Cahier des Charges - Site de Soutien Scolaire

Le présent cahier des charges vise à définir les spécifications fonctionnelles et techniques d'un site web de soutien scolaire, destiné à offrir aux élèves un environnement d'apprentissage interactif et adaptatif.

Nous allons détailler les 4 fonctionnalités principales qui guidera et centralisera notre projet qui sont :

- L'authentification – La communication -Les rencontres élève/tuteur -Les tâches

Puis nous proposerons la Veuille technologique sur les technologies web dans le contexte du développement d'un site de soutien scolaire

Nous listerons les spécifications techniques utilisées en se concentrant sur les parties front end et back end, ainsi que les technologies utilisées et leur réponse aux besoins du client

Et enfin nous allons donner le Diagramme de Classe UML ainsi que les attributs et les méthodes ainsi que les cardinalités entre les entités

1. Spécifications Fonctionnelles

1.1 Authentification :

- **Fonctionnalités :**

1. Inscription :

- Les utilisateurs peuvent s'inscrire en fournissant une adresse e-mail et un mot de passe.

- Les données personnelles telles que le nom, le prénom, etc., sont collectées lors de l'inscription.

2. Connexion

- Les utilisateurs peuvent se connecter en utilisant leur adresse e-mail et leur mot de passe.

3. Gestion du Mot de Passe :

- Possibilité de réinitialiser le mot de passe via un lien envoyé par e-mail.

4. Gestion des Données Personnelles :

- Les utilisateurs peuvent gérer et mettre à jour leurs données personnelles après la connexion.

1.2 Communication :

- **Fonctionnalités :**

1. Envoi de Messages Texte :

- Les utilisateurs peuvent envoyer des messages texte à d'autres élèves ou tuteurs et vice versa via une messagerie integrée

2. Épinglage de Messages :

- Les utilisateurs peuvent épingler des messages pour les retrouver facilement.

3. Notifications pour les Nouveaux Messages :

- Les utilisateurs reçoivent des notifications pour tout message non lu.

1.3 Rencontres Élève/Tuteur :

- **Fonctionnalités :**

1. Demande de soutien :

- Les élèves peuvent soumettre des demandes de soutien pour des matières spécifiques, en précisant le niveau et le créneau horaire souhaité.

2. Acceptation des demandes :

- Les tuteurs peuvent accepter les demandes de soutien qui correspondent à leurs compétences et disponibilités

3. Planification des Rencontres :

- Une fois qu'une demande est acceptée, Les élèves et les tuteurs peuvent planifier des rencontres en sélectionnant une date et un horaire.

4. Modification des Rencontres :

- Possibilité de modifier les détails d'une rencontre préalablement planifiée.

5. Annulation des Rencontres :

- Les utilisateurs peuvent annuler des rencontres préalablement planifiées.

6. Affichage des Rencontres à Venir :

- Un tableau de bord affiche les prochaines rencontres planifiées.

1.4 Tâches :

- **Fonctionnalités :**

1. Création de Listes de Tâches :

- Les utilisateurs peuvent créer des tâches liées à des sujets spécifiques ou à des séances de soutien

2. Attribution de Tâches :

- Les tuteurs peuvent attribuer des tâches spécifiques aux élèves, telles que des exercices à faire avant la prochaine séance de soutien.

3. Notifications pour les Tâches à Venir :

- Les utilisateurs reçoivent des notifications pour les tâches à venir.

4. Marquage des Tâches Réalisées :

- Les utilisateurs peuvent marquer les tâches comme réalisées.

1.5 Calendrier

1.Planification :

- Les utilisateurs peuvent visualiser leur emploi du temps personnel ainsi que les séances de soutien planifiées.

2.Gestion des disponibilités :

- Les tuteurs peuvent mettre à jour leur disponibilité sur le calendrier pour que les élèves puissent voir quand ils sont disponibles pour des séances de soutien.

Ces spécifications fonctionnelles fournissent une base solide pour le développement de notre site web de soutien scolaire, en couvrant les principales fonctionnalités liées à l'authentification, à la communication, aux rencontres élève/tuteur et à la gestion des tâches, tout en intégrant un tableau de bord et un calendrier pour une meilleure expérience utilisateur

2. Veuille Technologique

Les propositions sur la veille Technologique sur les technologies web dans le contexte du développement d'un site de soutien scolaire, si on veut optimiser par le langage python

1. Nouvelles versions de Python

- Suivi des annonces de nouvelles versions de Python, ainsi que des fonctionnalités et des améliorations introduites dans chaque version, pour s'assurer que le site utilise les dernières fonctionnalités du langage et bénéficie des améliorations de performances et de sécurité.

2. Frameworks Python

- Exploration des frameworks Python populaires tels que Django et Flask pour le développement backend, ainsi que des frameworks frontend comme Dash et Bokeh pour la visualisation de données et les interfaces utilisateur interactives.

3. Librairies Python

- Veille sur les nouvelles librairies Python pour des tâches spécifiques telles que le traitement du langage naturel (NLTK, spaCy), l'apprentissage automatique (scikit-learn, TensorFlow, PyTorch), la manipulation de données (Pandas, NumPy) et la visualisation (Matplotlib, Seaborn).

4. Outils de développement

- Recherche et évaluation des outils de développement Python tels que PyCharm, Jupyter Notebook, et VS Code, ainsi que des extensions et des plugins utiles pour améliorer la productivité et la qualité du code.

5. Performance et Optimisation

- Suivi des bonnes pratiques et des techniques d'optimisation de la performance en Python, notamment l'utilisation de structures de données efficaces, avec les mises à jour de traitement de données Python, telles que Pandas et Numpy , et l'optimisation des requêtes de base de données relationnelles PostgreSQl, MySQL et les bases de données NoSQL(MongoDB, Cassandra) pour garantir des performances rapides et évolutives du site.

6. Sécurité Python

- Veille sur les meilleures pratiques de sécurité en Python, y compris la prévention des injections SQL, la validation des entrées utilisateur, la gestion sécurisée des sessions et des cookies, et l'utilisation de bibliothèques de sécurité telles que cryptographie pour le chiffrement des données.

7. Communauté Python

- Participation à des forums de discussion, des groupes de discussion et des événements communautaires Python tels que Stack Overflow, Reddit(r/Python) et python.org forum pour rester connecté à la communauté des développeurs Python, échanger des idées, partager des connaissances et bénéficier des retours d'expérience d'autres développeurs.

- Assister à des conférences et des évènements Python tels que PyCon, EuroPython et PyData pour écouter des présentations techniques, participer à des ateliers techniques

8. Intelligence Artificielle et Adaptabilité :

- Exploration des applications de l'intelligence artificielle dans l'éducation, telles que les systèmes de recommandation adaptative et les modèles d'apprentissage automatique pour personnaliser le contenu pédagogique en fonction des besoins individuels des élèves.

- Veille sur les outils et les bibliothèques d'apprentissage automatique et de traitement du langage naturel (NLP) pour enrichir les fonctionnalités du site.

En réalisant une veille technologique sur ces différents aspects de l'optimisation via le langage Python, on peut garantir que le site de soutien scolaire tire pleinement parti des capacités et des avantages de Python en termes de performance, de sécurité, et de productivité de développement.

3. Spécifications Techniques

Voici la liste des spécifications techniques en se concentrant sur les parties front end et back end, ainsi que les technologies utilisées et leur réponse aux besoins du client :

1. Authentification

- Frontend :

Pour l'interface utilisateur de l'authentification, vous pouvez utiliser des outils comme HTML5, CSS3 pour la structure et la mise en forme, et JavaScript pour les interactions utilisateur. Pour une expérience utilisateur plus fluide, vous pouvez également envisager l'utilisation de frameworks JavaScript tels que React.js ou Vue.js pour la création de composants réactifs et dynamiques..

- Backend :

Utilisation d'un framework côté serveur comme Django (Python) ou Ruby on Rails (Ruby) pour gérer l'authentification et la sécurité des comptes utilisateur.

- Réponse aux besoins du client :

Les frameworks backend offrent des fonctionnalités intégrées pour gérer les sessions utilisateur, les cookies sécurisés, et les mécanismes de hachage de mot de passe, garantissant ainsi une authentification sécurisée et fiable.

2. Communication

- Frontend :

Pour la messagerie en temps réel, vous pouvez utiliser des *bibliothèques JavaScript* comme *Socket.IO* pour la gestion des *WebSockets.*

Ces bibliothèques facilitent la communication bidirectionnelle entre le navigateur des utilisateurs et le serveur, permettant ainsi

une messagerie instantanée et réactive.

- Backend :

Utilisation de WebSocket avec des bibliothèques telles que Socket.IO (JavaScript) ou *Django Channels* (Python) pour permettre une communication bidirectionnelle en temps réel entre les utilisateurs.

- **Réponse aux besoins du client :**

L'utilisation de WebSocket assure une communication instantanée et réactive entre les utilisateurs, améliorant ainsi l'expérience de messagerie et favorisant une interaction fluide.

3. Rencontres Élève/Tuteur

- Frontend :

Pour afficher les demandes de soutien et les horaires des séances, vous pouvez utiliser *HTML, CSS* et *JavaScript p*our la création de l'interface utilisateur. Pour rendre ces informations interactives et dynamiques, vous pouvez utiliser des frameworks frontend comme *React.js ou Vue.js,* éventuellement en combinaison avec des bibliothèques de gestion d'état telles que *Redux* (pour React) ou *Vuex* (pour Vue).

- Backend :

Utilisation de bases de données relationnelles comme *PostgreSQL ou MySQL* pour stocker et gérer les données liées aux rencontres élève/tuteur.

-Réponse aux besoins du client :

Les bases de données relationnelles offrent une structure de données robuste pour stocker et récupérer les demandes de soutien et les horaires des séances, assurant ainsi une gestion efficace des rencontres.

4. Gestion des tâches :

- Frontend :

Pour l'interface utilisateur de gestion des tâches, vous pouvez utiliser des technologies similaires à celles mentionnées pour l'authentification et la communication, telles que HTML, CSS, JavaScript, et des frameworks frontend comme *React.js ou Vue.js* pour créer une interface utilisateur interactive et conviviale pour la création, l'affichage et la mise à jour des tâches.

- Backend :

Utilisation d'une *API RESTful* avec des frameworks comme *Django REST Framework* (Python) ou Ruby on Rails API (Ruby) pour gérer les opérations *CRUD* (Create, Read, Update, Delete) des tâches.

- Réponse aux besoins du client :

L'utilisation d'une *API RESTful* facilite l'intégration avec d'autres fonctionnalités du site web et offre une expérience utilisateur cohérente pour la gestion des tâches.

5. Calendrier :

- Frontend :

Pour l'intégration du calendrier, vous pouvez utiliser des bibliothèques JavaScript spécialisées dans la gestion des calendriers, telles que *FullCalendar.js* ou *react-big-calendar* pour React.js. Ces bibliothèques offrent des fonctionnalités avancées pour l'affichage et la gestion des événements, ce qui permet une intégration facile du calendrier dans l'interface utilisateur frontend.

- Backend :

Gestion des données du calendrier côté serveur pour stocker les horaires des séances et gérer les disponibilités des tuteurs.

- Réponse aux besoins du client :

L'utilisation de bibliothèques de calendrier offre une visualisation intuitive des horaires des séances, facilitant ainsi la planification et la gestion des rendez-vous entre élèves et tuteurs.

En adoptant ces spécifications fonctionnelles et techniques, le site de soutien scolaire offrira une plateforme interactive et efficace pour la communication entre élèves et tuteurs, la planification des rencontres, la gestion des tâches et bien plus encore. Les technologies sélectionnées garantissent une expérience utilisateur fluide et une mise en œuvre technique solide.

4- DIAGRAMME DE CLASSE UML

Diagramme de classe UML pour le soutien scolaire entre élève et tuteur bénévole :

le diagramme de classes UML tenant compte des 4 fonctionnalités : **l'Authentification Elève/Tuteur**, de la **rencontre Elève/Tuteur**, de la **Communication Elève/Tuteur,** et des **Tâches entre Elève/Tuteur** , ainsi que les attributs et méthodes assignés à chaque classe sont établis ci-dessous

Explication du diagramme , :

*1. **Authentification :***

Class Utilisateur :

- Attributs : idUtilisateur, nomUtilisateur, email, motDePasse, role(elève\tuteur)
- Méthodes :s'inscrire(), se_connecter(), se_deconnecter(),modifierMotDePasse(), modifierProfil(),

Class Session :

- Attributs : idSession, idUtilisateur, dateConnexion, statut
- Méthodes : ouvrirSession(), fermerSession()

*2. **Communication :***

- **Class Message :**
  - Attributs : idMessage, idExpediteur, idDestinataire, contenu, dateEnvoi,
  - Méthodes : envoyerMessage(), recevoirMessage(), epinglerMessage(), notificationMessage()

*3. **Rencontres Élève/Tuteur :***

- **Class Événement :**
  - Attributs : idEvenement, idUtilisateur, dateEvenement, description
  - Méthodes : planifierEvenement(), modifierÉvénement(), supprimerÉvénement(), affichageEvenement()

- **Class Calendrier :**

  - Attributs : idCalendrier, idUtilisateur, listEvenements

  - Méthodes : afficherCalendrier(), ajouterÉvénement()

*4. **Tâches :***

Class Tâches :

  - Attributs : idTâche, idUtilisateur, description, dateCreation, dateÉchéance, statut

  - Méthodes : créerTâche(), modifierTâche(), supprimerTâche()

Ce diagramme de classe UML représente les entités principales et leurs relations pour le soutien scolaire entre élève et tuteur bénévole. Les attributs et les méthodes spécifiques à chaque classe sont définis en fonction des fonctionnalités requises, offrant une base solide pour le développement de l'application. Vous pouvez adapter ces classes en fonction de vos besoins spécifiques.

Description détaillée de chaque Entité créée :

Classe Utilisateur :

La classe Utilisateur représente un utilisateur de la plateforme de soutien scolaire. Elle contient les informations générales sur un utilisateur et gère les actions liées à son compte.

Attributs :

- idUtilisateur : Identifiant unique de l'utilisateur.

- **nomUtilisateur :** Nom de l'utilisateur.

- **email :** Adresse email de l'utilisateur.

- **motDePasse :** Mot de passe de l'utilisateur.

- **role :** Type de compte de l'utilisateur (élève ou tuteur).

Méthodes :

- **inscrire() :** Méthode permettant à un utilisateur de s'inscrire sur la plateforme.

- **connecter() :** Méthode permettant à un utilisateur de se connecter à son compte.

- **modifierMotDePasse() :** Méthode permettant à un utilisateur de modifier son mot de passe.

- **modifierDonnéesPersonnelles() :** Méthode permettant à un utilisateur de modifier ses données personnelles.

Classe Session :

La classe Session gère les sessions des utilisateurs sur la plateforme. Elle permet de suivre l'état de connexion des utilisateurs et de maintenir leur session active.

Attributs :

- **idSession :** Identifiant unique de la session.

- **idUtilisateur :** Identifiant de l'utilisateur associé à la session.

- **dateConnexion :** Date et heure de connexion de l'utilisateur.

- **statut :** Statut de la session (ouvert ou fermé).

Méthodes :

- **ouvrirSession() :** Méthode permettant d'ouvrir une nouvelle session pour un utilisateur.

- **fermerSession() :** Méthode permettant de fermer une session existante pour un utilisateur.

Classe Message :

La classe Message représente les messages échangés entre les utilisateurs de la plateforme. Elle permet la communication entre les élèves et les tuteurs.

 # Attributs :

- **idMessage : Identifiant unique du message.**

- **idExpediteur : Identifiant de l'utilisateur qui envoie le message.**

- **idDestinataire : Identifiant de l'utilisateur destinataire du message.**

- **contenu : Contenu du message.**

- **dateEnvoi : Date et heure d'envoi du message.**

- **lu : Indicateur indiquant si le message a été lu par le destinataire.**

- **épinglé : Indicateur indiquant si le message a été épinglé par le destinataire.**

 # Méthodes :

- **envoyerMessage() : Méthode permettant à un utilisateur d'envoyer un message à un autre utilisateur.**

- **épinglerMessage() : Méthode permettant à un utilisateur d'épingler un message dans sa boîte de réception.**

Classe **Événement** :

La classe Événement représente un événement planifié dans le calendrier d'un utilisateur. Elle permet de gérer les rendez-vous et les activités importantes.

 # Attributs :

- **idEvenement : Identifiant unique de l'événement.**

- **idUtilisateur : Identifiant de l'utilisateur associé à l'événement.**

- **date : Date de l'événement.**

- **description : Description de l'événement.**

 # Méthodes :

- **planifierÉvénement() : Méthode permettant à un utilisateur de planifier un événement dans son calendrier.**

- **modifierÉvénement() : Méthode permettant à un utilisateur de modifier un événement existant.**

- **supprimerÉvénement() : Méthode permettant à un utilisateur de supprimer un événement de son calendrier.**

Classe **Calendrier** :

La classe Calendrier gère le calendrier d'un utilisateur. Elle permet de stocker et de gérer les événements planifiés par l'utilisateur.

Attributs :

- **idCalendrier :** Identifiant unique du calendrier.
- **idUtilisateur :** Identifiant de l'utilisateur associé au calendrier.
- **événements :** Liste des événements associés à ce calendrier.

Méthodes :

- **afficherCalendrier() :** Méthode permettant à un utilisateur d'afficher son calendrier avec tous les événements associés.
- **ajouterÉvénement() :** Méthode permettant à un utilisateur d'ajouter un événement à son calendrier.
- **supprimerÉvénement() :** Méthode permettant à un utilisateur de supprimer un événement de son calendrier.

Classe **Tâche** :

La classe Tâche représente une tâche à réaliser pour un utilisateur. Elle permet de gérer les listes de tâches et les activités à accomplir.

Attributs :

- **idTâche :** Identifiant unique de la tâche.
- **idUtilisateur :** Identifiant de l'utilisateur associé à la tâche.
- **description :** Description de la tâche.
- **dateCréation :** Date de création de la tâche.
- **dateÉchéance :** Date d'échéance de la tâche.
- **statut :** Statut de la tâche (en cours, terminée, en attente, etc.).

Méthodes :

- **créerTâche() :** Méthode permettant à un utilisateur de créer une nouvelle tâche.

- **modifierTâche() : Méthode permettant à un utilisateur de modifier une tâche existante.**

- **supprimerTâche() : Méthode permettant à un utilisateur de supprimer une tâche de sa liste.**

-notificationTâche()**

-Tâcherealisee() :****

Nous pouvons voir en détail les cardinalités entre ces entités.

Cardinalités entre les classes

Voici comment les cardinalités peuvent être définies entre les différentes classes :

1. **Utilisateur - Session :**

Cardinalité : 1 utilisateur peut avoir 0 ou plusieurs sessions, mais chaque session est associée à exactement un utilisateur.

2. **Utilisateur - Message :**

Cardinalité : 1 utilisateur peut envoyer ou recevoir 0 ou plusieurs messages, mais chaque message est envoyé par un seul utilisateur expéditeur et reçu par un seul utilisateur destinataire.

3. **Utilisateur - Événement :**

Cardinalité : 1 utilisateur peut planifier 0 ou plusieurs événements, mais chaque événement est associé à exactement un utilisateur.

4. **Utilisateur - Calendrier :**

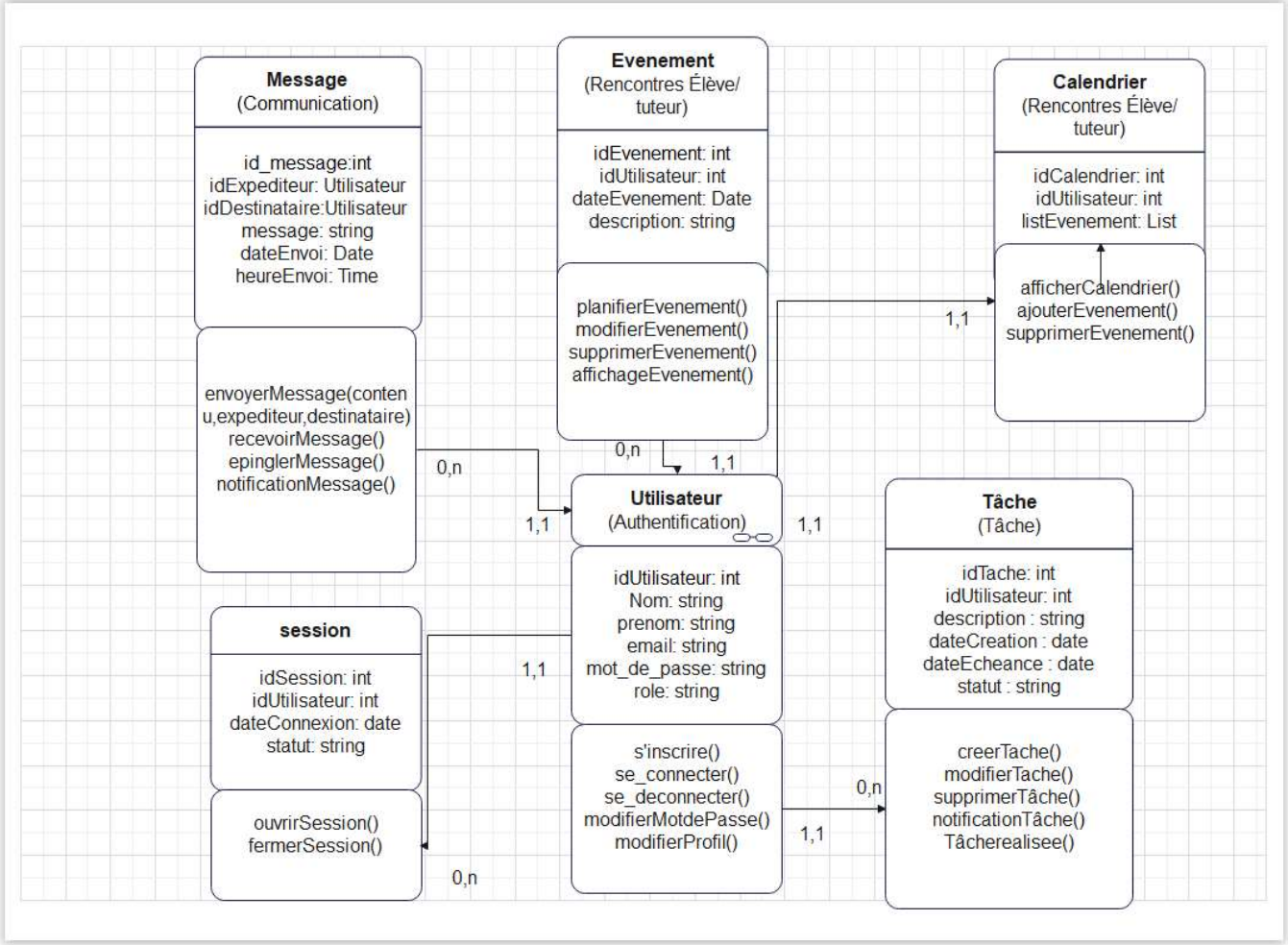
Cardinalité : Chaque utilisateur a exactement un calendrier, et chaque calendrier est associé à exactement un utilisateur.

5. **Utilisateur - Tâche :**

Cardinalité : 1 utilisateur peut créer, modifier ou supprimer 0 ou plusieurs tâches, mais chaque tâche est créée par un seul utilisateur.

Ces cardinalités définissent les relations entre les différentes entités de manière à refléter les interactions possibles dans le système. Elles permettent de comprendre comment les instances de chaque classe sont liées les unes aux autres dans le contexte de l'application.

Représentation du *Diagramme de Classe UML*


Message
(Communication)
id_message:int
idExpediteur: Utilisateur
idDestinataire:Utilisateur
message: string
dateEnvoi: Date
heureEnvoi: Time
envoyerMessage(contenu,expediteur,destinataire)
recevoirMessage()
epinglerMessage()
notificationMessage()
Evenement
(Rencontres Élève/ tuteur)
idEvenement: int
idUtilisateur: int
dateEvenement: Date
description: string
planifierEvenement()
modifierEvenement()
supprimerEvenement()
affichageEvenement()
Calendrier
(Rencontres Élève/ tuteur)
idCalendrier: int
idUtilisateur: int
listEvenement: List
afficherCalendrier()
ajouterEvenement()
supprimerEvenement()
Utilisateur
(Authentification)
idUtilisateur: int
Nom: string
prenom: string
email: string
mot_de_passe: string
role: string
s'inscrire()
se_connecter()
se_deconnecter()
modifierMotdePasse()
modifierProfil()
Tâche
(Tâche)
idTache: int
idUtilisateur: int
description : string
dateCreation : date
dateEcheance : date
statut : string
creerTache()
modifierTache()
supprimerTâche()
notificationTâche()
Tâcherealisee()
session
idSession: int
idUtilisateur: int
dateConnexion: date
statut: string
ouvrirSession()
fermerSession()
0,n
1,1
1,1
0,n
1,1
1,1
1,1
0,n
1,1
0,n

En clair,

1. A La fonctionnalité Authentification est attribuée la Classe Utilisateur

- Description : La classe Utilisateur gère les comptes utilisateur, y compris l'inscription, la connexion et la gestion des données personnelles. Cela correspond à la fonctionnalité d'authentification.

2. A La fonctionnalité Communication est attribuée la Classe Message

- Description : La classe Message représente les messages envoyés entre les utilisateurs de l'application. Cela correspond à la fonctionnalité de communication.

3. A La fonctionnalité Rencontres Élève/Tuteur est attribuée la Classe Evenement, et la Classe Calendrier

- Description : Les rencontres entre élèves et tuteurs sont gérées par la classe Événement, qui représente les événements planifiés dans le calendrier de l'utilisateur.

La classe Calendrier gère les événements associés à chaque utilisateur. Ces deux classes correspondent à la fonctionnalité de rencontres élève/tuteur.

4. A La fonctionnalité Tâches est attribuée la Classe Tâche

- Description : La classe Tâche représente les tâches à réaliser pour un utilisateur. Cela inclut les tâches résultant des rencontres élève/tuteur ainsi que celles créées individuellement par l'utilisateur. Cette classe correspond à la fonctionnalité de gestion des tâches.

Chaque classe est donc associée à une fonctionnalité spécifique de l'application, ce qui permet une séparation claire des responsabilités et une gestion modulaire des données et des opérations.

En conclusion,

Ce cahier des charges établit les bases essentielles pour le développement d'un site de soutien scolaire moderne et efficace. En mettant l'accent sur la personnalisation de l'expérience d'apprentissage, la sécurité des données, et l'optimisation des performances, ce projet vise à offrir aux élèves, aux parents et aux enseignants une plateforme interactive et adaptative pour renforcer leurs compétences et leur engagement dans l'éducation. En suivant les spécifications définies dans ce cahier des charges, nous aspirons à créer un site de soutien scolaire innovant et de haute qualité qui répondra aux besoins changeants du monde de l'éducation et contribuera à l'épanouissement académique de ses utilisateurs.