Florent

FoodPlanning

Dossier de conception technique

Version 1.0

Auteur
Florent GRENAILLE
Développeur Python

Table des matières

1 - Versions

Auteur	Date	Description	Version
Florent	28/11/2019	Création du document	1.0

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application FoodPlanning

2.2 - Références

Pour de plus amples informations, se référer également aux éléments suivants:

1. **DCF - FoodPlanning** : Dossier de conception fonctionnelle de l'application
2. **DE – FoodPlanning** : Dossier d’exploitation de l’application (mise en production)

3 - Architecture Technique

3.1 - Composants généraux

3.1.1 - Package Authentification

3.1.1.1 - Composant Usercontrol

Se composant regroupe toute les actions qui concerne le compte utilisateur.

Son rôle est de gérer toute les taches concernant le compte utilisateur (connexion, déconnexion, inscription, changement d'informations personnelles, suppression du compte et activation/désactivation des notifications par SMS).

3.1.2 - Package Interface utilisateur

3.1.2.1 - Composant Planning

Se composant regroupe toute les actions qui concerne le planning de repas de l'utilisateur.

Son rôle est de gérer toute les taches concernant le planning de repas (création de planning, consultation de planning, modification de planning et suppression de planning).

3.1.2.2 - Composant Fooddish

Se composant regroupe toute les actions qui concerne les plats stockés en base de données.

Son rôle est de gérer toute les taches concernant les plats stockés dans la base de données (listing des plats, ajout de plat et suppression de plat).

3.1.2.3 - Composant Notification

Se composant regroupe toute les actions qui concerne les notifications SMS.

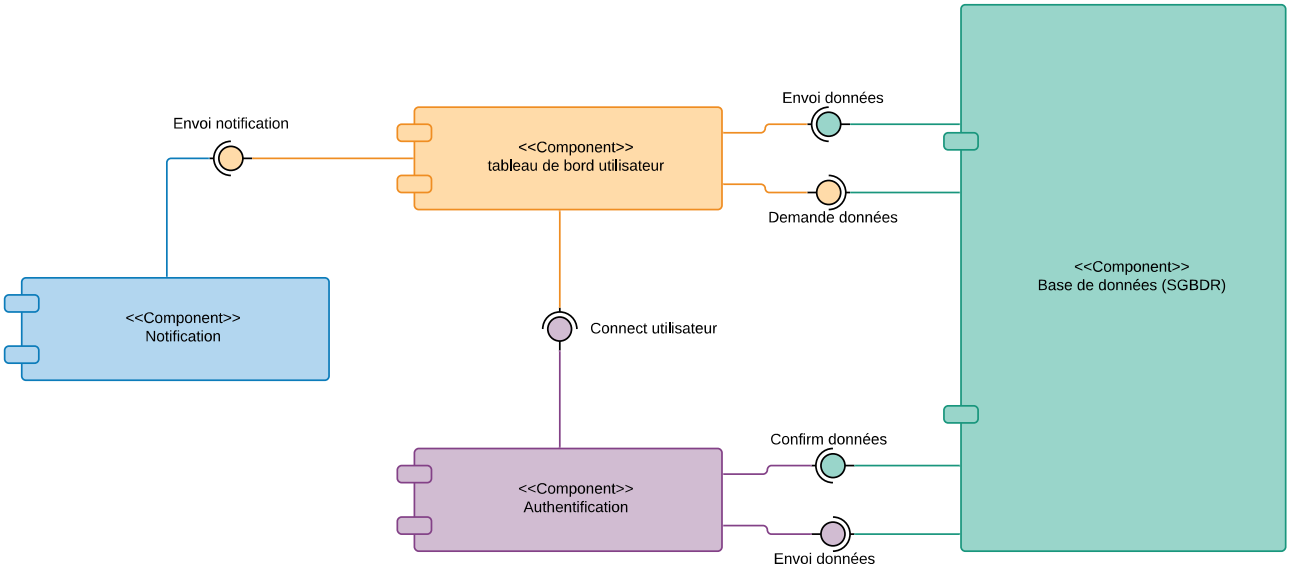
Son rôle est de gérer toute les taches concernant les notifications SMS (ajout de notification sur le planning et envoi de SMS à l'utilisateur).

3.2 - Application Web

La pile logicielle est la suivante :

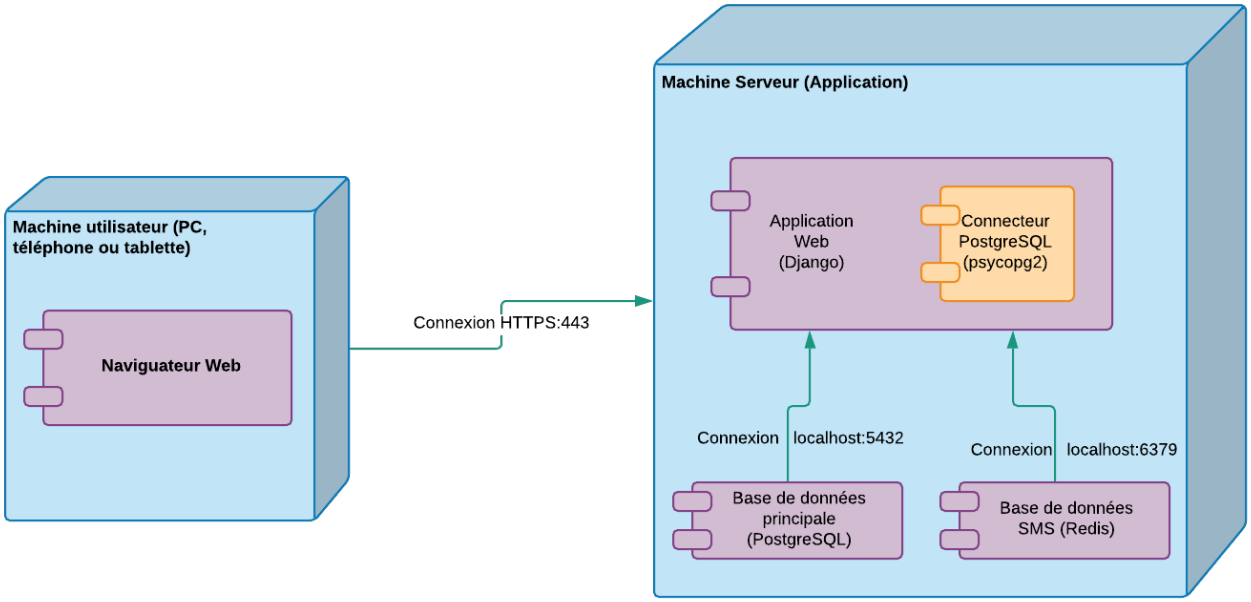
- Application **Python** (version 3.7)
- Serveur d'application **Django** (version 2.2)
- Base de données **PostgreSQL** (version 12.1)

Diagramme UML de Composants



4 - Architecture de Déploiement

Diagramme UML de déploiement



4.1 - Serveur de production

Je laisse libre choix à l'utilisateur, de déployer l'application sur le serveur qu'il souhaite (serveur physique, serveur virtuel, raspberry pi...).

La procédure de mise en production est expliquée en détail dans le **D.E** (dossier d'exploitation de l'application).

5 - Architecture logicielle

5.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git** et les dépendances par **Pipenv.**

5.1.1 - Les couches

L'architecture applicative est la suivante :

- une couche **model** : responsable de la communication avec la base de données
- une couche **view** : implémentation / gestion / opération des données
- une couche **template** : responsable de la représentation graphique des données

5.1.2 - Les modules

Les modules nécessaire à l'application :

- **django** : le cœur même de l'application
- **psycopg2-binary** : pour le liaison avec la base de données PostgreSQL
- **celery** : gestionnaire de tâche asynchrone pour Python
- **redis** : pour la liaison avec la base de données Redis
- **callr** : API d'envoi de SMS

5.1.3 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

```
racine
 ├─ manage.py
 ├─ Pipfile
 ├─ Pipfile.lock
 ├─ README.md
 ├─ requirements.txt
 ├─ .travis.yml
 ├─ .env
 ├─ fooddish
 │   ├─ __init__.py
 │   ├─ admin.py
 │   ├─ apps.py
 │   ├─ forms.py
 │   ├─ list_all_dish.py
 │   ├─ models.py
 │   ├─ urls.py
 │   ├─ views.py
 │   ├─ dishes_base.json
 │   ├─ management
 │   │   └─ commands
 │   │       └─ pull_dishes.py
```

```
│   ├ migrations
│   │   ├ __init__.py
│   │   └ 0001_initial.py
│   ├ static
│   │   └ fooddish
│   │       ├ css
│   │       │   └ listdish.css
│   │       └ js
│   │           ├ addnewdish.js
│   │           ├ deletedish.js
│   │           ├ navdishsselected.js
│   │           └ script.js
│   ├ templates
│   │   └ fooddish
│   │       └ index.html
│   └ tests
│       ├ __init__.py
│       ├ tests_integration.py
│       └ tests_unitaire.py
├ foodplanning
│   ├ __init__.py
│   ├ celery.py
│   ├ urls.py
│   ├ wsgi.py
│   └ settings
│       ├ default.py
│       ├ developpement.py
│       ├ production.py
│       └ travis.py
├ notification
│   ├ __init__.py
│   ├ admin.py
│   ├ apps.py
│   ├ callr.py
│   ├ models.py
│   ├ tasks.py
│   ├ urls.py
│   ├ views.py
│   ├ migrations
│   │   ├ __init__.py
│   │   └ 0001_initial.py
│   └ tests
│       ├ __init__.py
│       ├ tests_integration.py
│       └ tests_unitaire.py
├ planning
│   ├ __init__.py
│   ├ admin.py
│   ├ apps.py
│   ├ function.py
│   ├ models.py
│   ├ urls.py
│   ├ views.py
│   ├ static
│   │   └ planning
│   │       ├ css
│   │       │   ├ header.css
│   │       │   └ planning.css
│   │       ├ js
│   │       │   ├ addplanning.js
```

```
│   │   │   │   ├── buttonevent.js
│   │   │   │   ├── cls.js
│   │   │   │   ├── editplanning.js
│   │   │   │   ├── navamselected.js
│   │   │   │   ├── navpmselected.js
│   │   │   │   ├── notification.js
│   │   │   │   ├── removeplanning.js
│   │   │   │   ├── scripts.js
│   │   │   │   ├── secrueajax.js
│   │   │   │   ├── seeanotherplanning.js
│   │   │   │   └── validateplanning.js
│   │   │   └── img
│   │   │       ├── background_img_first.jpg
│   │   │       └── logo.png
│   │   ├── templates
│   │   │   └── planning
│   │   │       ├── another_planning_am.html
│   │   │       ├── another_planning_pm.html
│   │   │       ├── base.html
│   │   │       ├── index.html
│   │   │       └── pm.html
│   │   └── tests
│   │       ├── __init__.py
│   │       ├── tests_integration.py
│   │       └── tests_unitaire.py
│   ├── usercontrol
│   │   ├── __init__.py
│   │   ├── admin.py
│   │   ├── apps.py
│   │   ├── forms.py
│   │   ├── models.py
│   │   ├── urls.py
│   │   ├── views.py
│   │   ├── static
│   │   │   └── usercontrol
│   │   │       ├── css
│   │   │       │   ├── header.css
│   │   │       │   └── planning.css
│   │   │       ├── js
│   │   │       │   ├── changepwd.js
│   │   │       │   ├── edituserinfos.js
│   │   │       │   ├── managesms.js
│   │   │       │   ├── navaccountselected.js
│   │   │       │   └── removeaccount.js
│   │   │       └── img
│   │   │           ├── dog_cool.jpg
│   │   │           └── second_dog_cool.jpg
│   │   ├── templates
│   │   │   └── usercontrol
│   │   │       ├── account.html
│   │   │       ├── base.html
│   │   │       ├── sign_in.html
│   │   │       └── sign_up.html
│   │   └── tests
│   │       ├── __init__.py
└   │       └── tests_integration.py
```

6 - Points particuliers

6.1 - Fichiers de configuration

6.1.1 - Application web

Pour que l'application puisse être mise en production sur le serveur, celui-ci devras avoir Nginx d'installer et configuré. Tout est expliqué dans le **D.E** (Dossier d'Exploitation de l'application).

6.1.1.1 - Datasources

Le code source du projet est accessible sur github.com.

https://github.com/riderflo85/foodplanning

6.2 - Ressources

Ressources utiles destinées au développeur :

- **django** : https://docs.djangoproject.com/fr/2.2/
- **celery** : http://docs.celeryproject.org/en/latest/index.html
- **redis** : https://redis-py.readthedocs.io/en/latest/ - https://redis.io/documentation - https://redis.io/download
- **callrAPI** : https://www.callr.com/docs/

6.3 - Environnement de développement

Environnement de développement sera l'environnement fournis par Django, ainsi que l'environnement virtuel Pipenv.