DOCUMENT DE CONCEPTION

INTRODUCTION

Ce document de conception logiciel (SDD) est créé dans le cadre de décrire l'architecture et la structure du système de notre logiciel OPEN-PROJECT ainsi que l'implémentation des exigences spécifiques retenues dans le document de spécification (SRS). Pour ce faire nos études seront basée sur deux éléments principaux, notamment la conception générale et la conception détaillée.

1- OBJECTIF

La création de ce document de conception a plusieurs objectifs qui sont les suivants :

- **Décrire la structure fonctionnelle du système logiciel ainsi que les données et l'algorithme implémenté.**
- **Juger du changement des exigences spécifiques.**
- **Vérifier l'implémentation effective des exigences spécifiques.**
- **Faire les tests unitaires du système logiciel.**
- **Aider à la maintenance du système logiciel.**

2-client

Le client pour ce projet est :

Nom : Dr. DIALLO

Prénom : MOHAMED

Téléphone : 225 05 56 38 40 14

Email : diallo.med@gmail.com

CONCEPTION GENERALE

CONCEPTION ARCHITECTURALE

Dans cette partir il est question de déterminer la structure du système qui nous permet de décrire en détail les fonctions, méthodes et objet utiliser lors de la création de notre logiciel. Aussi nous allons fournir des détails sur l'algorithme utiliser et le processus de codage.

Type de référence

Ce type de référence ci-dessous présentera la manière dont sera détaillé les objets et les méthodes du logiciel OPEN-PROJECT.

- ***BUT*** : Permet de décrire la fonction ou l'objet.
- ***PROTOTYPE*** : Donne le nom de l'objet ou de la méthode.
- ***ENTRER*** : Faire la description des paramètres.
- ***SORTIR*** : Décrit les valeurs retournées.
- ***APPELER PAR*** : Objet ou fonction qui appelle notre objet ou fonction.

- ***APPELER A*** : Objet ou fonction que notre objet ou fonction appelle.
- ***ALGORITHME*** : Faire la description de l'algorithme ou la procédure.

CONCEPTION DES INTERFACES

Dans la conception des interfaces nous allons mettre en application le type de référence que nous avons définir dans la conception architecturale ci-dessus.

BUT : La bibliothèque de javascript qui nous offre un ensemble outil pour créer nos différents composants.

PROTOTYPE : App.js

ENTRER : non définie

SORTIR : non définie

APPELER PAR : class App extends React component.js

APPELER A : render ( )

ALGORITHME :

- Création du composant Menu (menu.js)

- Création du composant Article (article.js)

- Création du composant Aside (aside.js)

- Création du composant footer (footer.js)

Précision : App.js est le composant parent a l'intérieur du quel il y a d'autres composants notamment menu.js, article.js, aside.js, footer.js. Tous ces composants sont des fichiers js, chaque composant peut contenir d'autres composants appelé composant imbriqué.

Exemple :

```
< Menu/>
    <Logo/>
    <Nav/>
<Menu/>
```

CONCEPTION DETAILLEE

Pour la réalisation de notre logiciel OPEN-PROJECT, il nécessite un plan détaillé qui se fera en deux parties :

- ❖ Le Front-End
- ❖ Le Back-End

Le Front-End : Il s'agit de l'interface utilisateur.

Le Back-End : Il s'agit du coté serveur, la partie qui permet de gérer les bases de données.

De ces affirmations la question qui se pose est de savoir comment est fait le Front-End et aussi le Back-End de notre logiciel.

Comment est fait le Front-End ?

Pour ce qui est de la réalisation du Front-End, il est fait avec ReactJS.

Qu'est-ce que ReactJS ?

ReactJS (aussi appelé React ou React.js) est une bibliothèque javascript libre développée par Facebook depuis 2013 qui permet de créer des interfaces utilisateur. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état.

Pourquoi ReactJS ?

De nos jours, lorsque l'on crée un logiciel la principale problématique que l'on rencontre c'est qu'on est confronté devoir essayer de synchroniser l'Etat de notre logiciel avec l'interface utilisateur, au fur et a mesure que notre travail monte en complexité, on se retrouve avec plusieurs manipulations qui sont difficile à maintenir. Pour résoudre cette problématique, il est plus approprié d'utiliser Reactjs. Ainsi Reactjs va nous permet de mieux organiser notre interface utilisateur en séparant l'état et le fonctionnement de notre vue. Pour ce qui est d'une vue c'est une fonction de l'état.

Comment est fait le Back-End ?

Le Back-End de notre logiciel sera fait en Django.

Qu'est-ce que Django ?

Django est un Framework (cadre) de développement web open source en Python. Il a pour but de rendre le développement web 2.0 simple et rapide. Pour cette raison, le projet a pour slogan « Le Framework pour les perfectionnistes avec des deadlines ».

Pourquoi Django ?

Django est un cadre de développement qui s'inspire du principe MVC (**Modèle-vue-contrôleur**) ou MTV (la vue est gérée par un gabarit) composé de trois parties distinctes :

- ✓ Un langage de gabarits flexible qui permet de générer du HTML, XML ou tout autre format texte ;
- ✓ Un contrôleur fourni sous la forme d'un « *remapping* » d'URL à base d'expressions rationnelles ;
- ✓ Une API d'accès aux données est automatiquement générée par le cadre compatible CRUD. Inutile d'écrire des requêtes SQL associées à des formulaires, elles sont générées automatiquement par l'ORM.

En plus de l'API d'accès aux données, une interface d'administration fonctionnelle est générée depuis le modèle de données.

Django peut être considéré comme une boîte à outils où chaque module peut fonctionner de façon indépendante, il possède aussi une très bonne documentation en anglais, dont une partie, l'API stabilisée, est traduite en français. Cette documentation, au format ReStructuredText (ReST), est compatible avec le projet Sphinx.

Un système de validation des données entrées par l'utilisateur est également disponible et permet d'afficher des messages d'erreur automatiques.

Sont également inclus :

- Un serveur web léger permettant de développer et tester ses applications en temps réel sans déploiement ;
- Un système élaboré de traitement des formulaires muni de widgets permettant d'interagir entre du HTML et une base de données. De nombreuses possibilités de contrôles et de traitements sont fournies ;
- Un cadre de cache web pouvant utiliser différentes méthodes (MemCached, système de fichier, base de données, personnalisé) ;
- Le support de classes intermédiaires (intergiciel) qui peuvent être placées à des stades variés du traitement des requêtes pour intégrer des traitements particuliers (cache, internationalisation, accès…) ;
- Une prise en charge complète d'Unicode.

Les exceptions et backtraces Python sont bien gérées dans Django et apparaissent dans les pages d'erreur 500, en mode débogage. On peut avoir une bonne idée des variables d'environnement et du code ayant provoqué l'exception.