CodeIguanas

Cahier des charges

Application HomeSkolar

---

1. Spécifications fonctionnelles

Le site web **HomeSkolar** permettra aux utilisateurs, élèves et tuteurs, de réaliser plusieurs tâches. Ces fonctionnalités sont divisées en quatre grandes catégories :

1.1. Authentification

- Inscription et connexion pour les élèves et les tuteurs.
- Gestion des mots de passe : réinitialisation, modification.
- Gestion des données personnelles : modification des informations du compte.

1.2. Communication

- Messagerie entre les élèves et les tuteurs :
    - Envoi et réception de messages simples.
    - Notifications pour messages non lus.
    - Possibilité d'épingler des messages pour un accès rapide.

1.3. Gestion des rencontres (Calendrier)

- Affichage des rendez-vous à venir et passés pour les utilisateurs.
- Fonctionnalité de planification des rendez-vous :
    - Propositions de créneaux horaires.
    - Notifications pour les rendez-vous planifiés ou modifiés.

1.4. Suivi des tâches

- Création de listes de tâches :
  - Par les tuteurs, assignées aux élèves.
  - Par les élèves, pour leur propre gestion personnelle.
- Notifications pour les nouvelles tâches ajoutées ou modifiées.
- Suivi des tâches complétées/non complétées.

---

2. Spécifications techniques

Composant	Technologie choisie	Justification principale
Front-End	React.js	Interface utilisateur dynamique et modulaire.
Back-End	Django (Python)	Sécurisé, complet, adapté à la gestion relationnelle.
Base de Données	PostgreSQL	Puissance, flexibilité, gestion relationnelle.

---

3. Veille technologique

3.1. Front-End : React.js

- **Rôle** : Développement des interfaces utilisateur dynamiques et réactives.
- **Pourquoi React.js ?**
  - Permet de développer des **interfaces dynamiques et réactives** avec une architecture modulaire.
  - Utilisation de **composants réutilisables**, améliorant la productivité et réduisant le temps de développement.
  - Excellente intégration avec des outils tiers, notamment des bibliothèques pour gérer les calendriers (**FullCalendar**, par exemple).

Comparaison avec d'autres technologies :

Technologie	Points forts	Limites
React.js	Modulaire, documentation abondante, vaste communauté.	Nécessite l'ajout d'autres bibliothèques pour des fonctionnalités spécifiques.
Vue.js	Courbe d'apprentissage douce, simple à intégrer.	Moins adapté pour des projets complexes.
Angular	Framework complet et structuré.	Courbe d'apprentissage plus longue et syntaxe complexe.

Sources :

- Documentation officielle de React.js
- Article : React.js vs Vue.js vs Angular (Kinsta)

3.2. Back-End : Django (Python)

- **Rôle** : Gestion des fonctionnalités métiers, logique serveur et interactions avec la base de données.
- **Pourquoi Django ?**
    - Framework robuste et sécurisé pour la gestion d'applications web nécessitant une **authentification utilisateur** et des interactions complexes avec une base de données relationnelle.
    - Outils intégrés pour accélérer le développement : système de gestion des utilisateurs, ORM (**Object Relational Mapping**), API REST via **Django REST Framework (DRF)**.
    - Parfaitement adapté à l'intégration d'un calendrier et à la gestion des tâches.

Comparaison avec d'autres technologies :

Technologie	Points forts	Limites
Django	Sécurisé, rapide à développer grâce aux fonctionnalités intégrées.	Moins adapté pour des applications nécessitant beaucoup de temps réel.
Flask	Framework minimaliste et léger.	Nécessite des extensions pour des fonctionnalités avancées.
Node.js	Très performant pour les applications asynchrones.	Demande plus de configuration pour des applications relationnelles complexes.

Sources :

- Documentation officielle de Django
- Article : Django vs Flask (GeeksforGeeks)

3.3. Base de Données : PostgreSQL

- **Rôle** : Stockage des données des utilisateurs, messages, rendez-vous et tâches.
- **Pourquoi PostgreSQL ?**
  - Base de données **relationnelle puissante**, parfaite pour gérer les relations complexes entre utilisateurs, tâches et événements du calendrier.
  - Supporte des types de données avancés, comme le **JSON**, qui permettent une certaine flexibilité pour des structures non relationnelles.
  - Stabilité éprouvée dans des environnements de production critiques.

Comparaison avec d'autres technologies :

Technologie	Points forts	Limites

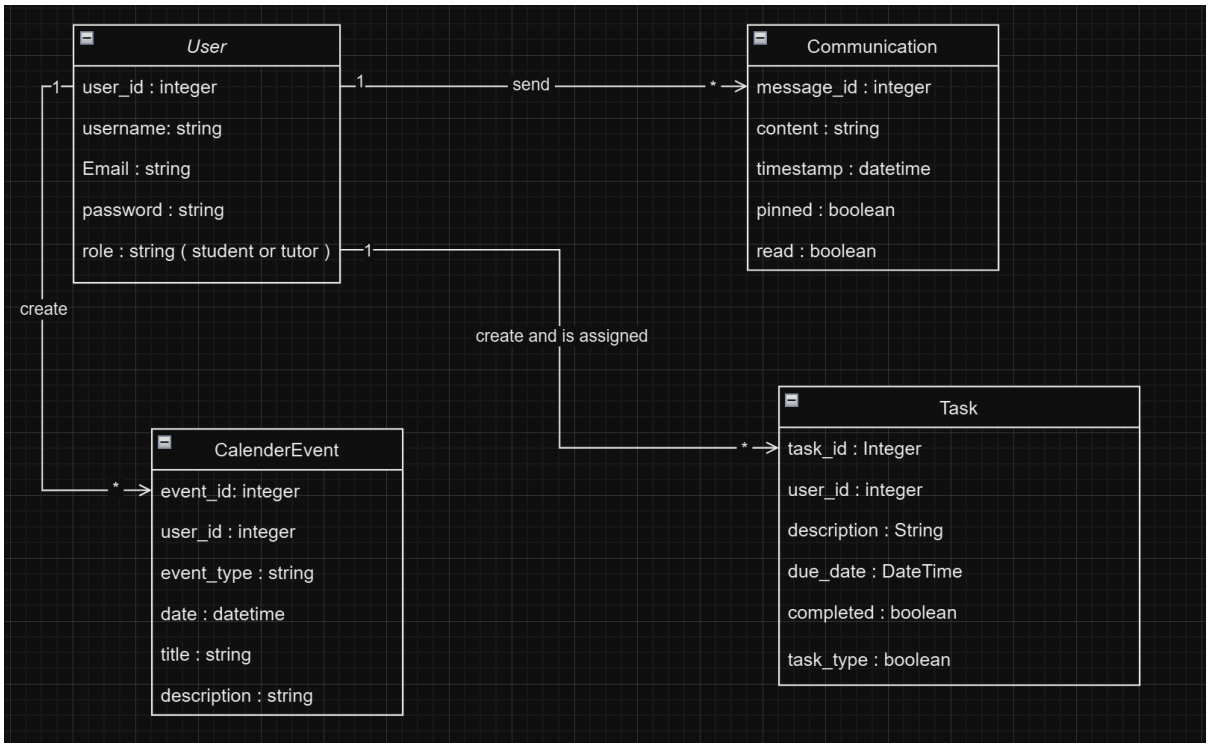
PostgreSQL	Relationnel, puissant, support JSON/JSONB.	Légèrement plus complexe à configurer que MySQL.
MySQL	Simple à utiliser, largement adopté.	Moins performant pour des relations complexes.
MongoDB	NoSQL, parfait pour des données non structurées.	Pas idéal pour les relations complexes.

Sources :

- Documentation officielle de PostgreSQL
- Comparaison PostgreSQL vs MySQL (DigitalOcean)

---

4. Diagramme de classes UML



5. Organisation

Tableau d'estimation des délais :

Sprint	Durée estimée	Tâches principales	Contributions
Sprint 1	4 jours	Authentification, tableau de bord	Interfaces utilisateur et API : Inscription, connexion, réinitialisation, tableau de bord.
Sprint 2	5 jours	Messagerie, calendrier	UI et API pour messagerie (envoi, notifications), planification de rendez-vous, affichage calendrier.
Sprint 3	5 jours	Gestion des tâches et tests finaux	Création de tâches, notifications associées, tests unitaires, corrections et ajustements.
Total	**14 jours**		