Conception d'un système de gestion des carnets de suivis en maison de retraite

SPÉCIFICATIONS TECHNIQUES

Lekishvili Rati - Développeur d'Application Python
Projet n°13
19/12/2021

Sommaire

I. VERSIONS

Auteur	Date	Description	Version
LEKISHVILI Rati	20/12/2021	Création du document de spécifications techniques	1.0

II. INTRODUCTION

A. Rappel du contexte

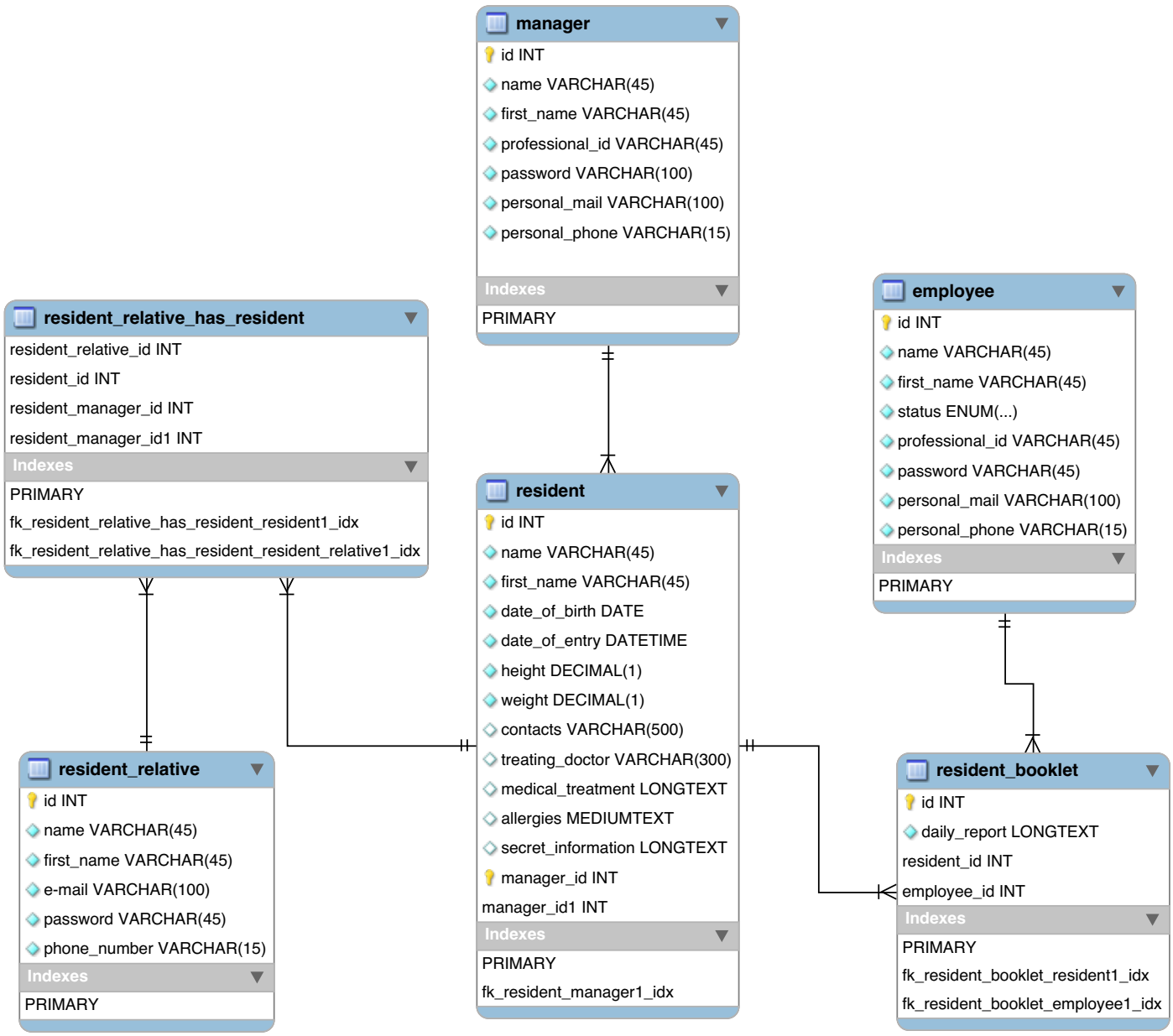
Dans le contexte actuel, en raison de la crise sanitaire, les personnes âgées dans les maisons de retraite ainsi que leurs proches doivent faire face aux restrictions et parfois même à l'interdiction de visites physique. Ces mesures de sécurité pour protéger les personnes les plus vulnérables au COVID-19 ont néanmoins l'inconvénient majeur de priver les familles et leurs proches d'informations sur l'état de santé des résidents au quotidien. Si la structure de la maison retraite est importante et le personnel se retrouve dépassé par les évènements, la communication entre les proches des résidents et les professionnels de santé peut alors être compromise en raison de la charge du travail en plus sur le personnel soignant.

Le nouveau système informatique prévoit de facilité le transfert d'information entre les maisons de retraite et les proches des résidents, en délivrant un compte rendu quotidien sur l'état de santé de chaque résident, facilement accessible sur le web.

B. Objet du document

Le présent document constitue le dossier de spécifications techniques de l'application qui sera développée. L'objectif du document est de définir le domaine fonctionnel et de concevoir l'architecture technique de l'application.

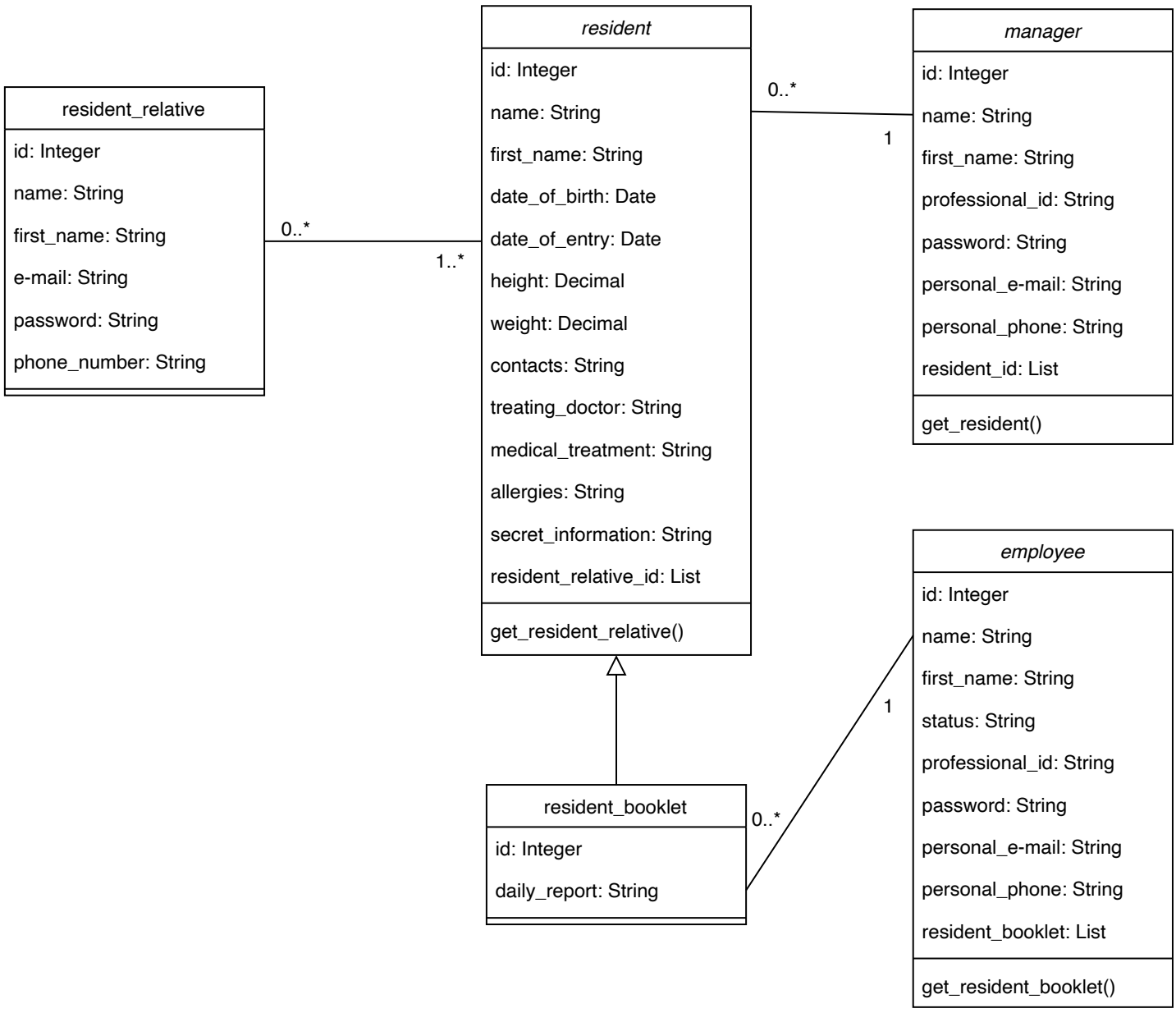
III. MODÈLE PHYSIQUE DE DONNÉES



IV. LE DOMAINE FONCTIONNEL

A. Le diagramme de classes

Le diagramme de classes (UML) est utilisé lors de la conception d'un logiciel pour représenter les classes ainsi que leurs relations et cardinalités. Ce dernier permet de montrer la structure interne du système, en fournissant une représentation abstraite des différents objets composants le domaine fonctionnel.



1. Règles de gestion

Définition des classes

* **resident** : désigne les résidants d'une EPHAD.
* **resident_relative** : désigne les proches des résidents.
* **resident_booklet** : représente les fiches / carnets des résidents visible par les proches.
* **manager** : regroupe les responsables dans un établissement.
* **employee** : regroupe les employé, le personnel soignant d'un établissement.

Relations entre les classes

- *Relation (n:n) entre « **resident** » et « **resident_relative** ».*
  Un résident peut être associé à zéro ou plusieurs proches, et le proche d'un résident (au minimum) peut avoir plusieurs résidents dans une même EPHAD.

- *Relation (1:n) entre « **resident** » et « **manager** ».*
  Un résident est associé à un seul manager, celui qui a créé la fiche du résident. Un manager peut avoir créé plusieurs resident dans le système.

- *Relation (1:n) entre « **resident_booklet** » et « **employee** ».*
  Une fiche résident peut être à un seul employé, celui qui a créé la fiche. Un employé peut avoir créé plusieurs fiches dans le système.

- *Relation entre (héritage) « **resident** » et « **resident_booklet** ».*
  Resident_booklet hérite de plusieurs attributs de la classe resident.

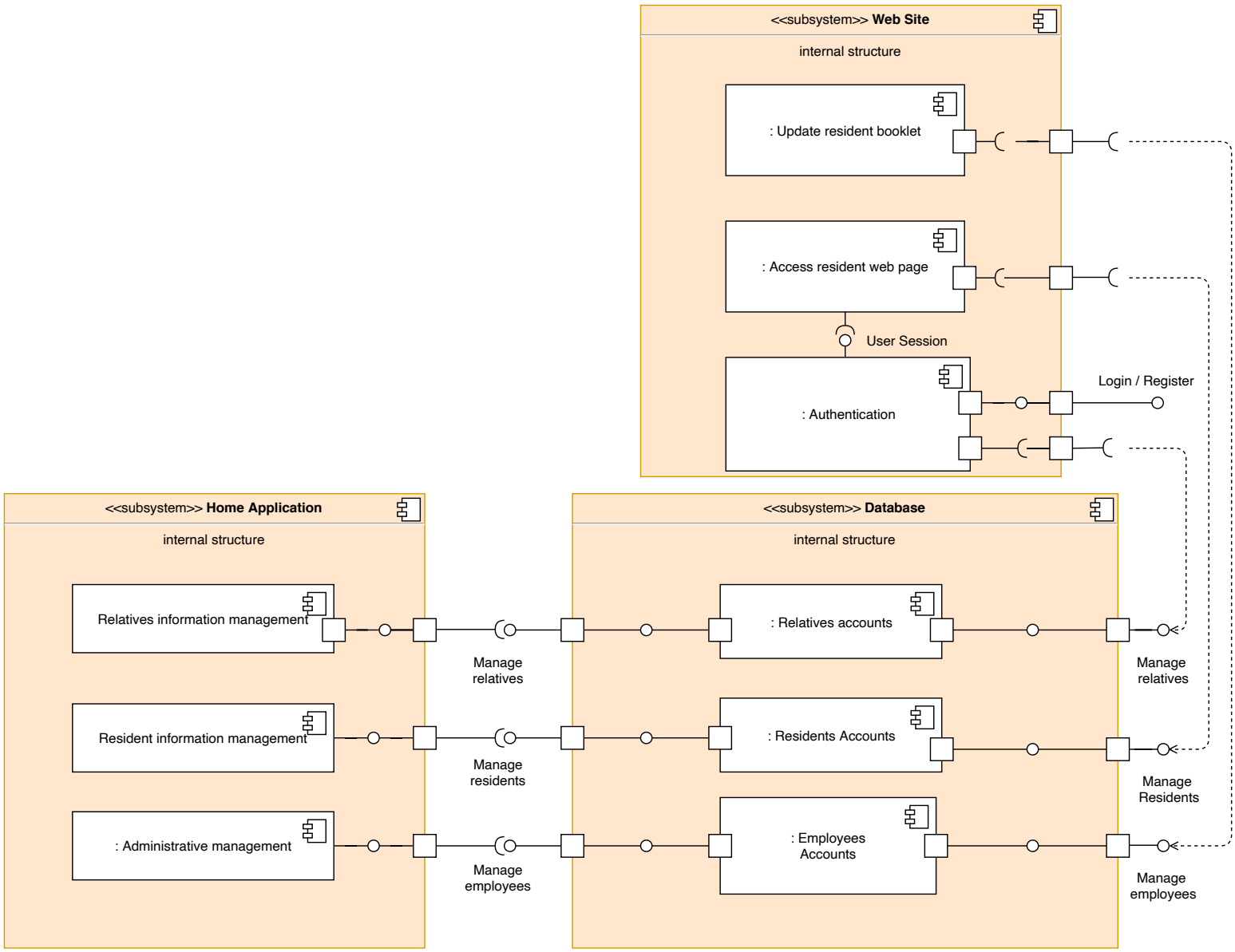
V. ARCHITECTURE TECHNIQUE

A. Diagramme de composants

Diagramme UML de Composants

Un diagramme de composants a pour objectif d'illustrer les relations entre les différents composants d'un système.

Les « subsystem » représentent les grandes familles de composants.



Les composants :

Web Site

Le composant « Web Site » fournit aux utilisateurs l'interface permettant de se connecter ou de créer un compte pour accéder à l'application web. Pour permettre aux différents acteurs de s'authentifier, ce composant aura besoin des informations provenant de la base de données.

Database

« Database » regroupe les informations concernant les différents acteurs : résidents, employés, proches et responsables. Ce composant permet aux deux autres composants « Web Site » et « Home Application » de gérer les informations relatives aux acteurs du système.

Home Application

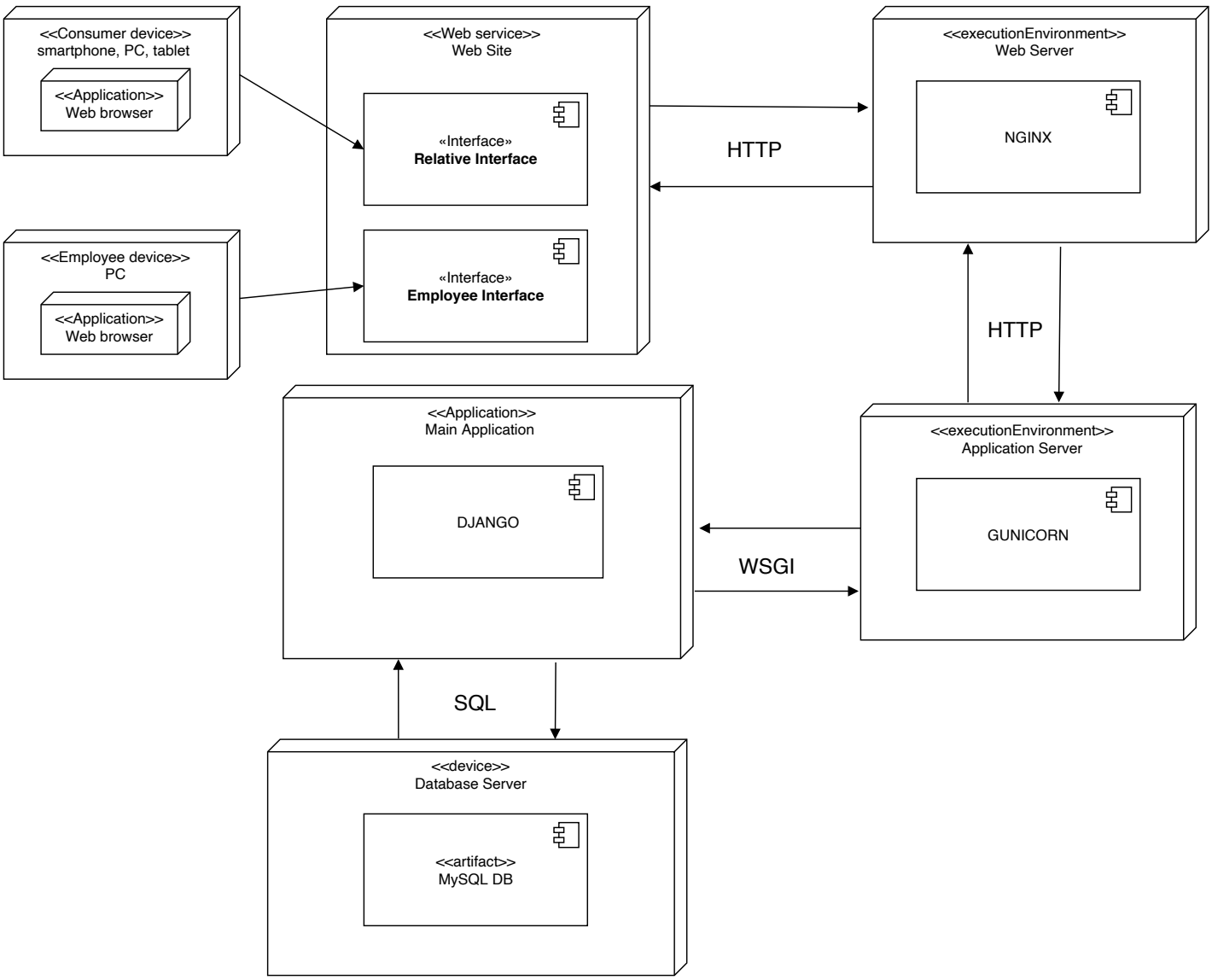
Le composant « Home Application » représente l'application interne pour les salariés, et les responsables et leurs permet de gérer les carnets des résidents ainsi qu'aux responsables de gérer le personnel. Ce composant requiert les informations employés, résident et proches des résidents du composant « Database ».

VI. ARCHITECTURE DE DÉPLOIEMENT

La pile de logiciels est la suivante :

- Application : **Python 3.10.1** (https://www.python.org/downloads/)
- Framework : **Django 4.0** (https://www.djangoproject.com/download/)
- Serveur d'application : **Gunicorn 20.1.0** (https://gunicorn.org)
- Serveur web : **NGINX 1.21.5** (http://nginx.org/en/download.html)
- Serveur base de données : **MySQL 8.0.27** (https://dev.mysql.com/downloads/mysql/)

Diagramme UML de déploiement



Le diagramme de déploiement décrit la disposition physique des ressources matérielles qui composent le système. Une ressource est représenté par un noeud, le diagramme nous aide à schématiser une vue statique de ces ressources ainsi que les connexions qui existent entre elles.

Application

L'application sera développée à l'aide de Python 3, et de son framework web le plus populaire **Django**.

Serveur de base de données

Le choix du serveur de la base de données s'est porté sur MySQL, qui est un système de gestion de base de données relationnelle populaire et reconnu pour sa fiabilité. MySQL utilise le langage SQL pour les opérations CRUD sur ses bases de données.

Serveur d'application Gunicorn

Gunicorn (Green Unicorn), est un serveur d'application HTTP python, qui communiquera avec notre application à l'aide de spécifications WSGI. WSGI ou Web Server Gateway Interface est une spécification qui définira une interface entre le serveur web et notre application web.

Serveur web NGINX

Le serveur web NGINX traite les requêtes HTTP envoyées par les clients. Associé à Gunicorn les deux serveurs permettent un **traitement plus rapide des demandes clients**, en raison de leurs complémentarité. NGINX se chargera de traiter les fichiers statiques et laissera Gunicorn s'occuper des fichiers dynamiques.