Application domotique

Projet HomeCenter

Dossier de Conception Technique

Version 1.0

Auteur
Sébastien Ziegler
Etudiant/Développeur

TABLE DES MATIÈRES

1. VERSIONS

Auteur	Date	Description	Version
Sébastien Ziegler	13/06/2020	Création du document	1.0

2.INTRODUCTION

2.1. Objet du document

Le présent document constitue le dossier de conception technique de l'application "HomeCenter".

Objectif du document est de fournir les spécifications technique du projet "HomeCenter" et de modéliser la conception technique de l'application.

Les éléments du dossier technique découlent du dossier de conception fonctionnelle.

2.2. Références

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

1. **DCF - Projet HomeCenter** : Dossier de conception fonctionnelle de l'application
2. **DE - Projet HomeCenter** : Dossier d'exploitation.

3.Le "Hardware" matériel du projet.

3.1. Prérequis

Pour mettre en oeuvre le projet projet domotique "HomeCenter", vous devez disposer ou vous procurer les périphériques pris en charge décrits dans cette section, et les installer selon les documentations des fabricants (voir détails section 3.2).

3.1.1.Le contrôleur du réseau "Zwave".

Aeotec Z-Stick Gen5

https://aeotec.com/z-wave-usb-stick/

3.1.2.Les modules "Roller Shutter" pour les Volets.

Fibaro Roller-Shutter 2 (FGR-222)

https://manuals.fibaro.com/roller-shutter-2/

Depuis le développement du projet "HomeCenter",
Fibaro à sortie le module "Roller-Shutter 3" donc les numéros de paramètres diffèrent
du "Roller_shutter 2".
Ce nouveau module n'est pas pris en charge à ce jour, mais sera pris en charge dans les futures versions du projet et sera signalé sur la page principale du projet sur github dès la prise en charge.

3.1.3.Les modules "Switch" pour les lumières.

Fibaro "Double Relay Switch" (FGS-222)
https://manuals.fibaro.com/double-relay-switch/

et/ou

Fibaro 'Single Switch" (FGS-212)
https://manuals.fibaro.com/relay-switch/

NB: Les nouveaux modules de relais doubles (FGS-223) et simples (FGS-213) devraient théoriquement fonctionner mais n'ont pas été testés à ce jour.

3.2. Installation

3.2.1.Installation électrique des modules.

Dans un premier temps il faut réaliser l'installation électrique des modules domotiques et selon les schémas de la documentation du fabricant Fibaro dans la rubrique "Installation".

Roller Shutter 2(FGR-222):
https://manuals.fibaro.com/roller-shutter-2/
https://manuals.fibaro.com/content/manuals/fr/FGR-222/FGR-222-FR-A-v1.2.pdf

Double Relay Switch (FGS-222):
https://manuals.fibaro.com/double-relay-switch/
https://manuals.fibaro.com/content/manuals/en/FGS-222/FGS-222-EN-A-v1.1.pdf

Relay Switch (FGS-212):
https://manuals.fibaro.com/relay-switch/
https://manuals.fibaro.com/content/manuals/en/FGS-212/FGS-212-EN-A-v1.1.pdf

3.2.2.Configuration.

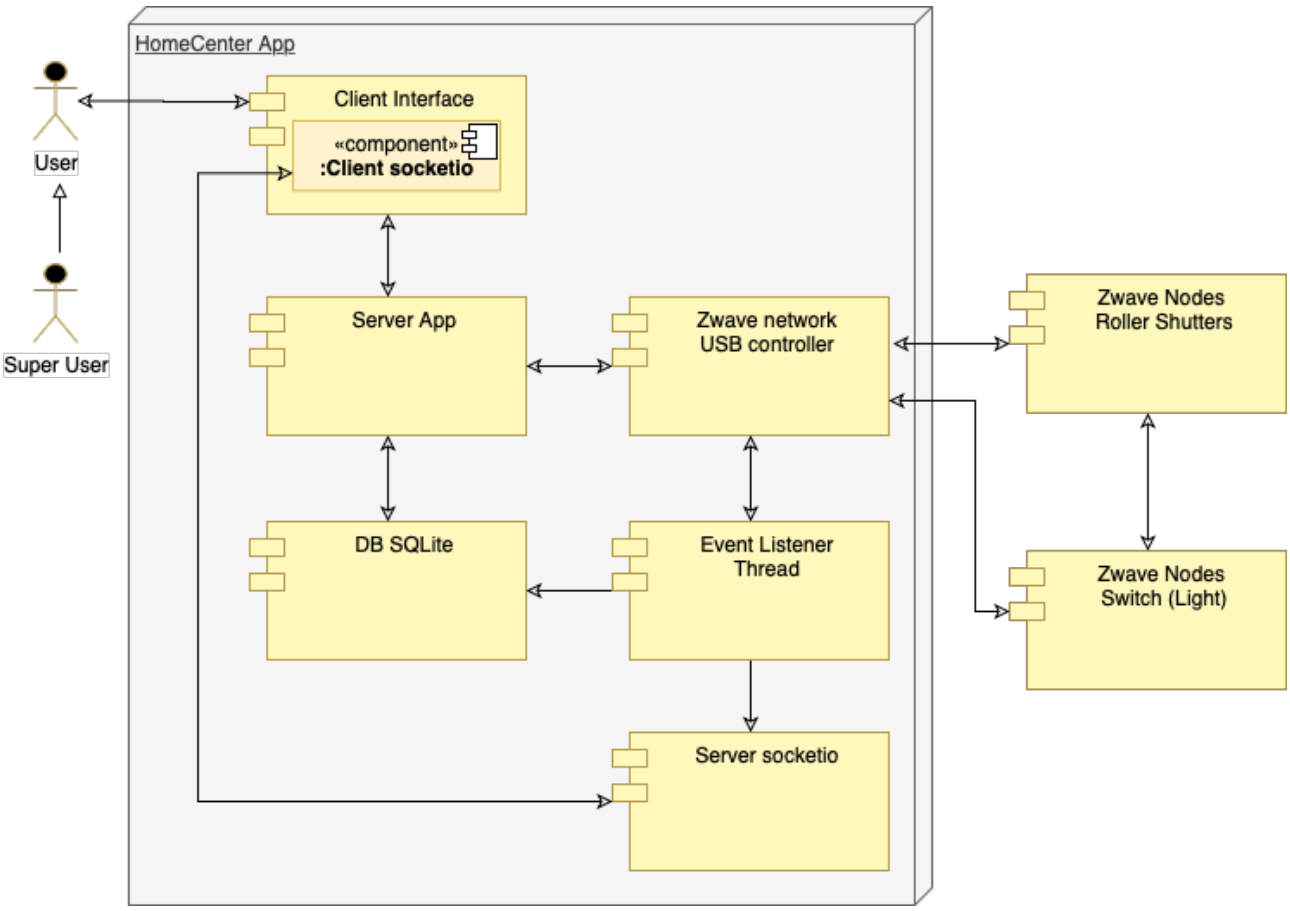
Il faut inclure la configuration de chaque module domotique dans le contrôleur USB. Cette opération se réalise avec le contrôleur non connecté au serveur. Le contrôleur est auto-alimenté électriquement par une batterie interne ce qui permet de rapprocher le contrôleur au plus près de chaque module lors de l'inclusion. Les opérations à réaliser pour l'inclusion sont les suivantes:

1. Mettre le contrôleur en mode d'inclusion (pairage) selon la rubrique "Inclusion-Mode" de la documentation Aeotec
   https://aeotec.freshdesk.com/support/solutions/articles/6000056439-z-stick-gen5-user-guide-
2. Inclure chaque module l'un après l'autre selon la rubrique "VI. COMMENT AJOUTER LE MODULE AU RESEAU Z-WAVE" selon la documentation du manuel fabricant. Le principe étant strictement le même quelque soit le type de module Figaro, vous pouvez utiliser la documentation en français ci-dessous.
   https://manuals.fibaro.com/content/manuals/fr/FGR-222/FGR-222-FR-A-v1.2.pdf
3. Après inclusion de tous les modules, insérer le stick USB dans un port USB du serveur.
4. Après installation de l'application "HomeCenter" selon la documentation, vous pourrez démarrer l'application, puis démarrer le réseau "Zwave" depuis l'interface utilisateur. Les modules inclus dans le contrôleur seront alors détectés par l'application et ceux-ci apparaîtront alors dans l'interface utilisateur.

4.Architecture Technique

4.1. Composants

4.1.1.Datasources Diagramme UML de composants



4.1.2.Description du composant "Client Interface".

Ce composant est l'interface cliente de l'utilisateur, la partie "Front-end".
L'utilisateur s'authentifie via cette interface puis, en fonction de ses droits, il peut démarrer le réseau "Zwave", piloter les modules du réseau (Volets, Lumière), configurer les instances des noeuds du réseau et administrer les utilisateurs de l'application (création/suppression de comptes). cette interface communique avec le back-end "Server App" de l'application "HomeCenter". Ce composant a également un sous composant "Client Socketio" dont le rôle est d'écouter les évènements envoyés par le serveur socketio afin d'actualiser les états des modules, en temps réel, sur toutes les interfaces clientes connectées.

4.1.3.Description du composant "Server APP".

Ce composant représente la partie "Back-end" de l'application, qui regroupe toutes les fonctionnalités. Ce composant traite toutes les demandes du client et lui retourne les résultats (pages).
Il communique avec la base de données pour:

- effectuer la mise à jour des noeuds du réseau lors du démarrage du réseau "Zwave".
- pour l'authentification, la création ou la suppression de compte utilisateur.
- pour récupèrer les derniers états de module enregistrés.

Il communique également avec le contrôleur USB du réseau pour envoyer les demandes clientes effectuées sur les modules du réseau "Zwave", tel que, le démarrage du réseau, le pilotage d'un module domotique (volet ou lumière) ou la configuration d'un module.

4.1.4.Description du composant "Zwave network USB controller".

Ce composant représente le plug USB qui contrôle les noeuds du réseau "Zwave". Il envoie aux modules, toutes les commandes reçues par le serveur d'application et reçoit tous les changements d'état des modules. Il peut également interroger l'état d'un module sur demande.

4.1.5.Description des composants "Zwave nodes" (Roller shutter et Switch).

Ces composants représentent les modules domotiques, ils reçoivent les commandes du contrôleur et modifient leurs états internes et retournent les changements d'état au contrôleur. Dans le cas spécifique des modules "Roller shutter" ou "switch", ils s'agit de relais électriques (simples, doubles ou à bascule) qui peuvent commander des appareils électriques tels que des lampes, des volets, des vannes etc…, mais ces modules pourraient être simplement des capteurs, auquel cas ils ne pourraient pas recevoir de commande mais simplement renvoyer un changement d'état ou être interrogés sur leur état par le biais du contrôleur.

4.1.6.Description des composants "DB SQLite".

Ce composant représente la base de données qui stocke toutes les informations des modules du réseau "Zwave". L'enregistrement de ces informations va permettre au programme de rechercher facilement les identifiants des noeuds et de les afficher lorsque le réseau n'est pas encore démarré. Les états y seront également enregistrés lors d'un changement, cela va permettre leur restauration sur une page lors d'un rafraîchissement par l'utilisateur.

4.1.7.Description des composants "Event Listener Thread".

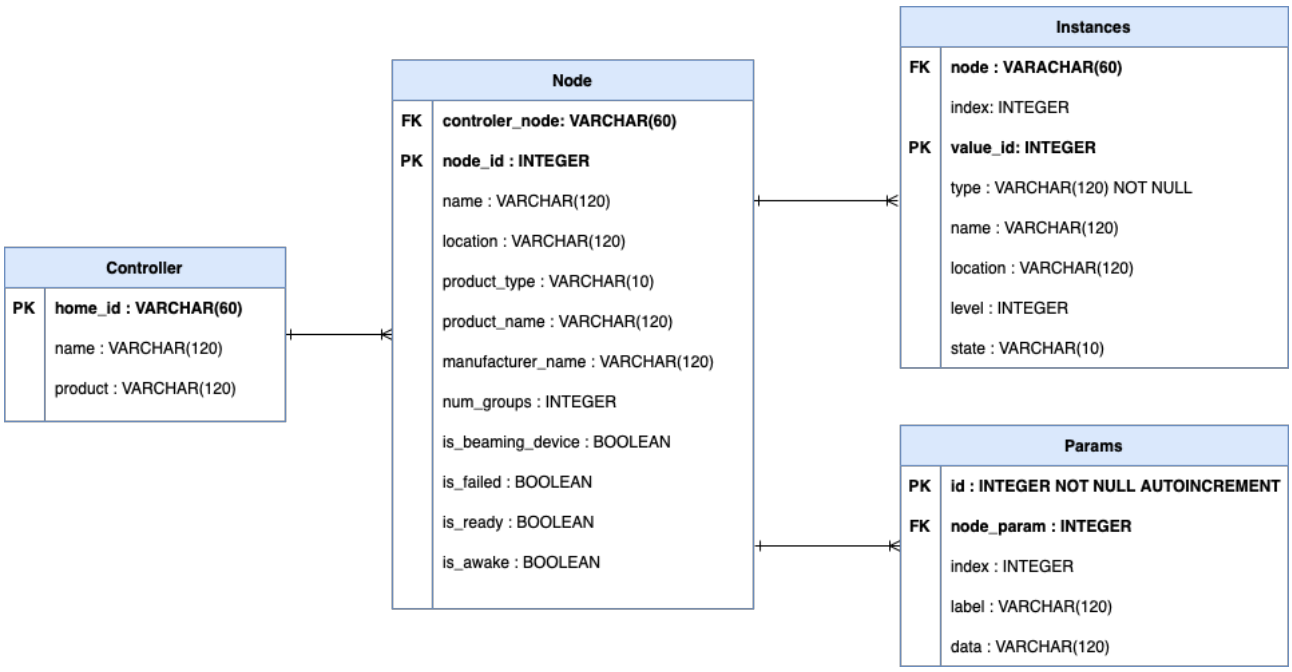
Ce composant est un "threard" qui tourne en tâche de fond de l'application et qui va écouter les événements sur le réseau "Zwave". Lorsque des changements d'états se produisent sur des modules, ce composant transmet ces modifications au composant "Server socketio" ainsi qu'au composant de base de données "DB SQLite".

4.1.8.Description des composants "Server Socketio".

Ce composant reçoit les changements d'états du composant "Event Listener Thread" et va les envoyer au client socketio du composant "Client Interface".

4.2.La base de données.

4.2.1.Le model physique de données.



4.2.2.Description générale des relations.

Le contrôleur (clé USB) peut avoir enregistré un ou plusieurs noeuds zwave (modules zwave).

Chaque noeud peut avoir une ou plusieurs instances.
Par exemple, un noeud (ou modules) de type "Roller Shutter" (volet) a une instance "switch" (interrupteur) ainsi qu'une instance "dimmer" (variateur).
De même, il existe des noeuds (ou module) de type "switch"(interrupteur) ou "double switch" qui disposent respectivement d'une ou deux instances "switch".

Chaque noeud dispose de plusieurs paramètres.

4.2.3.Les règles de gestion.

1. Un **contrôleur** a un ou plusieurs **noeuds**
   Chaque **noeud** est affecté a un **contrôleur**.

2. Un **noeud** dispose d'une ou plusieurs **instances**.
   Chaque **instance** appartient à un **noeud**.

3. Un **noeud** dispose de plusieurs **paramètres**.
   Chaque **paramètre** appartient à un **noeud**.

4.3. Applications Backend

La pile logicielle est la suivante :
- asgiref==3.2.7

- certifi==2020.4.5.1
- cffi==1.14.0
- chardet==3.0.4
- coverage==5.1
- cryptography==2.9.2
- Cython==0.29.19
- Django==3.0.4
- dnspython==1.16.0
- enum-compat==0.0.3
- eventlet==0.25.2
- gevent==20.5.2
- greenlet==0.4.15
- gunicorn==20.0.4
- idna==2.9
- Louie==2.0
- monotonic==1.5
- pycparser==2.20
- PyDispatcher==2.0.5
- pyOpenSSL==19.1.0
- pyserial==3.4
- python-engineio==3.12.1
- python-openzwave==0.4.19
- python-socketio==4.5.1
- pytz==2019.3
- requests==2.23.0
- six==1.14.0
- sqlparse==0.3.1
- urllib3==1.25.9
- zope.event==4.4
- zope.interface==5.1.0

Cette pile logiciel sera installée automatiquement lors de l'installation de l'environnement virtuel en suivant les instructions décrites dans le fichier "README.md" à la racine du projet Github.

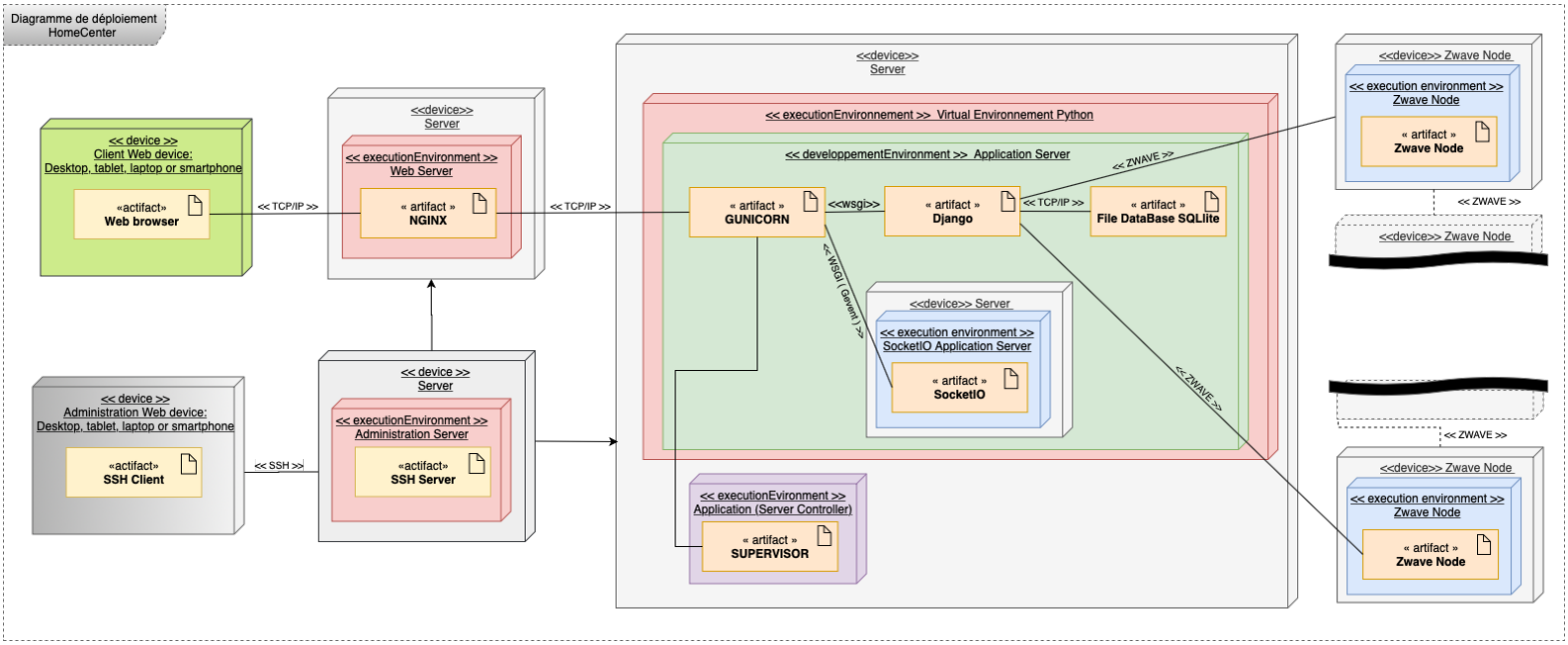
4.4. Applications Frontend

La pile logicielle est la suivante:

- JQuery >= v 3.4.1
- Bootstrap >= v 4
- Socketio >= v 2.3.0
- materialdesignicons >= v 4.9.95
- iconify-design >= v 1.0.4

5.Architecture de Déploiement

Diagramme UML de déploiement



Le système sera constitué de 5 serveurs logiciels installés sur un serveur physique, internes au système "HomeCenter"

5.1. Serveur web NGINX (Proxy inverse)

Le système utilise le serveur Nginx qui est un serveur web HTTP proxy inverse, qui va rediriger toutes les requêtes entrantes, effectuées par le client via son navigateur web sur l'adresse ip public, vers l'adresse ip privée du serveur de production Gunicorn et renverra le résultat de la requête au navigateur web du client. Nginx se servira directement des fichiers statiques sans passer par le serveur d'application Gunicorn. La communication se fait par protocole TCP/IP. Nginx est installée sur le serveur physique.
Nginx communique simultanément avec le serveur d'application en mode synchrone et avec le serveur SocketIO en mode asynchrone via la bibliothèque "Gevent".

5.2. Serveur d'application Gunicorn (synchrone)

Gunicorn est un serveur d'application wsgi de production qui s'exécute dans l' environnement python3 et qui peut recevoir les requêtes HTTP et renvoyer le résultat directement depuis une adresse IP local . Gunicorn exécute le code de l'application Django par le biais de l'interface wsgi (Web Server Gateway Interface). Il reçoit des requêtes du serveur Nginx et lui retourne les résultats. Gunicorn est installé dans l'environnement virtuel python 3, installé sur le serveur physique.

5.3.L'environnement de développement Django

Django est l'environnement d'exécution dans lequel se trouve le code python de l'application qui est interprétée et exécutée par le serveur d'application Gunicorn. Django fonctionne également dans l' environnement virtuel python3 , installé sur le serveur physique.

5.4.Le serveur 'SocketIO" (Asynchrone)

Dans l'application "HomeCenter", le serveur "SocketIO" va permettre la communication asynchrone entre le réseau "OpenZwave" et le client web, afin, notamment, de mettre à jour, le statut des lampes ("On" ou "Off") ainsi que les niveau d'ouverture de volet sans rechargement de la page web.
Dès lors que l'état d'un module est modifié et actualisé sur le réseau "OpenZwave", le "serveur socketIO" émet le nouvel état, à tous les clients connectés. Ce nouvel état est ensuite intercepté par le "client socketIO" qui met à jour les statuts des pages web de tous les clients connectés.
L'utilisation de la bibliothèque "Gevent" par l'application "HomeCenter", va permettre la communication asynchrone via le serveur d'application Gunicorn basé sur une communication synchrone (WSGI).

5.5.Les noeuds 'Zwave"

L'application "HomeCenter" communique avec les noeuds (modules) zwave par le protocole sans fils "zwave" grâce au contrôleur USB. Les noeuds sont donc des modules électrique qui peuvent être:

- des relais, de lampes, de volets etc…
- des capteurs, de proximité, de présence etc...
- des actionneurs tel que des moteurs, des vannes etc…

5.6. L'application de surveillance Supervisor.

Supervisor est une application de surveillance de processus. Il surveille les processus du serveur Gunicorn afin de s'assurer que celui-ci est bien en fonctionnement. Lorsque le processus Supervisor est démarré, celui-ci démarre automatiquement le serveur Gunicorn si ce dernier est à l'arrêt ou tente de le redémarrage si le démarrage à échoué. Supervisor est installé sur le serveur physique.

5.7. Le fichier de base de données SQLite (SGBD)

Le système utilise SQLite, qui est un simple fichier de base de données qui reçoit des requêtes SQL de l'application et qui lui retourne les résultats des requêtes. SQLite fonctionne grâce au pilote SQLite installé par défaut dans Django. Il n'y a donc pas de serveur à installer. L'application ne nécessite pas de grosse montée en charge pour communiquer avec la base de données, de plus, le nombre d' utilisateurs étant limité à quelques membres de la famille, un serveur de SGDB tel que MySQL ou PostgreSQL serait donc superflu. La base de données SQLite de l'application, stock toutes les informations des différents modules domotiques présents sur le réseau (identifiants, états est…).

5.8. Le serveur d'administration SSH (Secure Shell)

Il est recommandé d'installer et de configurer sur le serveur SSH sur le serveur physique. Cette option est indispensable si l'on ne dispose pas d'un accès direct au serveur avec un écran et un clavier. Celui-ci va permettre à l' administrateur, d'installer et de configurer le serveur à distance en mode console par le protocole SSH de communication sécurisé en utilisant une application cliente ssh (ex: Putty). Toutefois, "HomeCenter" peut être installé et configuré avec un accès direct au serveur en mode "console" via un écran et un clavier.

5.9. L'infrastructure

Le système est déployé sur un serveur local du domicile à domotiser. Le système fonctionne dans un environnement virtuel Python3 sur un serveur de type Raspberry Pi 3 ou supérieur.

6.ARCHITECTURE LOGICIELLE

6.1.Principes généraux

Les sources et versions du projet sont gérées par **Github**. L'intégration continue sera gérée par Travis-ci.

6.1.1.Les applications du projet.

L'architecture applicative est différente de celle des packages.
Le projet est donc composé des applications suivantes:

- L'application "**account**", dans laquelle est implémentée toute la partie authentification ((login, logout) ainsi que la gestion des utilisateurs (création/suppression de compte).
- L'application "**homecenter**", dans laquelle sont implémentées toute les fonctionnalités de l'application (Pilotage, configuration et communication des noeuds, gestion du réseau, création et remplissage de la base de données etc….)
- L'application "**homecenter_project**", qui est l'application principal du projet et qui contient toute la configuration du projet.
- L'application "**socketio_app**" qui contient l'initialisation du serveur socketio. Cette application aurait pu être remplacée par un simple fichier mais elle est présente dans l'optique d'évolution de l'application "HomeCenter" et pour séparation des différents serveurs.

Ces applications sont développées dans l'**environnement de développement** "**Django**" qui est installé dans un **environnement virtuel** "**Python 3**" avec tous les packages nécessaires au fonctionnement des applications (les "**requirements**").
Le serveur d'application "**Gunicorn**" est également installé dans l'**environnement virtuel python**.

6.1.2.Les applications du serveur hébergeur local.

Les serveurs logiciels, décrits dans la partie déploiement, sont installé sur le serveur physique local qui héberge l'application "HomeCenter" sous Linux ou Windows.
L'architecture logiciel installée sur serveur est donc la suivante.

- Le serveur proxy inverse **NGINX** .
- **Python 3,** l'environnement virtuel **Virtualenv** ou **pipenv**.
- Les packages de l'environnement virtuel qui comprend le serveur **Gunicorn** et le serveur **socketio.**
- Le serveur d'administration **SSH** (open-ssh) (optionnel pour une utilisation direct du serveur avec un clavier et un écran).
- Le protocole sécurisé **SSL** (open-ssl). (optionnel pour une utilisation intranet)

7.POINTS PARTICULIERS

7.1.Fichiers de configuration

7.1.1.Configuration de l'application "HomeCenter" en production.

Le fichier de configuration de production "product.py" doit-être créé pour la mise en production de l'application "HomeCenter". Ce fichier contient les variables d'environnement de production, les données d'accès à la base de données SQLite et les paramètres de votre serveur mail SMTP. Créer ce fichier "product.py" dans le répertoire "homecenter_project/settings/" de l'application, copier/coller les paramètres ci-dessous, personnaliser les champs selon les commentaires dans le fichier et enregistrer le fichier.
Les paramètres de ce fichier doit être complétés directement sur le serveur avec vos paramètres spécifiques.

production.py:

```
from . import *
SECRET_KEY = "Votre clé d'application" # Remplacer par votre clé d'application
DEBUG = False
ALLOWED_HOSTS = ['votre adresse ip', 'votre non de domaine'] # Entrer ici vos adresses ip d'accès
et/ou vos nom de domaine, séparé par des virgules.

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'homecenter_db.sqlite3'),
    }
}

# Configurer ici votre serveur de mail SMTP pour la réinitialisation du mot de #passe
# utilisateur
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'mail.gandi.net'# Remplacer par le nom d'hôte de votre serveur mail d'envoi
EMAIL_PORT = 587  # Remplacer par le port de votre serveur mail
EMAIL_USE_TLS = True  # Activation du Protocol TLS
EMAIL_HOST_USER = 'mon_d'utilisateur_du_serveur_SMTP'  # Remplacer par le nom d'utilisateur de votre
serveur SMTP
EMAIL_HOST_PASSWORD = 'mot_de_passe_du_serveur_SMTP'  # Remplacer par le Mot de passe de votre
serveur SMTP
DEFAULT_FROM_EMAIL = 'email@nom_de_domaine.com'  # Remplacer par votre adresse mail d'envoi par
défaut
SERVER_EMAIL = 'serveur_mail_envoi'  # Remplacer par le votre serveur mail d'envoi
```

7.1.2.Fichier de configuration NGINX (/etc/nginx/sites-available/homecenter.conf)

homecenter.conf:

```
server {
        listen 80;
        listen [::]:80;
        # Commenter les lignes ci-dessus et décommander les lignes ci-dessous en mode SSL
        #listen 443;
        #listen [::]:443;

        # Logs
        access_log  /var/log/nginx/permit_access.log;
        error_log   /var/log/nginx/permit_error.log;

        # Entrer l'adresse ip de votre serveur et/ou votre nom de domaine
        server_name localhost mon_adresse_ip mon_nom_de_domaine;

        # SSL
        # Décommenter les lignes ci-dessous si vous utiliser un certificat SSL
        # ssl on;
        # ssl_certificate /chemin/mon_certificat.crt;  # Remplacer par votre fichier certificat crt
        # ssl_certificate_key /chemin/mon_certificat.key;  # Remplacer par votre clé de certificat

        # Root directory
        root /chemin_vers_projet/homecenter_project/; # Remplacer par votre chemin d'installation

        # Staticfiles location
        location /static {
        alias /chemin_vers_projet/homecenter_project/static/;# Remplacer par votre chemin d'install.
        }

        location / {
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_redirect off;
                if (!-f $request_filename) {
                    proxy_pass http://127.0.0.1:5000; # Remplacer le port si vous le modifier
                    break;
                }
                proxy_set_header X-Real-IP $remote_addr;
        }

        location /socket.io {
                proxy_pass          http://0.0.0.0:5000/socket.io; # Remplacer le port si vous le
                                                                   # modifier
                proxy_redirect      off;
                proxy_buffering     off;

                proxy_set_header    Host              $host;
                proxy_set_header    X-Real-IP         $remote_addr;
                proxy_set_header    X-Forwarded-For   $proxy_add_x_forwarded_for;

                # Websockets support
                proxy_http_version 1.1;
                proxy_set_header    Upgrade           $http_upgrade;
                proxy_set_header    Connection        "upgrade";
        }
}
```

7.1.3.Fichier de configuration Supervisor (/etc/supervisor/conf.d/homecenter.conf)

homecenter.conf:

```
[program:homecenter]
# Remplacer par le chemin vers le serveur d'application gunicorn de votre environnement virtuel
command=/home/chemin_vers_serveur_app/bin/gunicorn -k gevent -w 1 -b 127.0.0.1:5000
homecenter_project.wsgi:application

# Remplacer par le nom d'utilisateur de votre session linux
user = votre_nom_utilisateur_de_session
directory = /home/chemin_de_votre_projet/homecenter_project
autostart = true
autorestart = true
environment = DJANGO_SETTINGS_MODULE='homecenter_project.settings.production'
```

7.1.4.Applications web

Les applications sont installées depuis le repository Github.

La procédure d'installation complète est décrite dans le fichier README.md du repository.

7.1.5.Datasources

Le fichier de base de données s'appelle par défaut "homecenter_db.sqlite3". Ce fichier sera créé dynamiquement lors du démarrage du réseau "Zwave" dans l'application "HomeCenter". L'application "HomeCenter" enregistre toutes les données des modules configurés sur la clé USB dans ce fichier de base de données. Si des modules sont ajoutés ou supprimés de la clé USB et que ces modifications ne sont pas prises en compte par l'application, vous pouvez simplement supprimer ce fichier et relancer le réseau "Zwave" pour le recréer.

7.2.Ressources

Afin de fonctionner dans les meilleurs conditions, un serveur qui dispose au minimum de:

- 1Gb de RAM
- Un processeur CPU 4 coeurs 1,2 GHz
- Un espace de stockage de 16Gb.

Un serveur de type Raspberry Pi 3b ou supérieur, avec un sytème d'exploitation Linux tel que Raspbian (Debian pour Raspberry) ou Debian sont des choix pertinents.
Le système 'HomeCenter" étant déployé dans un environnement virtuel Python3, une installation sur un serveur Linux en mode graphique ou sous un autre système d'exploitation tel que "Windows 10 lite" est également possible. Néanmoins, dans le cas d'une utilisation en mode graphique sous "Windows 10 lite", il serait raisonnable de prévoir une machine un peu plus puissante avec un minimum de 2GB de mémoire RAM, le système lui même étant plus gourmand en ressources. L'installation sous un sytème windows sort du cadre de cette documentation (fichier d'exemple ci-dessus sont pour Linux)

7.3.Protocole SSL (Pour un accès depuis l' internet)

Pour un accès à "HomeCenter" depuis l'internet, il est préférable, pour des raisons évidentes de sécurité, d'installer le protocole SSL sur le serveur afin d'accéder à l'application domotique "HomeCenter" via des pages web https cryptées et sécurisées.
L'installation du protocole SSL sort du cadre de cette documentation. Veuillez vous référer à la documentation en ligne du serveur NGINX.

7.4.Environnement de développement

L'environnement de développement est Django version 3.0.4.

7.5.Procédure de packaging / livraison

Le système "HomeCenter" est distribué gratuitement sur le site Github du projet.
La procédure d'installation et de déploiement est décrite à la racine du repository Github (Fichier "README.md.")

https://github.com/Mystic67/homecenter_project