IUT Villetaneuse
Université Sorbonne Paris Nord

Documentation technique

Réaliser par Rafael, Andon,
Adam, Hamza et Jawher

2025/2026

IUT Villetaneuse
Université Sorbonne Paris Nord

Tables des matières :

Réaliser par Rafael, Andon,
Adam, Hamza et Jawher

2025/2026

IUT Villetaneuse
Université Sorbonne Paris Nord

1. Présentation générale du projet

L'application Agenda Collaboratif est une application client–serveur permettant à plusieurs utilisateurs de gérer des agendas partagés avec des équipes et des tickets colorés. Elle a été réalisée dans le cadre de la SAE 3.02 « Développer des applications communicantes ».

Les objectifs principaux sont :

- Authentifier des utilisateurs et gérer leurs droits.
- Créer des agendas contenant des équipes.
- Associer des tickets (tâches/événements) aux équipes et aux agendas.
- Afficher les tickets sur un agenda avec drag & drop.
- Conserver l'historique des modifications des tickets.

2. Architecture générale

L'application suit une architecture client–serveur :

- Serveur : écrit en Python, il gère la logique métier, la base de données et les communications réseau.
- Client : interface graphique (web) permettant à l'utilisateur de se connecter, de gérer ses agendas, équipes et tickets.

Les échanges entre client et serveur sont transportés via des sockets TCP.

3. Technologies utilisées

- Langage : Python 3
- Framework web : Flask (micro-framework)
- ORM : SQLAlchemy (gestion des modèles et requêtes)
- Gestion des sessions & login : Flask-Login
- Formulaires : Flask-WTF / WTForms (si présent dans requirements.txt)
- Base de données : SQLite (URI définie dans database.py, facilement remplaçable par PostgreSQL/MySQL)
- Gestion des dépendances : pip avec requirements.txt

4. Modèle de données

User :

- Champs principaux : id, username, email, password_hash, created_at
- Rôles gérés via relations (propriétaire d'agendas, créateur de tickets, etc.).

Agenda :

- Champs : id, name, description, owner_id, created_at
- Relation : un owner (User), plusieurs teams et tickets.

Team :

- Champs : id, name, color, agenda_id
- Relation : appartient à un Agenda, possède plusieurs TeamMember et Ticket.

TeamMember :

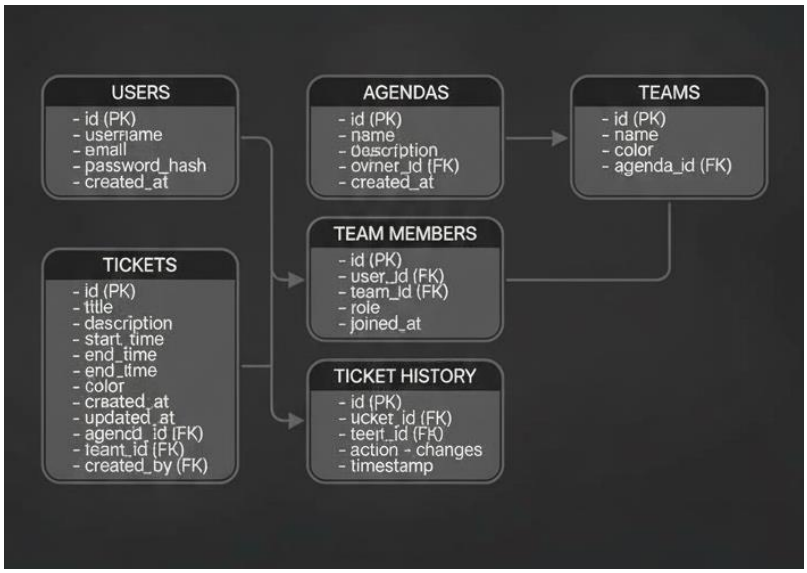
- Champs : id, user_id, team_id, role, joined_at
- Relation de liaison entre User et Team.

Ticket :

- Champs : id, title, description, start_time, end_time, color, created_at, updated_at, agenda_id, team_id, created_by
- Relation : lié à un Agenda, une Team, un User (créateur), et plusieurs TicketHistory.

TicketHistory :

- Champs : id, ticket_id, user_id, action, changes, timestamp
- Relation : trace les actions effectuées sur un Ticket par un User.



5. Organisation du code

- app.py
  - Crée l'objet Flask, charge la configuration, initialise la base (via database) et les modèles SQLAlchemy.
  - Déclare les routes principales : page d'accueil, gestion des agendas, équipes et tickets, vues du calendrier, etc.
  - Enregistre les blueprints éventuels (par ex. auth).
- auth.py
  - Contient les routes d'inscription, de connexion et de déconnexion.
  - Utilise Flask-Login pour charger l'utilisateur courant, protéger les routes avec @login_required et gérer les sessions.
- models.py
  - Déclare toutes les classes SQLAlchemy (User, Agenda, Team, TeamMember, Ticket, TicketHistory).
  - Définit les relations entre les tables (via db.relationship et db.ForeignKey).
- database.py
  - Configure l'instance SQLAlchemy (db) et l'URI de la base (par exemple sqlite:///agenda.db).
  - Fournit la fonction d'initialisation de la base (db.create_all() ou équivalent).
- requirements.txt
  - Liste les packages nécessaires à l'exécution du projet (Flask, Flask-Login, Flask-WTF, SQLAlchemy, etc.).

6. Fonctionnalités implémentées

Authentification :

- Inscription avec stockage du mot de passe haché (password_hash).
- Connexion, déconnexion, gestion de l'utilisateur courant avec Flask-Login.

Gestion des agendas :

- Création d'agenda par un utilisateur connecté (propriétaire).
- Liste et consultation des agendas appartenant à l'utilisateur.

Gestion des équipes :

- Création d'équipes associées à un agenda, avec une couleur.
- Ajout de membres à une équipe via le modèle TeamMember avec un rôle.

Gestion des tickets :

- Création, édition et suppression de tickets associés à un agenda et une équipe.
- Champs : titre, description, horaires de début/fin, couleur, créateur.
- Mise à jour de updated_at lors des modifications.

Historique des tickets :

- Insertion dans TicketHistory à chaque action importante (création/modification).
- Consultation possible de l'historique d'un ticket pour voir les changements.

7. Installation et exécution

7.1 Préparation

Créez un dossier pour le projet et placez-y tous les fichiers fournis : app.py, database.py, models.py, auth.py et requirements.txt.

Créez un environnement virtuel :

```
python -m venv venv
```

Activez-le :

- Windows :

venv\Scripts\activate

- macOS / Linux :

source venv/bin/activate

7.2 Installation des dépendances

Installez les packages nécessaires :

pip install -r requirements.txt

7.3 Lancement de l'application

Lancez l'application avec :

python app.py

7.4 Accès à l'application

Ouvrez votre navigateur et allez à :

http://localhost:5000/

8. Scénarios d'utilisation

Scénario 1 – Créer un agenda et ajouter des équipes

1. L'utilisateur se connecte à l'application.
2. Il clique sur "Nouveau Agenda" et crée un agenda nommé "Test 2" avec la description "tester".
3. Dans cet agenda, il crée une première équipe "les blacks" en cliquant sur "Créer équipe".
4. Il crée une deuxième équipe "équipe 2" dans le même agenda.
5. Chaque équipe est automatiquement associée à une couleur (affichée dans l'interface).

Scénario 2 – Créer des tickets et les associer à des équipes

1. L'utilisateur clique sur "Nouveau Ticket" pour créer une tâche.
2. Il remplit les détails du ticket :
    a. Équipe : "les blacks"
    b. Horaire : 13h à 11h30 (ou autre plage horaire)

   c. Code couleur : "cooler" (la couleur s'ajuste selon l'équipe)
   d. Titre et description (optionnels selon l'implémentation)
3. Il clique sur "Créer" pour enregistrer le ticket.
4. Le ticket apparaît dans l'agenda avec la couleur associée à l'équipe "les blacks".
5. Il peut créer d'autres tickets pour la même équipe ou une autre équipe.

Scénario 3 – Inviter des collaborateurs

1. L'utilisateur clique sur "Inviter" pour ajouter des membres à l'agenda.
2. Il entre l'adresse email d'un collaborateur (ex. "adamlut.fr" ou "hamzalut.fr").
3. Le rôle du collaborateur est défini comme "Collaborateur peut voir et commenter" (selon les rôles disponibles).
4. L'invitation est envoyée et le collaborateur peut se connecter pour voir l'agenda et ses tickets.
5. Il peut aussi inviter d'autres collaborateurs pour d'autres équipes ou pour la même équipe.

Scénario 4 – Consulter l'agenda partagé

1. Un collaborateur invité (ex. "hamza" ou "adam") se connecte à l'application.
2. Il clique sur "Ouvrir l'agenda" (ex. "Test 2").
3. Il voit tous les tickets créés par le propriétaire, organisés par équipe avec les bonnes couleurs.
4. Selon son rôle, il peut voir les tickets et potentiellement les commenter ou les modifier.
5. Les modifications effectuées (création, édition, suppression) sont enregistrées dans l'historique.