Maïeuclic

Dossier de conception technique

Version 1.0

Auteur
Eloïse Lamontagne
Développeuse d'applications

Table des matières

1 - Versions

Auteur	Date	Description	Version
Eloïse Lamontagne	05/01/21	Création du document	01/01/00

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier de conception technique du site internet Maïeuclic.

Ce document comprend :

- une description du domaine fonctionnel
- Le modèle physique de données (MPD) ou modèle relationnel de la base de données
- les différents composants du système et les composants externes utilisés par celui-ci et leur interaction
- la description de l'organisation physique de ces composants (déploiement)

Les éléments du présent dossier découlent :

- de réflexions personnelles et d'échanges avec de potentiels utilisateurs.

2.2 - Références

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

1. **DCF – 1.0** : Dossier de conception fonctionnelle de l'application

3 - Domaine fonctionnel

3.1 - Référentiel

Le diagramme de classes qui suit servira de support au programmeurs. Il représente l'organisation de l'information.

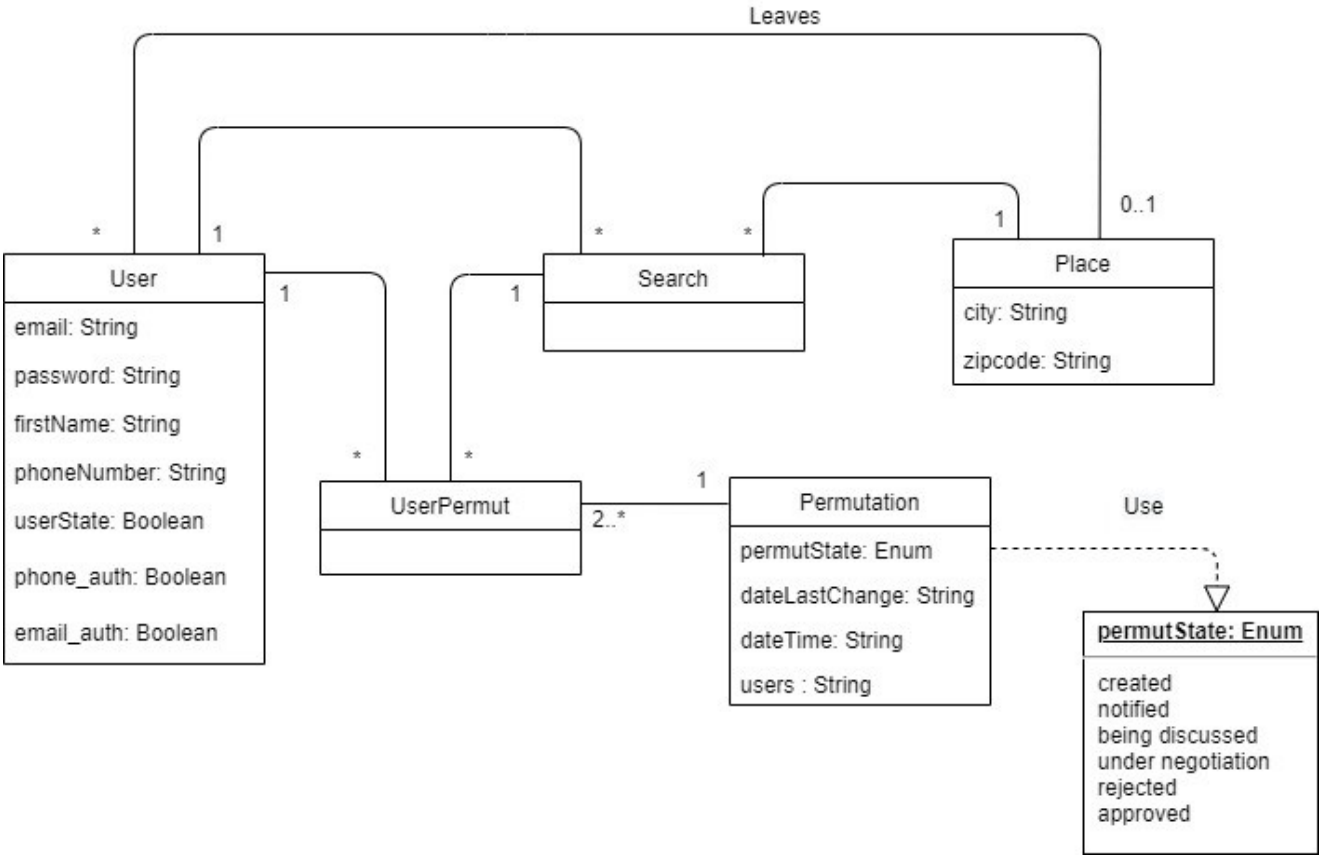


Diagramme de classes

3.1.1 - Règles de gestion

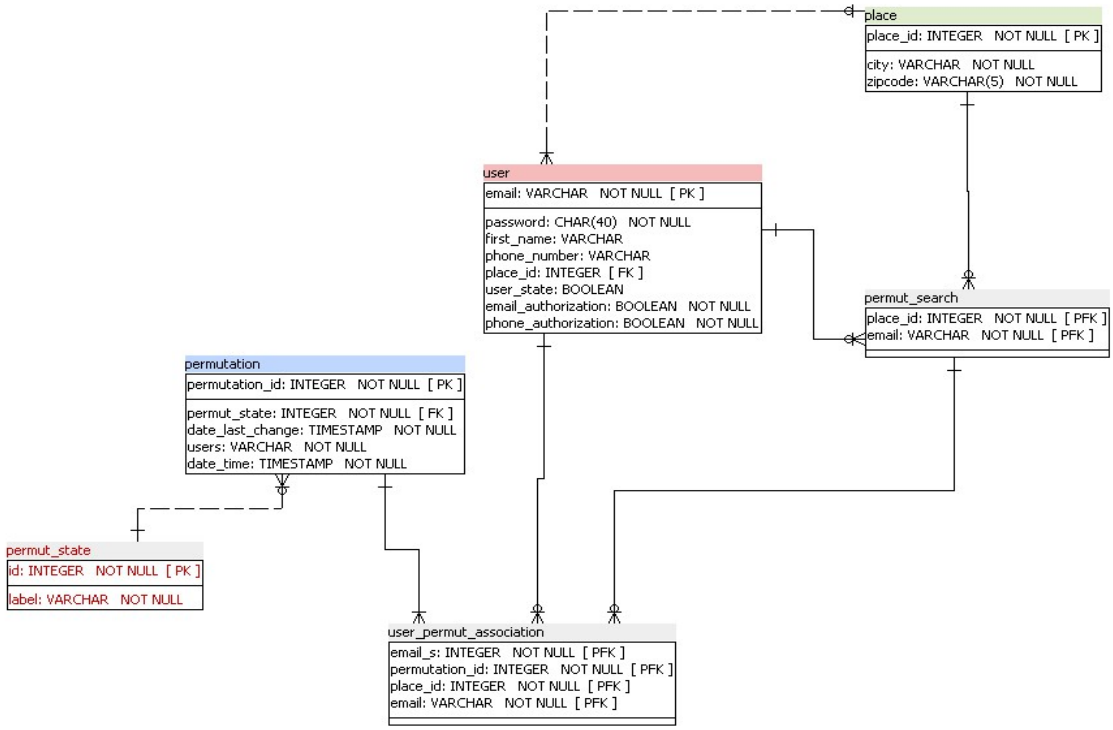
Une permutation a lieu entre au moins 2 personnes.

Lorsqu'une personne est impliquée dans une permutation, son statut passe à inactif (False) : elle ne peut pas être impliquée dans une autre permutation en parallèle.

Lorsqu'une permutation se clôt, son statut passe à approuvée ou rejetée. Si la permutation est rejetée, les utilisateurs repassent à un statut actif (True).

4 - Modèle physique de données

D'après le diagramme de classes précédemment présenté, nous avons élaboré le modèle physique de données suivant. Celui-ci modélise la base de données du site Maïeuclic, et permettra la création de la base PostgreSQL correspondante.



Modèle Physique de Données

5 - Architecture technique

5.1 - Application Web

La pile logicielle est la suivante :

- Application **Python 3.8.5 / Django 3.1.5**
- Serveur d'application **Gunicorn 20.0.4**
- Serveur web **Nginx 1.18.0**
- PostgreSQL **13.2**

Le diagramme de composants qui suit modélise l'organisation du système au niveau des éléments logiciels. Il permet de visualiser les dépendances et interfaces entre composants internes et externes.

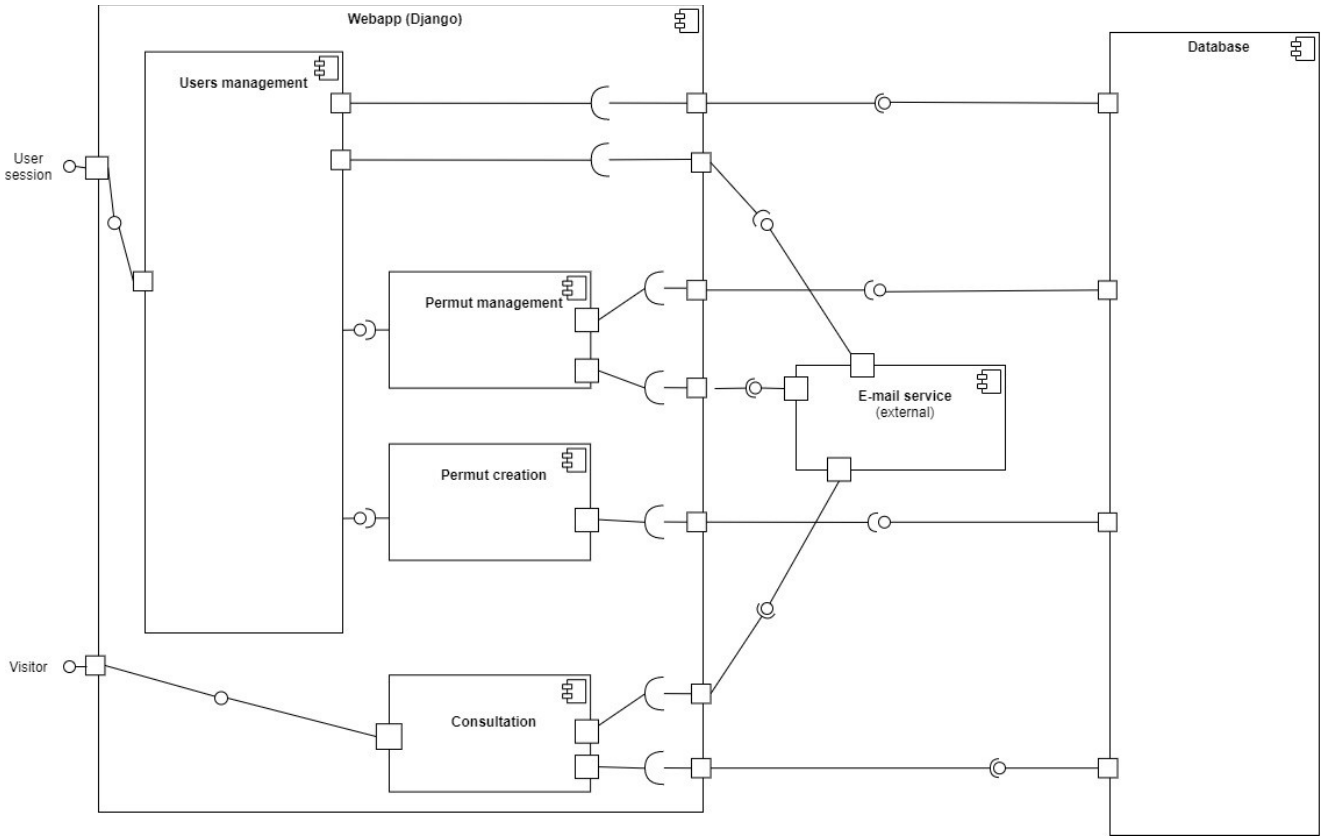


Diagramme de composants

5.1.1 - Composants

L'interface utilisateur fait appel à une webapplication Django, permettant un développement complet alliant rapidité et robustesse. Cette webapplication fait appel à une base de données PostgreSQL pour la gestion de l'information afin d'obtenir un système d'une grande stabilité, et à un service externe pour la gestion des e-mails.

6 - Architecture de Déploiement

Le diagramme de déploiement suivant décrit l'organisation physique des composants retenus.

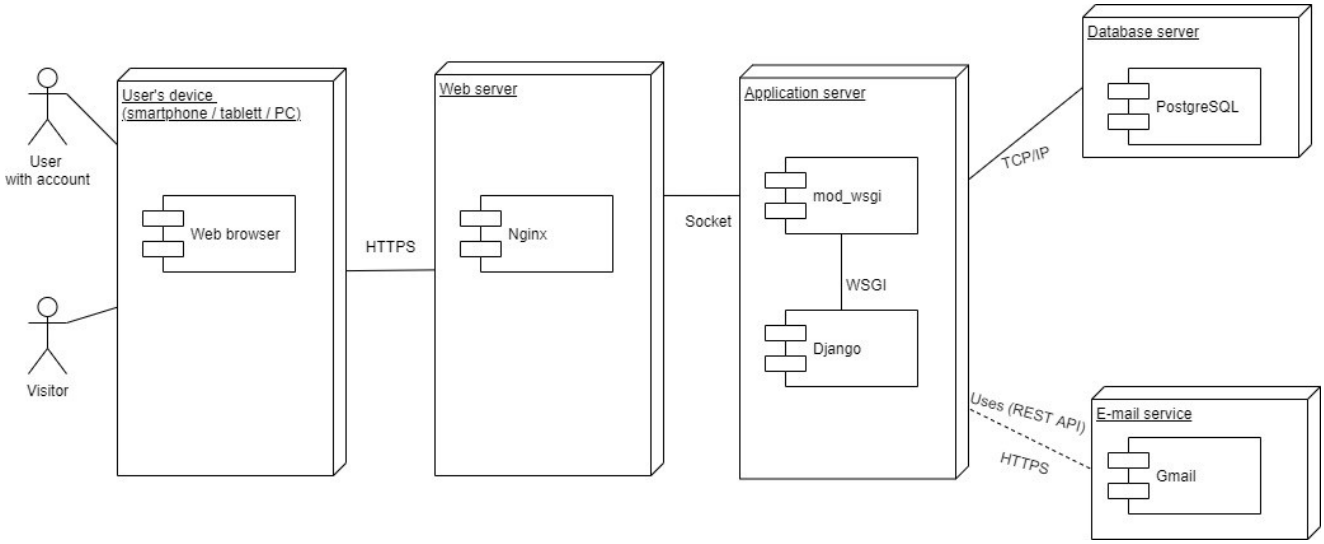


Diagramme de déploiement

Le matériel de l'utilisateur sera varié, puisqu'il peut s'agir de smatrphone, tablette ou PC.

Nous avons choisi un serveur web NginX (très performant et léger) qui utilise un serveur d'application Django (développement rapide, robuste). Celui-ci interagit avec un serveur de base de données relationnelle PostgreSQL, le service de messagerie électronique Gmail.

7 - Architecture logicielle

7.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git**, les dépendances et le packaging par **pip**.

7.1.1 - Les couches

L'application étant développée à l'aide de Django, son architecture est une architecture Model-Vue-Template.

7.1.2 - Les modules

L'application est gérée sur le serveur par Gunicorn. Les requêtes sont traitées par Gunicorn qui les transmet à Django.

7.1.3 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

– les répertoires sources sont crées de façon à respecter la philosophie Django :

```
racine
 ├─ manage.py
 ├─ user
 │   ├─ templates
 │   │     ├─ signin.html
 │   │     ├─ signup.html
 │   │     ├─ confirmation_email.html
 │   │     └─ my_account.html
 │   ├─ __init__.py
 │   ├─ admin.py
 │   ├─ apps.py
 │   ├─ forms.py
 │   ├─ models.py
 │   ├─ urls.py
 │   ├─ tokens.py
 │   └─ views.py
 ├─ permut_creation
 │   ├─ templates
 │   │     └─ permut_search.html
 │   ├─ __init__.py
 │   ├─ admin.py
 │   ├─ apps.py
 │   ├─ forms.py
 │   ├─ models.py
 │   ├─ urls.py
 │   └─ views.py
 ├─ permut_management
 │   ├─ managment
 │   │     └─ commands
 │   │           └─ permutation_manager.py
 │   ├─ templates
 │   │     ├─ notif_email.html
 │   │     └─ my_permut.html
 │   ├─ __init__.py
 │   ├─ admin.py
 │   ├─ apps.py
 │   ├─ forms.py
 │   ├─ models.py
 │   ├─ urls.py
 │   └─ views.py
 ├─ core
 │   ├─ templates
 │   │     ├─ legal_notices.html
 │   │     ├─ contact.html
 │   │     ├─ contact_email.html
 │   │     └─ home.html
 │   ├─ __init__.py
 │   ├─ admin.py
 │   ├─ apps.py
 │   ├─ urls.py
 │   └─ views.py
 ├─ tests
 │   └─ ... cf plan de test
```

```
├─ static
│   ├─ css
│   │   ├─ bootstrap.css
│   │   └─ maieuclic.css
│   ├─ img
│   │   └─ ...
│   └─ js
│       ├─ bootstrap.js
│       └─ maieuclic.js
├─ templates
│   ├─ base.html
│   └─ 404.html
└─ maieuclic_project
    ├─ settings
    │   ├─ __init__.py
    │   ├─ production.py
    │   ├─ travis_settings.py
    │   └─ test_settings.py
    ├─ __init__.py
    ├─ asgi.py
    ├─ urls.py
    └─ wsgi.py
```

8 - Points particuliers

8.1 - Gestion des logs

La gestion des logs sera effectuée via Django grâce au module logging. L'API de Sentry permettra un suivi des événements.

8.2 - Environnement de développement

L'application est éveloppée en local via le framework Django, qui a tous les outils nécessaires au développement local. Nous utilisons des navigateurs modernes existants pour visualiser le rendu.

Le code est versionné sur Git.

4 types de settings seront disponibles :

__init__.py (dans le répertoire settings) procurera les settings de développement.

test_settings.py ajoutera une surcouche pour les tests.

travis_settings.py une surcouche pour les tests effectués par travis.

production.py une surcouche pour le déploiement.

8.3 - Procédure de packaging / livraison

L'intégration est continue via une exécution des tests par Travis.

Le déploiement continu est également géré par Travis.

9 - Glossaire
