Yassine BELAAROUS, Clément VIGNON CHAUDEY,
Corentin LACOUME, Ilane RIOTTE

SAE Projet

Dossier de spécifications

Année 2025-2026

IUT'O IUT'ORLÉANS INFORMATIQUE

Université d'ORLÉANS

Architecture technique :

Schéma d'architecture

- Application web développée en Python avec le framework Flask
- Organisation en modules : séparation du code en fichiers pour les modèles (database/), les vues (views.py), les formulaires (forms.py), les templates HTML (templates/), et les fichiers statiques (static/).
- Gestion des modèles de données avec SQLAlchemy..
- Authentification et gestion de session avec Flask-Login.
- Gestion des formulaires avec Flask-WTF et WTForms.
- Développement local sur des postes de travail étudiants en Linux.
- Serveur web Flask lancé en mode développement..
- Base de données locale SQLite.
- Accès via navigateur web sur le réseau local (http://127.0.0.1:5000/).

Infrastructures

- Serveur applicatif : Python, Flask.
- Base de données : SQLite.
- Stockage des fichiers : Fichiers justificatifs stockés dans un dossier static/ ou uploads/ du projet.

Technologies utilisées

- Langages : Python pour le backend, HTML/CSS/Jinja2 pour le frontend et un peu de JavaScript pour ajouter de l'interactivité.
- Frameworks et bibliothèques : Flask, Flask-Login, Flask-WTF, WTForms, SQLAlchemy, Jinja2 et Tailwind CSS
- Tests unitaires : Pytest

Données :

MPD

```
CREATE TABLE ASSURE (
  PRIMARY KEY (id_assure),
  id_assure      VARCHAR(42) NOT NULL,
  nom            VARCHAR(42),
  prenom         VARCHAR(42),
  date_naissance VARCHAR(42),
  email          VARCHAR(42),
  mdp_assure     VARCHAR(42),
  telephone      VARCHAR(42)
);

CREATE TABLE ASSUREUR (
  PRIMARY KEY (id_assureur),
  id_assureur  VARCHAR(42) NOT NULL,
  nom          VARCHAR(42),
  prenom       VARCHAR(42),
  email        VARCHAR(42),
  telephone    VARCHAR(42),
  mot_de_passe VARCHAR(42),
```

```
  societe      VARCHAR(42),
  id_assure_1  VARCHAR(42) NOT NULL,
  id_logement  VARCHAR(42) NOT NULL,
  id_assure_2  VARCHAR(42) NOT NULL,
  date_debut   VARCHAR(42)
);

CREATE TABLE BIEN (
  PRIMARY KEY (id_bien),
  id_bien         VARCHAR(42) NOT NULL,
  nom_bien        VARCHAR(42),
  description     VARCHAR(42),
  categorie       VARCHAR(42),
  date_achat      VARCHAR(42),
  prix_achat      VARCHAR(42),
  etat            VARCHAR(42),
  valeur_actuelle VARCHAR(42)
);

CREATE TABLE IMPACTE (
  PRIMARY KEY (id_bien, id_sinistre),
```

```
  id_bien      VARCHAR(42) NOT NULL,
  id_sinistre  VARCHAR(42) NOT NULL,
  degat_estime VARCHAR(42)
);

CREATE TABLE JUSTIFICATIF (
  PRIMARY KEY (id_justificatif),
  id_justificatif   VARCHAR(42) NOT NULL,
  type_justificatif VARCHAR(42),
  chemin_fichier    VARCHAR(42),
  date_ajout        VARCHAR(42),
  id_bien           VARCHAR(42) NOT NULL
);

CREATE TABLE JUSTIFIE (
  PRIMARY KEY (id_logement, id_justificatif),
  id_logement     VARCHAR(42) NOT NULL,
  id_justificatif VARCHAR(42) NOT NULL
);

CREATE TABLE LOGEMENT (
```

```
  PRIMARY KEY (id_logement),
  id_logement   VARCHAR(42) NOT NULL,
  adresse       VARCHAR(42),
  type_logement VARCHAR(42),
  surface       VARCHAR(42),
  description   VARCHAR(42),
  id_piece      VARCHAR(42) NOT NULL
);

CREATE TABLE PIECE (
  PRIMARY KEY (id_piece),
  id_piece   VARCHAR(42) NOT NULL,
  nom_piece  VARCHAR(42),
  type_piece VARCHAR(42),
  surface    VARCHAR(42),
  etage      VARCHAR(42),
  id_bien    VARCHAR(42) NOT NULL
);

CREATE TABLE POSSEDE (
  PRIMARY KEY (id_assure, id_logement),
```

```
  id_assure   VARCHAR(42) NOT NULL,
  id_logement VARCHAR(42) NOT NULL
);

CREATE TABLE SINISTRE (
  PRIMARY KEY (id_sinistre),
  id_sinistre     VARCHAR(42) NOT NULL,
  date_sinistre   VARCHAR(42),
  type_sinistre   VARCHAR(42),
  description     VARCHAR(42),
  numero_sinistre VARCHAR(42)
);

ALTER TABLE ASSUREUR ADD FOREIGN KEY (id_assure_2) REFERENCES ASSURE (id_assure);
ALTER TABLE ASSUREUR ADD FOREIGN KEY (id_logement) REFERENCES LOGEMENT (id_logement);
ALTER TABLE ASSUREUR ADD FOREIGN KEY (id_assure_1) REFERENCES ASSURE (id_assure);

ALTER TABLE IMPACTE ADD FOREIGN KEY (id_sinistre) REFERENCES SINISTRE (id_sinistre);
ALTER TABLE IMPACTE ADD FOREIGN KEY (id_bien) REFERENCES BIEN (id_bien);
```

```
ALTER TABLE JUSTIFICATIF ADD FOREIGN KEY (id_bien) REFERENCES BIEN
(id_bien);


ALTER TABLE JUSTIFIE ADD FOREIGN KEY (id_justificatif) REFERENCES JUSTIFICATIF
(id_justificatif);

ALTER TABLE JUSTIFIE ADD FOREIGN KEY (id_logement) REFERENCES LOGEMENT
(id_logement);


ALTER TABLE LOGEMENT ADD FOREIGN KEY (id_piece) REFERENCES PIECE
(id_piece);


ALTER TABLE PIECE ADD FOREIGN KEY (id_bien) REFERENCES BIEN (id_bien);


ALTER TABLE POSSEDE ADD FOREIGN KEY (id_logement) REFERENCES LOGEMENT
(id_logement);

ALTER TABLE POSSEDE ADD FOREIGN KEY (id_assure) REFERENCES ASSURE
(id_assure);
```

Gestion des interfaces :

Protocoles

HTTP/HTTPS : Toutes les interactions entre le client et le serveur Flask se font via le protocole HTTP.

REST : L'application suit le modèle REST pour l'organisation des routes, même si l'API n'est pas exposée publiquement.

Formats d’échanges

HTML/Jinja2 : Les pages web sont générées côté serveur avec Jinja2 et envoyées au navigateur sous forme de HTML.

Formulaires web : Les données sont échangées via des formulaires HTML (méthodes POST/GET).

JSON : Si besoin d’échanges asynchrones (AJAX), le format JSON peut être utilisé (ex : pour des routes de type @app.route('/api/...') ou via jsonify dans Flask).

Sécurité des échanges

CSRF : Protection contre les attaques CSRF grâce à Flask-WTF, qui ajoute un token CSRF dans chaque formulaire.

Sessions sécurisées : Gestion de l’authentification et des sessions avec Flask-Login, qui utilise des cookies sécurisés.

Hashage des mots de passe : Les mots de passe sont stockés sous forme de hash (SHA-256).

Contrôle d’accès : Les routes sensibles sont protégées par le décorateur @login_required pour empêcher l’accès aux utilisateurs non authentifiés.

Normes & standards :

Conventions de développement

Nommage : Les classes sont en PascalCase (Logement, Piece, Bien).

Les variables et fonctions sont en snake_case (ajouter_bien, user_logements).

Les fichiers Python sont en minuscules avec underscores (views.py, forms.py).

Séparation des responsabilités : Les modèles sont dans le dossier database/.

Les vues sont dans views.py.

Les formulaires sont dans forms.py.

Les templates HTML sont dans templates/.

Les fichiers statiques sont dans static/.

Utilisation de l’ORM SQLAlchemy pour éviter les requêtes SQL brutes et garantir la portabilité.

Qualité du code

Lisibilité : Le code est commenté pour expliquer les parties complexes ou importantes.

Modularité : Le projet est organisé en modules/fichiers pour faciliter la maintenance et l'évolution.

Tests unitaires : Présence de tests dans le dossier tests/ pour vérifier le bon fonctionnement des modèles principaux.

Respect des standards HTML/CSS : Les templates utilisent Jinja2 et respectent la structure HTML5, le CSS est organisé par page/fonctionnalité.

Environnements :

Développement

Notre application actuelle est en cours de développement, elle utilise Flask 3.1.2, tous les modules à télécharger sont dans requirements.txt

Tests

Quelques tests sur la BD ont été effectués mais tout n'est pas couvert

Releases

Production

Outils de versionning et conventions :

Nous avons utilisé toutes les fonctionnalités Git qui étaient demandées notamment l'utilisation des branches. Nous avons fait une branche PageWeb, test-BD, setup-arborescence et bien sûr develop. La plupart de nos commit était dans la branche PageWeb car c'était là où il fallait faire le plus de choses.