*Version 1.0 – 22/10/20*

**Résumé**

L’objectif du document est de spécifier, décrire et préciser les fonctionnalités de la solution informatique à développer.   
Le recours à des représentation visuelles permettra d’aboutir  
à une cartographie exhaustive des usages supposés.

*Ce document sera soumis à des mises à jour régulière.*



Dossier de conception fonctionnelle ET TECHNIQUE

**« Projet 7 : GrandPy Bot »**

Loïc ROMERO

Etudiant OpenClassrooms

Parcours Développeur d’Application – Python (v.1)

loic.romero1@gmail.com

**Table des matières**

[1 - Versions 1](#_Toc54346594)

[2 - Introduction 2](#_Toc54346595)

[2.1 - Objet du document 2](#_Toc54346596)

[2.2 - Liens 2](#_Toc54346597)

[2.3 - Cahier des charges 3](#_Toc54346598)

[2.3.1 - Projet 3](#_Toc54346599)

[2.3.2 - Fonctionnalités 3](#_Toc54346600)

[2.3.3 - Parcours utilisateur 3](#_Toc54346601)

[3 - Description générale de la solution 4](#_Toc54346602)

[3.1 - Les étapes de réalisation 4](#_Toc54346603)

[3.2 - Les principes de fonctionnement 4](#_Toc54346604)

[3.3 - Les acteurs 5](#_Toc54346605)

[3.3.1 - Diagramme de contexte 5](#_Toc54346606)

[3.3.2 - Liste des fonctionnalités attendues 6](#_Toc54346607)

[3.4 - Les cas d’utilisation généraux 7](#_Toc54346608)

[3.4.1 - Diagramme de packages 7](#_Toc54346609)

[3.5 - Maquette 7](#_Toc54346610)

[3.6 - Le workflow 9](#_Toc54346611)

[3.6.1 - Diagramme d’activité (workflow) 9](#_Toc54346612)

[3.6.2 - Diagramme de séquence 11](#_Toc54346613)

[4 - Les cas d’utilisations 12](#_Toc54346614)

[4.1 - Package interface web 12](#_Toc54346615)

[4.1.1 - USE CASE : Demander une adresse 12](#_Toc54346616)

[5 - Architecture de déploiement 16](#_Toc54346617)

[5.1 - Synthèse 16](#_Toc54346618)

[5.2 - Diagramme de déploiement 16](#_Toc54346619)

[5.3 - Détail des terminaux 18](#_Toc54346620)

[5.3.1 - « External » 18](#_Toc54346621)

[5.3.2 - « App » 18](#_Toc54346622)

[5.3.3 - « Users » 18](#_Toc54346623)

[6 - Glossaire 19](#_Toc54346624)

[7 - Annexes 20](#_Toc54346625)

[7.1 - Annexe 1 : Ressources 20](#_Toc54346626)

[7.1.1 - Ressources : tableau de cas d’utilisation 20](#_Toc54346627)

[7.1.2 - Ressources : carte d’user story 20](#_Toc54346628)

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Loïc ROMERO | 22/10/2020 | Création du document. | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Objet du document

Le présent document constitue le **dossier de conception fonctionnelle et technique** du « Projet 7 : Créez GrandPy Bot, le papy-robot ».

**L’objectif** **pour la partie fonctionnelle** du document est de **spécifier, décrire et préciser les fonctionnalités** de la solution informatique à développer. Pour la partie **technique**, il s’agit de **décrire les méthodes, procédés et technologies sélectionnées** pour faire face aux contraintes de réalisation du projet.

Celle-ci prendra la forme d’une **application** **web**.

Les éléments du dossier rappelleront la compréhension de l’environnement tel qu’évoqué dans le cahier des charges. Il s’agit donc d’y décrire précisément le **périmètre fonctionnel** de l’application.

Il y sera exprimé les réponses aux questions telles que :

* Quel est l’objectif de l’application?
* Qui sont les différents **acteurs** interagissant avec l’application ?
* Quelles actions cherchent-ils respectivement à faire avec l’application ?
* Quelles **fonctionnalités** vont leur permettre de mener à bien ces actions ?

Ces fonctionnalités seront décrites avec précision.

Le recours à des représentation visuelles permettra d’aboutir à une **cartographie** **exhaustive** **des** **usages** supposés. Par « usages supposés » il est induit que ce document est destiné à évoluer, avec le cas échéant, l’ajout ou même la suppression de fonctionnalités si cela s’avère adéquat (se référer à la table des **versions**).

Enfin, nous détaillerons l’**architecture** par la décomposition de ses éléments illustrée dans le **diagramme** **de** **déploiement**.

## Liens

|  |  |
| --- | --- |
| Page | Lien |
| OpenClassrooms  Parcours Développeur d’Application Projet 7 | [OC - Créez GrandPy Bot, le papy-robot](https://openclassrooms.com/fr/paths/68/projects/158/assignment) |
| Dépôt Github du projet | [Github - P7-GrandPy-Bot](https://github.com/ROL-1/P7-GrandPy-Bot) |
| Tableau de suivi Trello | [Trello - p7-grandpy-bot](https://trello.com/b/HvdsLlAh/p7-grandpy-bot) |
| Dépôt Github publique de « stopwords » au format Json | [Github - Base de "stop words"](https://github.com/6/stopwords-json/blob/master/dist/fr.json) |

## Cahier des charges

### Projet

*« […] créer un robot qui vous répondrait comme votre grand-père ! Si vous lui demandez l'adresse d'un lieu, il vous la donnera, certes, mais agrémentée d'un long récit très intéressant. »*

### Fonctionnalités

* Interactions en AJAX : l'utilisateur envoie sa question en appuyant sur entrée et la réponse s'affiche directement dans l'écran, sans recharger la page.
* Vous utiliserez l'API de Google Maps et celle de Media Wiki.
* Rien n'est sauvegardé. Si l'utilisateur charge de nouveau la page, tout l'historique est perdu.
* Vous pouvez vous amuser à inventer plusieurs réponses différentes de la part de GrandPy mais ce n'est pas une obligation. Amusez-vous !

### Parcours utilisateur

L'utilisateur ouvre son navigateur et entre l'URL que vous lui avez fournie. Il arrive devant une page contenant les éléments suivants :

* header : logo et phrase d'accroche
* zone centrale : zone vide (qui servira à afficher le dialogue) et champ de formulaire pour envoyer une question.
* footer : votre prénom & nom, lien vers votre repository Github et autres réseaux sociaux si vous en avez

L'utilisateur tape "Salut GrandPy ! Est-ce que tu connais l'adresse d'OpenClassrooms ?" dans le champ de formulaire puis appuie sur la touche Entrée. Le message s'affiche dans la zone du dessus qui affiche tous les messages échangés. Une icône tourne pour indiquer que GrandPy est en train de réfléchir.

Puis un nouveau message apparaît : "Bien sûr mon poussin ! La voici : 7 cité Paradis, 75010 Paris." En-dessous, une carte Google Maps apparaît également avec un marqueur indiquant l'adresse demandée.

GrandPy envoie un nouveau message : "Mais t'ai-je déjà raconté l'histoire de ce quartier qui m'a vu en culottes courtes ? La cité Paradis est une voie publique située dans le 10e arrondissement de Paris. Elle est en forme de té, une branche débouche au 43 rue de Paradis, la deuxième au 57 rue d'Hauteville et la troisième en impasse.

# Description générale de la solution

Cette partie va passer en revue les grands axes de la solution proposée.

## Les étapes de réalisation

1. Hébergé sur **Github**. Le tout sera réalisé selon le **Test Driven Développement**, par le biais, notamment, de « **mocks** » (api uniquement ?). En suivant les étapes **Rouge/Vert/Refactorisation** test par test.
2. Le micro-framework **Flask** est utilisé pour gérer le serveur de notre application développée en **Python**.

Deux urls au minimum seront nécessaires dont les templates seront rédigés en **HTML** pour être lus par les navigateurs clients. Possibilité de rassembler les parties communes (header/footer) aux deux pages dans un « base.html ».

Viendra en complément **Bootstrap** pour formatter la page en blocs pouvant s’adapter et ainsi avoir un tout « responsive ». (front-end).

1. Les actions GET et POST seront utilisées, notamment pour l’envois d’un **formulaire**.

Le traitement du formulaire sera réalisé grâce à **Ajax** / **JQuery** / **JavaScript** (la page n’aura pas besoin d’être actualisée).

1. Pour analyser la question posée par l’utilisateur il sera nécessaire d’utiliser un « **parser** ».

Le parser devra permettre d’identifier et transmettre les termes clefs (« **stop** **words** ») pouvant servir aux requêtes (recherches) passées aux **API externes** : GOOGLE MAPS et WIKIPEDIA.

1. Il faudra étudier les deux API : ce qu’elles demandent en entrer, ce qu’elles proposent en sortie (type d’informations et formats). (Json ?).

Afficher une carte (map) avec la localisation sous le message. (A tester avec un mock).

1. Une fois l’application fonctionnelle en local, il faudra utiliser **Heroku** pour le mettre en ligne (fichier « Procfile »).

## Les principes de fonctionnement

[…]

## Les acteurs

### Diagramme de contexte

Le visuel, ci-dessous, présente les différents types d’utilisateurs (acteurs) qui sont concernés par **l’application** à développer(**système** : représenté par le carré noir). On y distingue les utilisateurs qui auront une interaction directe avec le système (**acteurs internes** : à gauche), des utilisateurs qui ont un lien secondaire avec lui (**utilisateurs externes** : à droite).



Dans le cas présent, nous pouvons donc lister les types suivants :

Internes :

* **User** : utilisateur de l’application web.

Externes :

* **Google Maps API** : api fournit une adresse selon les mots clefs proposés et une carte de localisation.
* **Media Wiki API** : api de Wikipedia fourni des informations sur l’endroit recherché.

*Autres :*

* ***Heroku :****PaaS(Platform as a Service) qui permet de déployer une application en ligne (Cloud), prenant en charge notamment le Python. Cette plateforme créée et tiens à jour l’environnement de production nécessaire à l’application web. Heroku suggère Gunicorn comme serveur web. [N’est pas un acteur, mais une partie de la solution].*

### Liste des fonctionnalités attendues

Ci-dessous, la liste des fonctionnalités pouvant être attendues. La répartition de ces fonctionnalités et leurs descriptions détaillée seront exposées dans les chapitres suivants, parfois regroupées dans un lot de fonction sous un autre terme dans les cas d’utilisations ([partie 4](#_Les_cas_d’utilisations)).

|  |  |
| --- | --- |
| **Acteurs** | **Fonctionnalités** |
| User | * Ouvrir la page d’accueil avec l’url fournie. * Remplir un formulaire (saisir une phrase dans un unique champ texte). * Envoyer le formulaire (appuyer sur entrer / cliquer sur bouton). * Visualiser la question posée (et les suivantes) dans un champ au dessus du champ de saisie. * Visualiser une icone tourner (gif ?) le temps de la recherche. * Visualiser la réponse (adresse) (Google Map). * Visualiser une carte ciblant sur l’adresse (Google Map). * Visualiser le message complémentaire (Wikipédia). |
| Google Map Api | * Reçoit une liste de mots-clefs. * Retourne une adresse (format [ ?]). * Met à disposition une carte / ciblant sur l’adresse. |
| Media Wiki  Api | * Reçoit une adresse (ou mots clefs ?). * Retourne [x]phrases au format [Json ?]. |

Il n’y a **aucun historique**: le rechargement de la page démarre une nouvelle session.

## Les cas d’utilisation généraux

### Diagramme de packages

Le diagramme suivant représente un découpage des groupes de fonctionnalités attendues de manière à pouvoir faciliter le travail de réflexion. La séparation est ici faite par thèmes (**packages**) selon les types d’utilisations.



Nous pouvons distinguer un unique package :

Interface **web** : Toutes les fonctionnalités sont regroupées autour d’une seule interface : une page web et le formulaire qui y est inclus.

## Maquette

*Maquette du site visible page suivante.*



## Le workflow

### Diagramme d’activité (workflow)

Le diagramme disponible page suivante, présente le **workflow**. Il s’agit du **parcours** de l’utilisateur lors de l’utilisation de l’application et du parcours des **requêtes** qu’il passe au site web et qui sont relayées aux API externes. On y distingue les actions des acteurs de celles du système, en divisions verticales (« **swimlanes** »).

Légende du diagramme d’activité :





### Diagramme de séquence

# Les cas d’utilisations

Ce chapitre va présenter le package, définis précédemment, par le biais de diagrammes représentant différents cas d’utilisations. Ces cas d’utilisation représentent des **fonctionnalités ou lots de fonctionnalités** qui seront décrit dans les **descriptions textuelles**. Enfin seront proposées quelques « **users stories** » : fiches décrivant des tâches simples du point de vue de l’utilisateur. Celles-ci introduisant la conception à venir de la liste des tâches et de leur priorisation (par le biais du « product backlog » par exemple).

## Package interface web

### USE CASE : Demander une adresse



#### UC1-P1 – Cas d’utilisation « Demander une adresse »

|  |  |  |  |
| --- | --- | --- | --- |
| UC1-P1 | Auteur : Loïc ROMERO | Date : 22/10/20 | Version : 1.0 |
| NOM | **« Demander une adresse »** | | |
| PACKAGE | **Interface Web** | | |
| DESCRIPTION | L’utilisateur peut poser une question visant à obtenir une adresse. | | |
| ACTEURS | User. | | |
| PRE-CONDITIONS | L’utilisateur est sur la page d’accueil du site. | | |
| DONNEES EN ENTREE | / | | |
| SCENARIO NOMINAL | | | |
| |  |  |  | | --- | --- | --- | | N° | UTILISATEUR | SYSTEME | |  |  | Affiche le formulaire de saisie de question. | |  | Saisi une question. |  | |  | Valide l’envoi de la question. |  | |  |  | […] IDEM WORKFLOW | |  |  | |  |  | |  |  | |  |  | |  |  | | | | |
| SCENARIOS ALTERNATIFS | | | |
| |  |  |  | | --- | --- | --- | | N° | UTILISATEUR | SYSTEME | | 1 | *Aucun* | *Aucun* | | | | |
| SCENARIO D’EXCEPTION | | | |
| |  |  |  | | --- | --- | --- | | N° | UTILISATEUR | SYSTEME | | 1 | *Aucun* | *Aucun* | | | | |
|  | | | |
| FINS | Scénario nominal : A chaque étape, l’utilisateur peut quitter ou actualiser la page sur un abandon du cas d’utilisation. | | |
| POST-CONDITIONS | Scénario nominal : Le système a retourné toutes les informations, l’utilisateur peut proposer une nouvelle question. | | |
| COMPLEMENTS | | | |
| ERGONOMIE | Prévoir des indications pour aider au remplissage du champ du formulaire. Mettre en place les outils de vérification du champ.  Prévoir des messages d’erreur en cas de requêtes échouées avec les API. | | |
| PERFORMANCE | La vérification doit s’effectuer pendant la saisie de l’utilisateur (charactères autorisés).  Si le temps de réponse est trop long (à définir), le cas d’utilisation doit conduire à un échec et proposer à l’utilisateur de réaliser une nouvelle saisie. | | |
| PROBLEMES | Différences fonctionnelles ou de temps de réponse liées aux navigateurs clients.  Réponses d’échec des API. | | |

|  |  |
| --- | --- |
| Légende des « users stories » | |
| H | Priorité Haute |
| M | Priorité Moyenne |
| B | Priorité Basse |

##### User story : Cas d’utilisation « Demander une adresse »

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Un utilisateur souhaite demander une adresse à GrandPy Bot | | | H |
|  | *En tant que* | utilisateur. | | |
|  | *Je veux que* | je puisse envoyer une question à GrandPy Bot pour obtenir une adresse. | | |
|  | *Afin* | d’obtenir l’adresse, une carte et une histoire de GrandPy Bot à propos du lieu. | | |
| Critères d’acceptations | | | | |
|  | *Etant donné que* | | j’ai saisi ma question. | |
|  | *Lorsque* | | je clique sur « envoyer » ou que j’appuie sur ENTREE. | |
|  | *Je vérifie que j’obtiens bien* | | une réponse pour l’adresse, la carte et une histoire de GrandPy. | |

#### Cas d’utilisation « Actualiser la page »

|  |  |  |  |
| --- | --- | --- | --- |
| UC1-P1 | Auteur : Loïc ROMERO | Date : 22/10/20 | Version : 1.0 |
| NOM | **« Actualiser la page »** | | |
| PACKAGE | **Interface Web** | | |
| DESCRIPTION | L’utilisateur peut actualiser la page | | |
| ACTEURS | User. | | |
| PRE-CONDITIONS | L’utilisateur est sur la page d’accueil du site. | | |
| DONNEES EN ENTREE | / | | |
| SCENARIO NOMINAL | | | |
| |  |  |  | | --- | --- | --- | | N° | UTILISATEUR | SYSTEME | |  | Actualise la page. |  | |  |  | Ouvre une nouvelle page d’accueil. | | | | |
| SCENARIOS ALTERNATIFS | | | |
| |  |  |  | | --- | --- | --- | | N° | UTILISATEUR | SYSTEME | | 1 | *Aucun* | *Aucun* | | | | |
| SCENARIO D’EXCEPTION | | | |
| |  |  |  | | --- | --- | --- | | N° | UTILISATEUR | SYSTEME | | 1 | *Aucun* | *Aucun* | | | | |
|  | | | |
| FINS | Scénario nominal : La page d’accueil est à nouveau ouverte, il n’y a pas d’historique de la session précédente. | | |
| POST-CONDITIONS | Scénario nominal : / | | |
| COMPLEMENTS | | | |
| ERGONOMIE | / | | |
| PERFORMANCE | / | | |
| PROBLEMES | / | | |

##### User story : Cas d’utilisation « Demander une adresse »

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Un utilisateur souhaite obtenir une nouvelle page d’accueil. | | | H |
|  | *En tant que* | utilisateur | | |
|  | *Je veux que* | je puisse obtenir une nouvelle page d’accueil. | | |
|  | *Afin* | de commencer un nouveau dialogue avec GrandPy. | | |
| Critères d’acceptations | | | | |
|  | *Etant donné que* | | je suis sur la page web. | |
|  | *Lorsque* | | je demande à actualiser la page. | |
|  | *Je vérifie que j’obtiens bien* | | une nouvelle page d’accueil vierge de toutes données de ma session précédente. | |

# Architecture de déploiement

## Synthèse

Le **diagramme de déploiement** est une vue « statique**»** des diagrammes **UML**. Il illustre l’utilisation de l’**infrastructure** **physique** par le système et ses acteurs, ainsi que la nature de leurs **relations**. Il identifie les **éléments** **matériels** (serveurs) nécessaires et précise, au cœur des nœuds, par le biais de ses composants, quels **logiciels** seront utilisés.

La **pile logicielle** est la suivante :

|  |  |  |
| --- | --- | --- |
|  | Logiciel | Version |
| Application | Python | 3.8 |
| Framework | Flask | 1.1.2 |
| Serveur d’application (WSGI) | Gunicorn | 20.0 |
| Platform as a Service | Heroku | - |
| Serveur base de données | PostgreSQL | 12.3 |

*Les versions proposées correspondent à celles envisagée au moment de la rédaction du présent document mais sont sujettes à évoluer.*

## Diagramme de déploiement

*Schéma visible page suivante.*

*Les terminaux (« devices ») présentés sont regroupés dans trois groupes, seuls ceux identifiés par « app » (couleur bleue) font partit du système que nous développons.*



## Détail des terminaux

### « External »

Notre système sera amené à communiquer avec des terminaux externes tels que :

* « **Google Maps** » : **API** qui permettra d’obtenir une adresse postale selon les mots clefs qui lui seront transmis, ainsi qu’une carte du lieu.
* « **Media Wiki** » : **API** qui permettra d’obtenir des informations sur l’adresse, selon les mots clefs qui lui seront transmis.

### « App »

Notre solution proposera donc de mettre en place :

* Par le biais d’une **PaaS** (Platform as a Service) : **HEROKU**, notre site web sera mis en ligne (dans le cloud).
* Un serveur d’application, **Gunicorn** (conseillé par Heroku) : qui hébergera les composants logiciels que nous développerons, par le biais de la communication « WSGI » (Web Server Gateway Interface). Il contient la logique métier, cœur de notre programme, rédigé avec le langage **Python** et son framework **Flask**.
* Un serveur de base de données, **PostgreSQL**(conseillé par Heroku) qui servira à stocker, gérer et extraire les données dans une base de données (en remplacement de notre base de développement avec SQLite3).

### « Users »

Les usages des utilisateurs pouvant être sur **différents** **terminaux** nous mettrons en place une solution « **responsive** » : le site d’adaptera et se redimensionnera automatiquement selon les contraintes de résolution des **navigateurs** et de leurs supports : smartphone/tablette/ordinateur. Leurs requêtes seront transmises généralement en HTTP.

# Glossaire

|  |  |
| --- | --- |
| App | Application mobile |
| Back-end | Partie invisible par les utilisateurs du site. |
| BDD | Base De Données |
| Front-end | Partie visible par les utilisateurs du site. |
| Framework | Désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d’une partie d'un logiciel. |
| R.A.S. | Rien à signaler |
| Site | Site internet |
| UCX-PX | Use Case numéro ‘X’ – Package numéro ‘X’ |

|  |  |
| --- | --- |
| Légende des « users stories » | |
| H | Priorité Haute |
| M | Priorité Moyenne |
| B | Priorité Basse |

# Annexes

## Annexe 1 : Ressources

### Ressources : tableau de cas d’utilisation

|  |  |  |  |
| --- | --- | --- | --- |
| UCXX-PX | Auteur : | Date : XX/XX/20 | Version : 1.0 |
| NOM |  | | |
| PACKAGE |  | | |
| DESCRIPTION |  | | |
| ACTEURS |  | | |
| PRE-CONDITIONS |  | | |
| DONNEES EN ENTREE |  | | |
| SCENARIO NOMINAL | | | |
| |  |  |  | | --- | --- | --- | | N° | UTILISATEUR | SYSTEME | |  |  |  | | | | |
| SCENARIOS ALTERNATIFS | | | |
| |  |  |  | | --- | --- | --- | | N° | UTILISATEUR | SYSTEME | | 1 |  |  | | | | |
| SCENARIO D’EXCEPTION | | | |
| |  |  |  | | --- | --- | --- | | N° | UTILISATEUR | SYSTEME | | 1 | *Aucun* | *Aucun* | | | | |
|  | | | |
| FINS |  | | |
| POST-CONDITIONS |  | | |
| COMPLEMENTS | | | |
| ERGONOMIE | *R.A.S.* | | |
| PERFORMANCE | *R.A.S.* | | |
| PROBLEMES | *R.A.S.* | | |

### Ressources : carte d’user story

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | Titre | | | H |
|  | *En tant que* |  | | |
|  | *Je veux que* |  | | |
|  | *Afin* |  | | |
| Critères d’acceptations | | | | |
|  | *Etant donné que* | |  | |
|  | *Lorsque* | |  | |
|  | *Je vérifie que j’obtiens bien* | |  | |