieurs événements> | | |
| **Entité source** | <Utilisateur> | **Entité destination** | <Evenements> |

4. PROCHAIN SPRINT (1 à 2 pages, max)

4.1 Revue du sprint

Revue technique :

Initialement, il avait été prévu de fournir pour le premier sprint une application d’agenda classique fonctionnelle avec la possibilité de se connecter, ainsi qu’une base de données fonctionnelle afin de pouvoir l’utiliser. Cependant, la charge de travaille a été sous-estimée et de nombreux évènements sont venus ralentir le processus.

Ce qui a pu être accompli est énoncé ci-dessous :

* + L’implémentation de la base de données avec les tables principales et les champs principaux
  + La création d’une interface utilisateur pour : la page d’accueil, la page de connexion, la page des informations de l’utilisateur.
  + L’implémentation des classes « utilisateur » et « database"
  + L’apprentissage des bases de l’outil Bootrstrap-4
  + La révision des langages de programmation « Python, Javascript »
  + La recherche et l’étude (infructueuse pour l’instant) d’outils et de frameworks pouvant faciliter le processus de création (Node.js, Extjs, Heroku, Joomla)

Ce qui a été planifié mais qui n’a pas pu être fait est :

* + L’implémentation du calendrier
  + La possibilité d’ajouter des évènements
  + La possibilité de modifier les évènements
  + L’interface graphique de L’agenda
  + L’association d’une planification à un utilisateur

Revue gestion (contenu du *sprint retrospective*)

*C*e premier Sprint a été l’occasion d’évaluer mon comportement face à l’inconnu et à des exigences diverses (Autant en programmation qu’en rédaction).

Objectivement, ce qui a été :

* + L’apprentissage des technologies a été rapide.
  + Les périodes de travails étaient de durées plutôt homogène
  + La concentration et la motivation étaient à un bon niveau au début

Ce qui n’a pas été :

* + Avec le temps les périodes de travails rétrécissaient, et s’espaçaient.
  + Un manque de concentration s’est fait sentir au courant de la semaine de relâche
  + La rédaction du Sprint Backlog s’est effectuée à la dernière minute

4.2 Plan pour prochain sprint

Dans le deuxième Sprint, il est prévu d’implémenter les cas d’utilisations suivants :

* + Afficher un calendrier
  + Consulter son agenda : Nécessite l’implémentation de l’interface graphique de l’agenda et de l’association à la classe utilisateur
  + Ajouter un événement
  + Supprimer un événement
  + Ajouter un à plusieurs objectifs

5. ANNEXES

Inclure :

1. le product backlog en date d’aujourd’hui (fin du sprint courant et début du prochain sprint)
2. Le sprint backlog du sprint courant qui se termine
3. Le sprint backlog pour le prochain sprint

Barême

|  |  |  |
| --- | --- | --- |
| Section | Pondération | Commentaire |
| 1 | 2.5% | Reprendre le contenu du ConOps. 1 à 1.5 pages, max |
| 2  2.1  2.2 | 5%  2.5%  2.5% | Reprendre contenu (amélioré) de 5.3 du ConOps  (Résumé du) contenu textuel du product backlog |
| 3  CU  DS  CO  MO | 25%  10%  2.5%  5%  7.5% | C’est l'une des deux parties majeures du document (description comportementale/fonctionnelle) |
| 4  4.1  4.2 | 7.5%  5%  2.5% | Prendre le 4.1 au sérieux  Le contenu du sprint suivant peut être différent de ce qui était prévu initialement |