Soutenance

Home Skolar

Présentation:

HomeSkolar est une association qui met en relation des enfants en difficulté scolaire et des bénévoles à distance.

Elle a pour objectif de permettre à tout élève, où qu'il soit, d'accéder à un soutien scolaire.

Chaque élève inscrit sur le site se verra assigner un tuteur bénévole.

Cahier De charges

- Authentification
  - Les élèves et tuteurs doivent avoir la possibilité de s’inscrire, se connecter et gérer leur mots de passe et autres données personnels
- Communication
  - Les élèves doivent être en mesure de communiquer avec leur tuteurs et d'épingler des messages
- Rencontres
  - Les élèves doivent pouvoir prendre rendez-vous avec leur tuteurs grâce à un calendrier commun
- Tâches
  - les élèves vont être notifié au sein de l'application d’une liste de tâches à réaliser pour la prochaine rencontre.

Spécifications fonctionnelles

1- Objectif du système

HomeSkolar est une plateforme web qui permet de mettre en relation un élève en difficulté scolaire et des tuteurs bénévole.

L'objectif principal est de permettre à chaque élève, où qu'il soit, d'accéder à un soutien scolaire adapté à ses besoins.

Spécifications fonctionnelles

2- Acteurs

- Élève
  - S’inscrit sur la plateforme, consulte les disponibilités, demande RDV, accède aux séances, consulte le suivi, échange avec le tuteur via messagerie.
- Tuteur
  - S’inscrit, indique ses matières et compétences, définit ses disponibilités, accepte/refuse les RDV, remplit le suivi des séances, échange avec l’élève via messagerie.
- Administrateur
  - Gère l’inscription et les comptes des utilisateurs, assigne élèves ↔ tuteurs, consulte statistiques et rapports.

Spécifications fonctionnelles

3- Cas d’usage fonctionnels

- Gestion des utilisateurs
  - Inscription / connexion
  - Gestion du profil
- Gestion des séances
  - Assignation tuteur ↔ élève
  - Gestion des disponibilités
  - Confirmation des RDV
- Suivi pédagogique
  - Accès aux séances (visioconférence)
  - Remplissage du suivi
  - Consultation des suivis

Spécifications fonctionnelles

3- Cas d'usage fonctionnels

- Messagerie interne
  - Communication sécurisée entre binôme élève/tuteur
  - Restrictions : seuls les utilisateurs assignés peuvent échanger des messages
- Notifications et alertes
  - Demande ou confirmation de RDV
  - Réception d'un message
  - Modifications de planning ou annulations de séance
- Administration
  - Gestion des comptes utilisateurs (création, suppression, modification)
  - Assignation des élèves aux tuteurs
  - Consultation de rapports et statistiques : nombre de séances, progression des élèves, taux de disponibilité des tuteurs

Spécifications fonctionnelles

4- Règles métier

- Un élève ne peut être assigné qu'à un seul tuteur à la fois.
- Un tuteur peut gérer plusieurs élèves selon sa disponibilité.
- Les séances ne peuvent être réservées que sur des créneaux disponibles.
- Seuls les binômes assignés peuvent accéder à la messagerie et aux séances
- Les suivis de séance sont visibles uniquement par le tuteur concerné, l'élève et l'administrateur

Spécifications fonctionnelles

5- Critères de qualité

- **Sécurité**: authentification et chiffrement des mots de passe
- **Accessibilité**: interface responsive, compatible desktop et mobile
- **Fiabilité**: sauvegarde régulière des données et notifications fiables.
- **Performance**: accès aux données et visioconférence sans latence excessive

Veille Technologique

- Back-end
  - Spring
- Front-end
  - Angular
  - React
- Base(s) de Donnée(s)
  - PostgreSQL
  - MongoDB (optionnel)
- Gestion temps réel et messagerie
  - WebSocket avec Spring
  - Firebase Cloud Messaging

Veille Technologique

- Architecture recommandée
  - Backend : Spring Boot + Spring Security + Spring Data JPA
  - Base de données principale: PostgreSQL
  - Base optionnelle pour messages : MongoDB
  - WebSocket pour messagerie temps réel
  - Intégration Jitsi / WebRTC pour visioconférences
  - JWT + HTTPS + RBAC pour sécurité
- Avantages
  - Code robuste et maintenable
  - Communauté Java importante
  - Compatible microservices pour scalabilité future
  - Bonne intégration avec front-end React ou Vue

Veille Technologique

- Back-end
  - Spring
- Front-end
  - Angular
  - React
- Base(s) de Donnée(s)
  - PostgreSQL
  - MongoDB (optionnel)
- Gestion temps réel et messagerie
  - WebSocket avec Spring
  - Firebase Cloud Messaging

Spécifications Techniques

1- Frameworks Java pour le backend

Spring Boot

- Framework mature, très utilisé pour les applications web et microservices. Supporte REST, sécurité, base de données.
- Vs Jakarta EE : Spring Boot plus simple à configurer, plus riche en communauté et documentation.

Source: https://spring.io/projects/spring-boot

Spécifications Techniques

1- Frameworks Java pour le backend

Spring Security

- Gestion complète de l'authentification et des autorisations, support JWT, OAuth2.
- Vs Apache Shiro : Spring Security mieux intégré à Spring Boot et plus utilisé en entreprise.

Source: https://spring.io/projects/spring-security

Spécifications Techniques

1- Frameworks Java pour le backend

Spring Data JPA

- Simplifie l’accès aux bases de données relationnelles (PostgreSQL/MySQL).

- Simplifie l’accès aux bases de données relationnelles (PostgreSQL/MySQL)

  Source: https://spring.io/projects/spring-data-jpa

Spécifications Techniques

2- Base de données

PostgreSQL

- Base relationnelle robuste, ACID compliant, idéale pour gérer comptes, RDV, suivis.

- Vs MySQL : PostgreSQL plus riche en fonctionnalités (JSONB, transactions complexes).

Source: https://www.postgresql.org/

Spécifications Techniques

2- Base de données

MongoDB

- Pour stocker l'historique des messages et logs de sessions si besoin de flexibilité.

- Vs PostgreSQL : MongoDB plus flexible pour documents JSON, PostgreSQL plus structuré.

Source: https://www.mongodb.com/

Spécifications Techniques

3- Gestion temps réel et messagerie

WebSocket avec Spring

- Supporte messagerie temps réel pour chat élève ↔ tuteur et notifications instantanées.

- Supporte messagerie temps réel pour chat élève ↔ tuteur et notifications instantanées

Source: https://spring.io/guides/gs/messaging-stomp-websocket

Spécifications Techniques

3- Gestion temps réel et messagerie

Firebase Cloud Messaging

- Notifications push multi-plateformes (optionnel pour mobile).

- Vs WebSocket : FCM gère push hors app, WebSocket pour temps réel in-app.

Source: https://firebase.google.com/docs/cloud-messaging

Spécifications Techniques

4- Visioconférence

Jitsi Meet

- Open source, intégrable via iframe ou API REST/SDK, pas de dépendance à un service payant.

- Vs Zoom SDK : Jitsi gratuit, respect vie privée, Zoom plus fiable mais payant.

Source: https://jitsi.org/

Diagramme de classes

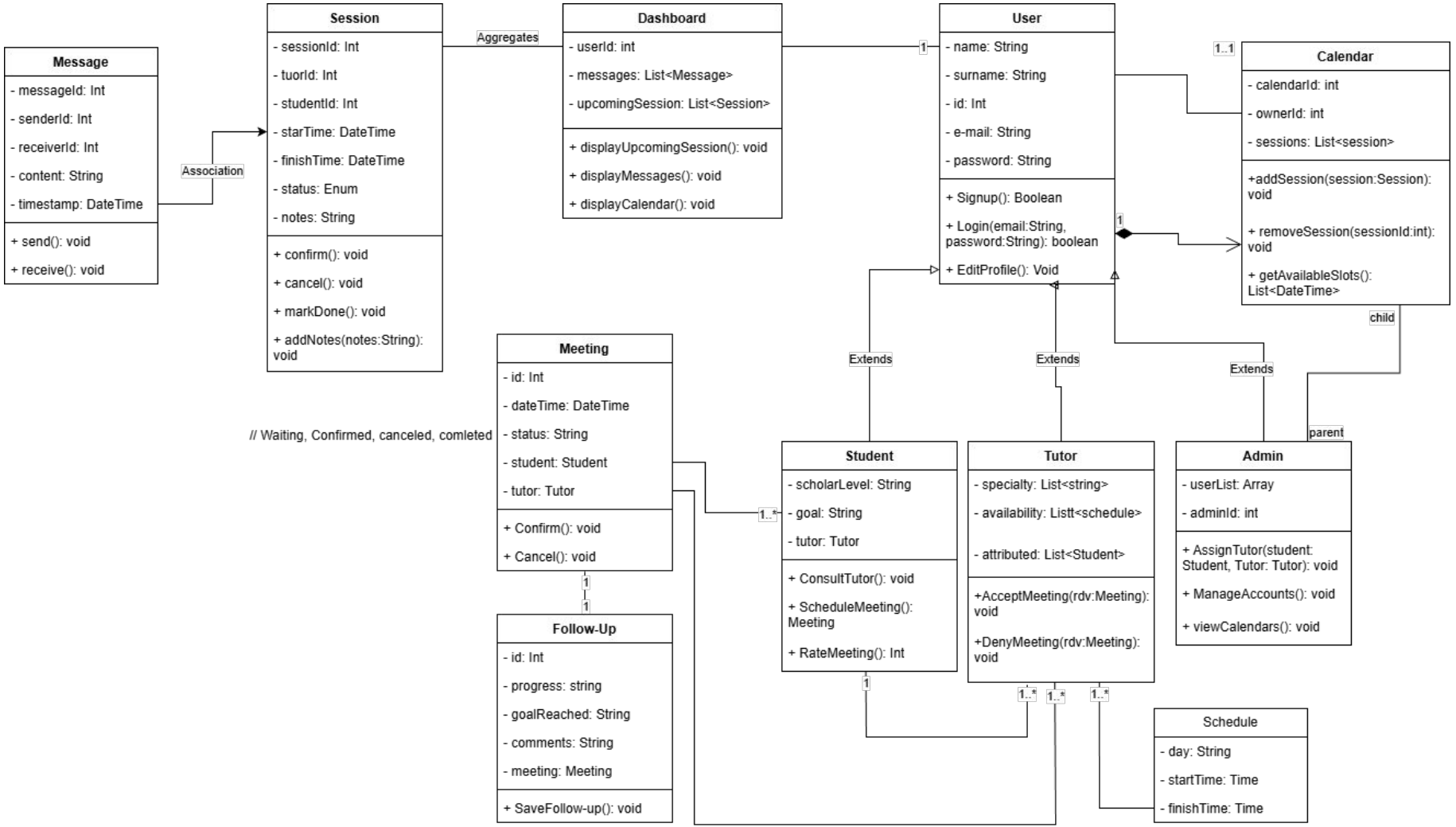
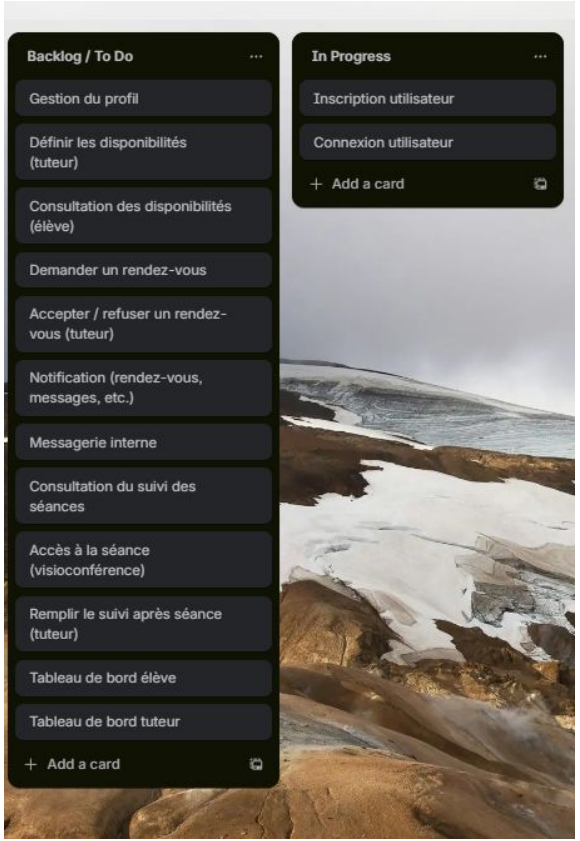

Message
- messageId: Int
- senderId: Int
- receiverId: Int
- content: String
- timestamp: DateTime
+ send(): void
+ receive(): void
Association
Session
- sessionId: Int
- tuorId: Int
- studentId: Int
- starTime: DateTime
- finishTime: DateTime
- status: Enum
- notes: String
+ confirm(): void
+ cancel(): void
+ markDone(): void
+ addNotes(notes:String): void
Aggregates
Dashboard
- userId: int
- messages: List<Message>
- upcomingSession: List<Session>
+ displayUpcomingSession(): void
+ displayMessages(): void
+ displayCalendar(): void
1
User
- name: String
- surname: String
- id: Int
- e-mail: String
- password: String
+ Signup(): Boolean
+ Login(email:String, password:String): boolean
+ EditProfile(): Void
1..1
Calendar
- calendarId: int
- ownerId: int
- sessions: List<session>
+addSession(session:Session): void
+ removeSession(sessionId:int): void
+ getAvailableSlots(): List<DateTime>
child
parent
Extends
Extends
Extends
// Waiting, Confirmed, canceled, comleted
Meeting
- id: Int
- dateTime: DateTime
- status: String
- student: Student
- tutor: Tutor
+ Confirm(): void
+ Cancel(): void
1..*
1
1
Follow-Up
- id: Int
- progress: string
- goalReached: String
- comments: String
- meeting: Meeting
+ SaveFollow-up(): void
Student
- scholarLevel: String
- goal: String
- tutor: Tutor
+ ConsultTutor(): void
+ ScheduleMeeting(): Meeting
+ RateMeeting(): Int
1
Tutor
- specialty: List<string>
- availability: Listt<schedule>
- attributed: List<Student>
+AcceptMeeting(rdv:Meeting): void
+DenyMeeting(rdv:Meeting): void
1..*
1..*
1..*
Admin
- userList: Array
- adminId: int
+ AssignTutor(student: Student, Tutor: Tutor): void
+ ManageAccounts(): void
+ viewCalendars(): void
Schedule
- day: String
- startTime: Time
- finishTime: Time

Tableau kanban

**Vide a titre d'exemple*



Merci

Fin de la présentation

Cordialement