

Informaticien/-ne CFC

Travail pratique individuel 2024 (TPI)



ICT Berufsbildung
Formation professionnelle
Freiburg-Fribourg

Nom du candidat : Shendar ali

Candidat N° 148444

En Méga Forme

Sommaire

Résumé du rapport du TPI.....	2
1 Les grandes lignes du projet.....	3
1.1 Analyse de la situation initiale	3
1.2 Analyse de l'état désiré	3
1.3 Cahier des charges / exigences du système	4
1.4 Organisation du projet.....	4
2 Analyse préliminaire	6
2.1 Objectifs du système.....	6
2.2 Variantes.....	18
2.3 Choix de variante	19
2.4 Rentabilité.....	19
2.5 Analyse de risque	20
2.6 Sécurité de l'information et protection des données	21
3 Concept.....	22
3.1 Exigences du système	22
3.2 Architecture du système.....	27
3.3 Plan d'intégration des systèmes.....	28
3.4 Concept d'implémentation.....	28
3.5 Concept de formation.....	32
3.6 Concept de tests	33
3.7 Moyens nécessaires	36
Réalisation.....	37
3.8 Spécifications détaillées.....	37
3.9 Design du système	37
3.10 Configuration xyz	50
4 Test.....	52
4.1 Procédure de test.....	52
4.2 Protocole de test	52
4.3 Signature du protocole de test	57
5 Conclusion.....	58
5.1 Améliorations possibles	58
5.2 Auto-évaluation	59
6 Bibliographie: liste des sources et références.....	60
7 Glossaire.....	61
8 Signatures.....	63
9 Annexes	64

Résumé du rapport du TPI

Situation de départ

En Méga Forme, une entreprise de fitness, souhaite élargir son offre en proposant une application de coaching personnel pour la maison. Cette application, développée en tant que PWA avec des technologies comme HTML et CSS (Bootstrap), JS et PHP, générera des exercices ciblés pour les utilisateurs. Elle chronométrera les temps d'exercice et de pause, fournira des statistiques et enverra des notifications pour rappeler l'hydratation. L'objectif principal est de permettre aux clients de travailler différents groupes musculaires sans répétition excessive et d'assurer un suivi complet de leurs sessions d'entraînement.

Mise en œuvre

Le développement de l'application implique plusieurs phases, incluant l'analyse des besoins, la conception et la réalisation technique. L'analyse comprend la création de diagrammes UML et une documentation détaillée des besoins. La conception nécessite la modélisation relationnelle de la base de données et des diagrammes d'interactions. La réalisation se focalise sur le développement des interfaces utilisateur, la mise en place d'une architecture Frontend-Backend, l'implémentation des accès à la base de données et des fonctionnalités de notification. Le projet exige également des tests pratiques pour garantir son bon fonctionnement.

Résultats

L'application finale permet aux utilisateurs de sélectionner des muscles à travailler, de définir des temps d'exercice et de pause, et de générer des sessions d'entraînement personnalisées. Les administrateurs peuvent gérer les utilisateurs et les sessions d'entraînement, tandis que les utilisateurs peuvent suivre leurs progrès via une interface conviviale. Des notifications pour l'hydratation et une voix indiquant le temps restant et les prochains exercices enrichissent l'expérience de l'utilisateur.

1 Les grandes lignes du projet

1.1 Analyse de la situation initiale

Actuellement, l'équipe d'En Méga Forme crée des sessions sur papier, ce qui pose de nombreux problèmes. Les clients perdent souvent ces papiers, et les coachs ne peuvent pas suivre les sessions précédentes. De plus, les clients doivent demander à l'équipe comment réaliser correctement les exercices, ce qui n'est pas pratique.

D'un point de vue économique, cette méthode est inefficace. L'équipe perd beaucoup de temps à créer et imprimer chaque programme pour le remettre aux clients. Cela limite l'accès aux services car de nombreuses personnes souhaitent faire leurs sessions chez eux mais n'ont pas le temps de contacter l'équipe pour demander une nouvelle session ou aller chercher ces papiers.

Écologiquement, cette méthode génère une quantité importante de déchets papier. Sans trop entrer dans les détails, il est clair que la méthode actuelle pourrait être améliorée pour réduire son impact environnemental.

1.2 Analyse de l'état désiré

Pour chaque client enregistré dans ce service, l'administrateur crée un compte et peut également attribuer un statut d'administrateur à n'importe quel utilisateur. Dans la même interface, il peut voir tous les utilisateurs et leur dernière connexion. Ensuite, il peut générer des sessions d'entraînement en cas de besoin. Il lui suffit de donner un nom à la session, bien que l'application génère par défaut un nom du type "session_liste_de_muscles". L'administrateur sélectionne les utilisateurs, les muscles à travailler, le temps de travail, le temps de pause et le nombre d'exercices. Depuis un autre onglet, il peut aussi copier une session existante pour l'attribuer à d'autres utilisateurs.

Une fois que le compte du client est créé, celui-ci peut se connecter et visualiser toutes les sessions qui lui sont attribuées, avec des informations telles que le temps total de la session et le nombre de fois qu'il a déjà effectué chaque session. Il peut ensuite démarrer une session, ce qui le conduit à la page de la session en cours, ou modifier une session, ce qui l'amène à la page de création de session avec les données de la session choisie. Il peut ainsi modifier tous les détails de la session, sauf le nom. Il peut également supprimer une session. Il peut aussi indiquer la quantité d'eau qu'il souhaite boire par jour depuis la page de liste des sessions, et l'application se chargera de programmer les notifications pour l'hydratation.

L'utilisateur peut également générer une nouvelle session en choisissant les muscles qu'il veut exercer, le temps total de travail, le temps de pause et le nombre d'exercices. Comme l'administrateur, il peut donner un nom à la session ; là aussi, l'application génère par défaut un nom du type "session_liste_de_muscles". L'application se chargera de générer la session avec des exercices adaptés aux choix du client, en veillant à ce que les exercices ne se répètent pas.

Lorsqu'il clique sur le bouton "Démarrer" dans la liste des sessions, l'utilisateur est dirigé vers la page de la session en cours. Une fois la page chargée, un minuteur vocal s'exécute pour lui annoncer le nom de l'exercice à commencer dans 3 secondes. Le temps total de la session commence, et le minuteur affiche le temps restant. Un gif montrant le premier exercice se lance, avec en dessous le nom du prochain exercice et son gif représentatif. Dix secondes avant la fin du temps de travail pour l'exercice courant, un message vocal annonce qu'il y a une pause et compte les trois dernières secondes. Dix secondes avant la fin de la pause, un message vocal annonce le nom du prochain exercice, et finalement, trois secondes avant la fin de la pause, un message vocal compte les trois dernières secondes.

1.3 Cahier des charges / exigences du système

Le projet de développement d'une application de coaching personnel pour En Méga Forme vise à fournir une solution complète et intuitive pour les utilisateurs souhaitant s'entraîner à domicile. Utilisant les technologies HTML, CSS (Bootstrap), JavaScript et PHP, cette application sera développée en tant que Progressive Web App (PWA) compatible avec tous les supports. Les utilisateurs pourront sélectionner des groupes musculaires, définir des temps d'exercice et de pause, et générer automatiquement des séquences d'exercices tout en évitant la répétition excessive des mêmes muscles. L'application chronométrera les exercices et les pauses, enverra des notifications pour l'hydratation et enregistrera les statistiques dans une base de données.

Les fonctionnalités-clés incluent une interface utilisateur intuitive avec des pages de connexion pour les utilisateurs et les administrateurs, la gestion des comptes utilisateurs par les administrateurs, et la création, modification et suppression des sessions d'entraînement par les utilisateurs. Les notifications vocales indiqueront le temps restant et les prochains exercices, et des rappels d'hydratation configurables seront envoyés tout au long de la journée. La base de données MySQL enregistrera les statistiques et les sessions d'entraînement, assurant une gestion efficace des utilisateurs et des administrateurs.

L'infrastructure nécessaire comprend un ordinateur personnel Windows avec accès Internet, des logiciels de développement comme VSCode, Enterprise Architect, FileZilla, et MySQL Workbench, ainsi qu'un hébergement PHP et SQL. Le projet suivra plusieurs phases : analyse des besoins et création des diagrammes UML, conception de la base de données et des diagrammes d'interactions, réalisation des interfaces utilisateur et de l'architecture Frontend-Backend, et tests fonctionnels. La documentation sera produite tout au long du projet, incluant la planification, le journal de travail, et des guides d'utilisation.

1.4 Organisation du projet

Pour ce projet, j'ai décidé d'adopter une gestion de projet hybride entre « agile pour gitlab » et « canban pour planning », car je la trouve plus adaptée aux besoins spécifiques. L'approche hybride permet une grande flexibilité et une adaptation rapide aux changements, ce qui est essentiel pour le développement d'une application de coaching personnel. Avec des feedbacks fréquents, je peux continuellement améliorer l'application en fonction des retours des administrateurs.

L'agilité nous permettra également de mieux gérer les priorités et de garantir que les fonctionnalités les plus importantes et les plus demandées sont développées en premier.

Nom prénom de participant	Rôle
Shendar Ali	Développeur
Dimitri Colella	Le supérieur professionnel
Frédéric Hertling	Formateur en entreprise
Marc Denervaud	Expert-e principal-e
Hans Wilhelm Schwendimann	Expert-e secondaire

Les sauvegardes sont effectuées à la fin de chaque journée de travail. Une copie sera enregistrée sur GitLab via un commit-push contenant les données et la version actuelle de l'application. Une autre copie sera faite sur l'hébergement via une connexion FTP, une autre sur mon OneDrive, une copie locale sur l'ordinateur de l'école, et une dernière sur un disque dur externe.

Les versions de l'application seront gérées en deux phases principales : la version de base (sans notification pour boire de l'eau) et la version avec notifications. Les petites versions seront identifiées par un numéro décimal. À chaque tâche planifiée et complétée, le chiffre après la virgule sera incrémenté d'une unité. Cette approche permet de suivre les progrès et les mises à jour de manière systématique et organisée.

2 Analyse préliminaire

2.1 Objectifs du système

2.1.1 Analyse de l'état actuel

Tout commence par le client qui prend contact avec l'équipe d'En Méga Forme pour demander une session d'entraînement. L'administrateur lui demande les muscles qu'il souhaite travailler, le temps de travail, le temps de pause souhaité, ainsi que le nombre d'exercices par muscle.

Une fois ces informations recueillies, l'administrateur crée une session dans un fichier Word, calcule le temps total de la session, et ajoute ces détails dans le fichier. Ensuite, il exporte le document en PDF avec la liste des exercices, le temps de travail pour chaque exercice et le temps de pause entre chaque exercice. Ce fichier est ensuite stocké dans le dossier du client et envoyé à celui-ci.

Ce processus prend généralement entre 1 et 2 heures depuis l'appel du client jusqu'à l'envoi du PDF. Par ailleurs, garder une trace des sessions pour chaque client n'est pas toujours facile, ce qui peut entraîner une répétition excessive des mêmes exercices. Cela crée des problèmes pour l'équipe, car les clients peuvent se lasser et ne pas obtenir une variété suffisante dans leurs entraînements.

Une autre problématique est que le temps d'attente pour chaque client peut augmenter de plusieurs jours, surtout pendant les périodes de forte demande. En raison du nombre limité de coaches dans l'équipe, il devient difficile de répondre à toutes les demandes en temps voulu. Le temps de travail de chaque coach s'accroît également, ce qui constitue un problème pour certains d'entre eux. Cette situation non seulement impacte la satisfaction des clients mais aussi la qualité du service offert par l'équipe.

2.1.2 Analyse de l'état désiré

Un des axes majeurs que l'application va améliorer est l'automatisation du processus de génération de sessions. Grâce à l'application, les clients pourront générer eux-mêmes leurs sessions d'entraînement, ce qui libérera du temps pour les coaches. Ces derniers deviendront les administrateurs de l'application et leur rôle sera principalement d'ajouter de nouveaux utilisateurs et de générer des sessions en cas de besoin.

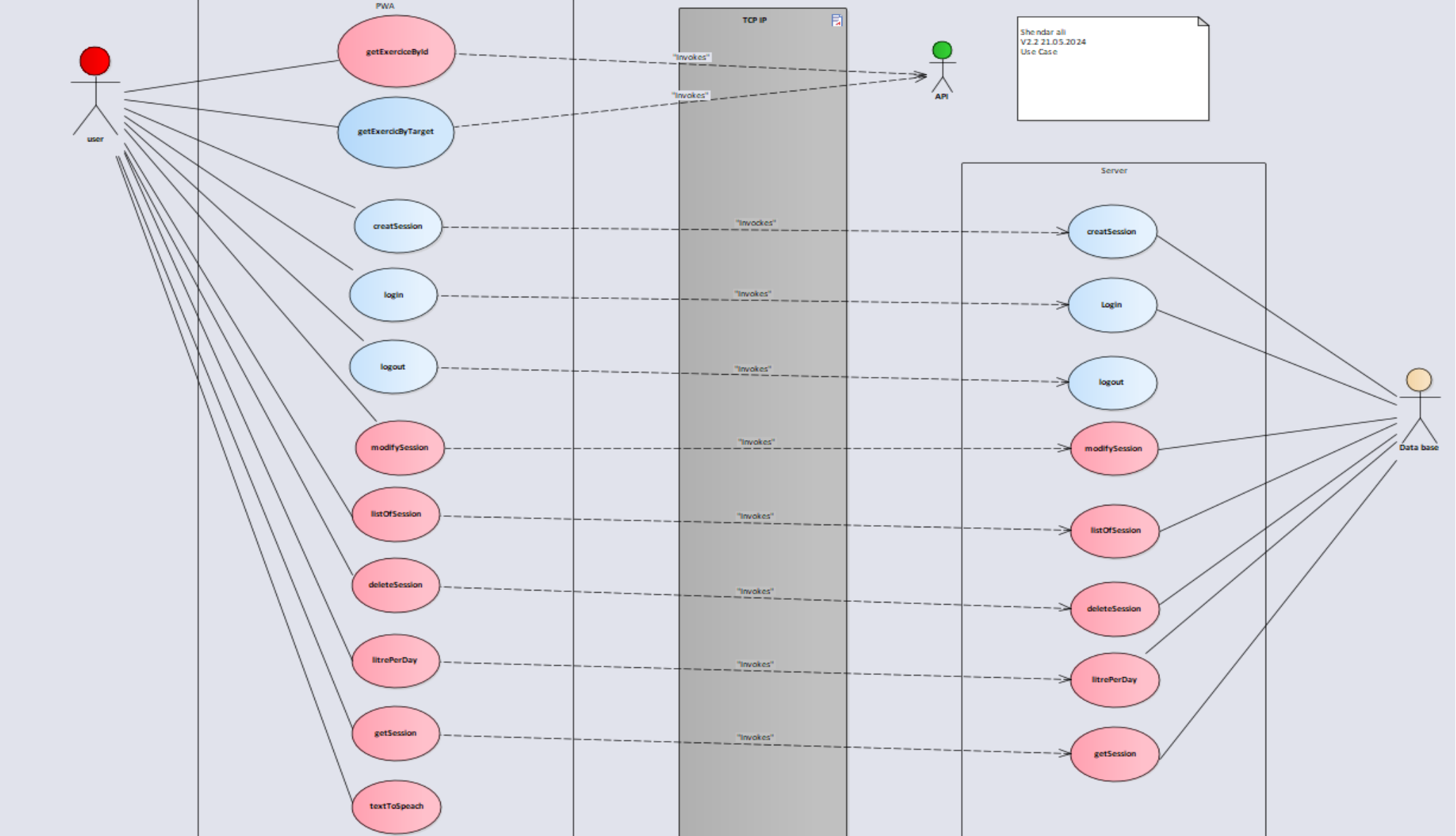
Use case :

Dans le diagramme suivant, j'ai décrit les actions que l'utilisateur et l'administrateur peuvent effectuer, ainsi que les interactions avec le Backend PHP ou l'API.

Les actions en rose représentent celles de l'utilisateur, comme `getExerciceById` ou `deleteSession`.

Les actions en bleu foncé sont celles de l'administrateur, comme `addUser` ou `copySession`. Toutes ces actions communiquent avec le Backend PHP.

Les actions en bleu clair sont communes entre l'utilisateur et l'administrateur, comme login, logout et createSession.



Use case globale user Entreprise architecteur

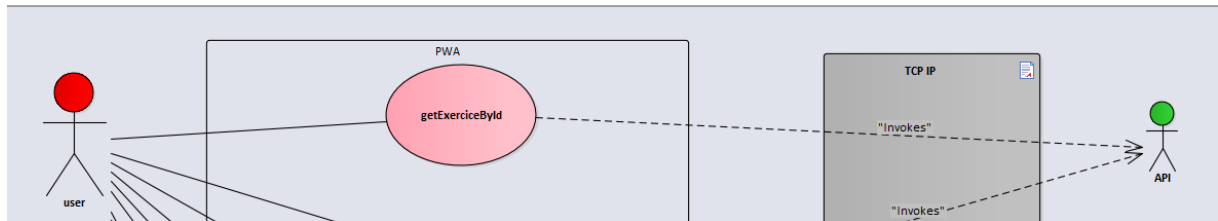
Première partie : Frontend-API

Cette partie a été modifiée, il ne suit plus cette méthodologie mais j'ai laissé la réflexion de base pour garder une trace de réflexion, la modification est documentée dans le chapitre **réalisation-> design du système -> problème**

Dans cette partie, les actions demanderont des données directement depuis l'API.

- **GetExerciceById** : Cette action est utilisée lorsque l'utilisateur démarre une session. Les identifiants des exercices sont stockés dans la session et seront lancés les uns après les autres. Cette action est chargée de récupérer les détails tels que le nom de l'exercice et le gif correspondant depuis l'API.

Ces actions permettent à l'application de fournir une expérience personnalisée et dynamique en récupérant des informations actualisées sur les exercices à partir de l'API.



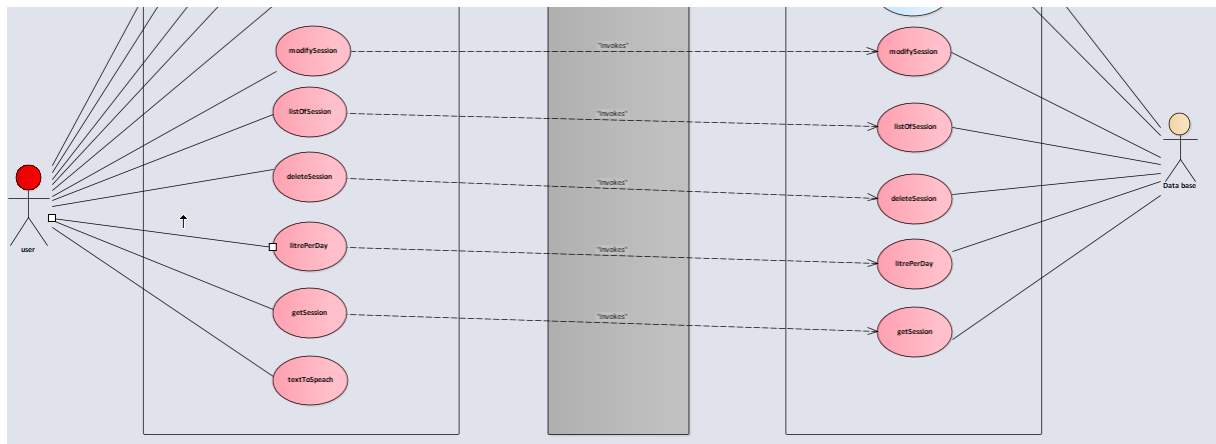
Exemple de use case pour se connecter à l'API entreprise architecteur

Deuxième partie : Frontend-Backend

Ces actions nécessitent une interaction entre le Frontend et le Backend :

- **modifySession** : Cette action demande les données d'une session depuis le Backend, puis les affiche pour le Frontend en rendant le nom de la session non-éritable. Tous les autres paramètres, comme les muscles, le temps de travail, le temps de pause et le nombre d'exercices, sont éditables. Une fois les modifications enregistrées, les nouvelles données de la session sont envoyées au Backend, qui les met à jour dans la base de données.
- **listOfSession** : Cette action demande toutes les données des sessions liées à l'utilisateur courant, puis les affiche pour l'utilisateur.
- **deleteSession** : Cette action envoie une requête au Backend pour supprimer la session concernée. Le Backend la supprime alors de la base de données.
- **litrePerDay** : Une fois que l'utilisateur a entré la quantité d'eau qu'il souhaite boire par jour, cette information est envoyée au Backend, qui l'enregistre dans la base de données.
- **getSession** : En démarrant une session, une requête est envoyée au Backend pour demander les données de la session. Le Backend renvoie les données nécessaires pour les afficher pour l'utilisateur.
- **textToSpeech** : Cette action est appelée chaque fois qu'un message vocal doit être envoyé à l'utilisateur.

Ces actions garantissent que les données utilisateur sont correctement gérées et stockées sur le Backend, tout en offrant une expérience utilisateur fluide et interactive.



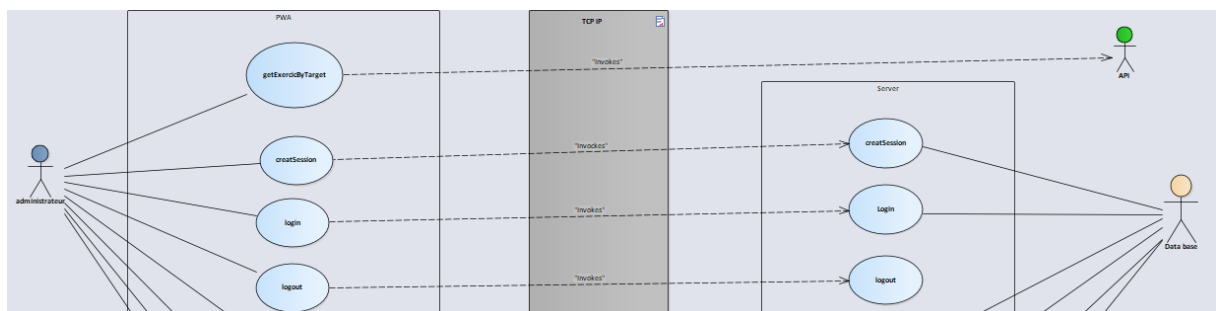
Use case les interactions entre Frontend user et Backend entreprise architecteur

Troisième partie : les actions en commun Frontend-administrateur,

Ces actions impliquent généralement une interaction avec le Backend PHP :

- **createSession** : Cette action est appelée une fois que l'utilisateur ou l'administrateur a entré tous les paramètres. Elle appelle l'action **getExerciceByTarget** pour récupérer les identifiants des exercices, puis envoie ces données au Backend, qui se charge de les enregistrer dans la base de données.
- **login** : Cette action envoie le nom d'utilisateur et le mot de passe au Backend pour vérification. Si la vérification est réussie, une session est créée et l'information est envoyée au Frontend pour confirmer la réussite de la connexion. En cas de connexion réussie, le Backend enregistre également la date et l'heure de la connexion dans la base de données.
- **logout** : Cette action demande au Backend de terminer la session de connexion.
- **GetExerciceByTarget** : Cette action est utilisée lorsque l'application génère une session. Elle prend les muscles que le Frontend a choisis et les envoie à l'API pour récupérer les ids des exercices proposés par l'API.

Ces actions assurent une gestion sécurisée et efficace des sessions utilisateur, en garantissant que les informations sont correctement vérifiées et stockées.

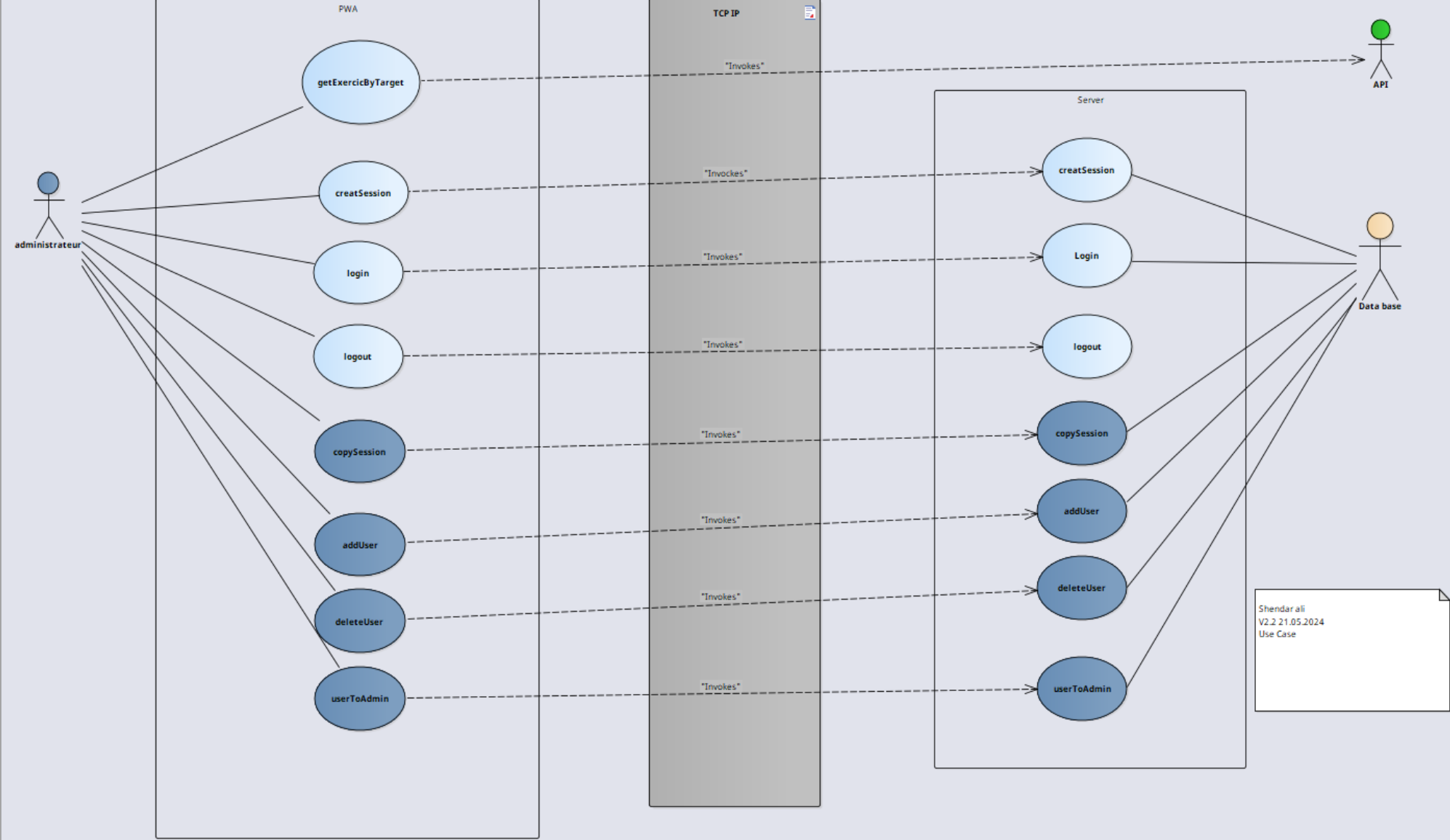


Use case interaction entre Frontend administrateur et backend entreprise architecteur

Quatrième partie : administrateur- Backend

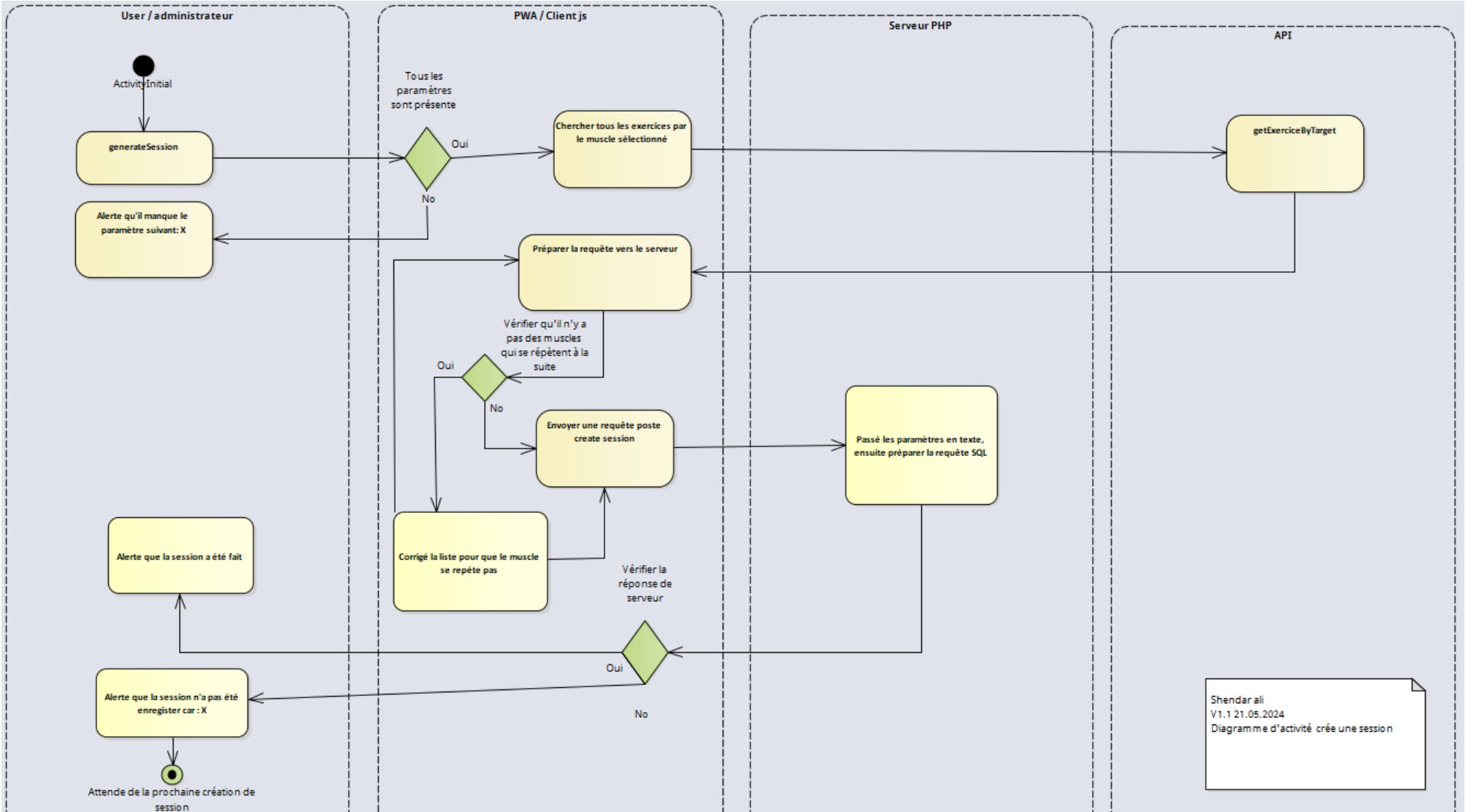
Ces actions concernent uniquement l'administrateur :

- **copySession** : Cette action demande la liste des sessions d'un utilisateur et les copie chez un autre utilisateur. L'application envoie les identifiants des deux utilisateurs et l'identifiant de la session au Backend, qui se charge de créer une nouvelle session dans la base de données.
- **addUser** : Après que l'administrateur a entré le nom d'utilisateur et le mot de passe, ces informations sont envoyées au Backend, qui se charge de les enregistrer dans la base de données.
- **deleteUser** : L'administrateur entre le nom d'utilisateur à supprimer, et cette information est envoyée au Backend. Le Backend supprime l'utilisateur et tous les enregistrements liés à son compte de la base de données.
- **userToAdmin** : l'administrateur peut changer la statuts d'une utilisateur ou administrateur de admin vers utilisateur et l'inverse



Use case pour interaction entre Frontend administrateur et Backend et API entreprise architecteur

Diagramme d'activité :



Shendar ali
V1.1 21.05.2024
Diagramme d'activité créer une session

Diagramme activité entreprise architecteur

Le diagramme d'activité illustre le processus de création d'une session d'entraînement par un utilisateur ou un administrateur dans l'application. Le processus commence par l'initiation de l'activité « générer une session » par l'utilisateur ou l'administrateur. Cette action déclenche une vérification par le Frontend PWA/JS pour s'assurer que tous les paramètres nécessaires sont présents. Si tous les paramètres sont présents, le JS recherche tous les exercices correspondant au muscle sélectionné.

Si des paramètres manquent, une alerte est envoyée à l'utilisateur pour indiquer quels paramètres sont manquants, et le processus de création de session est arrêté jusqu'à la prochaine tentative.

Ensuite, si les paramètres sont complets, le Frontend prépare une requête à envoyer au Backend. Avant d'envoyer cette requête, il vérifie qu'il n'y a pas des muscles répétitifs à la suite dans la sélection. Si des répétitions sont détectées, il corrige et remet la liste vers la méthode de renvoi.

Une fois que toutes les vérifications sont passées, le Frontend envoie une requête POST au Backend PHP pour créer la session. Le Backend prend les paramètres en texte et prépare une requête SQL pour l'enregistrer dans la base de données.

Le Frontend vérifie ensuite la réponse du Backend. Si la réponse est positive et la session est bien créée, une confirmation est envoyée à l'utilisateur pour indiquer que la session a été enregistrée avec succès. Si la réponse est négative et la session n'a pas pu être enregistrée, une alerte est envoyée à l'utilisateur pour indiquer l'échec de l'enregistrement, et le processus attend la prochaine création de session.

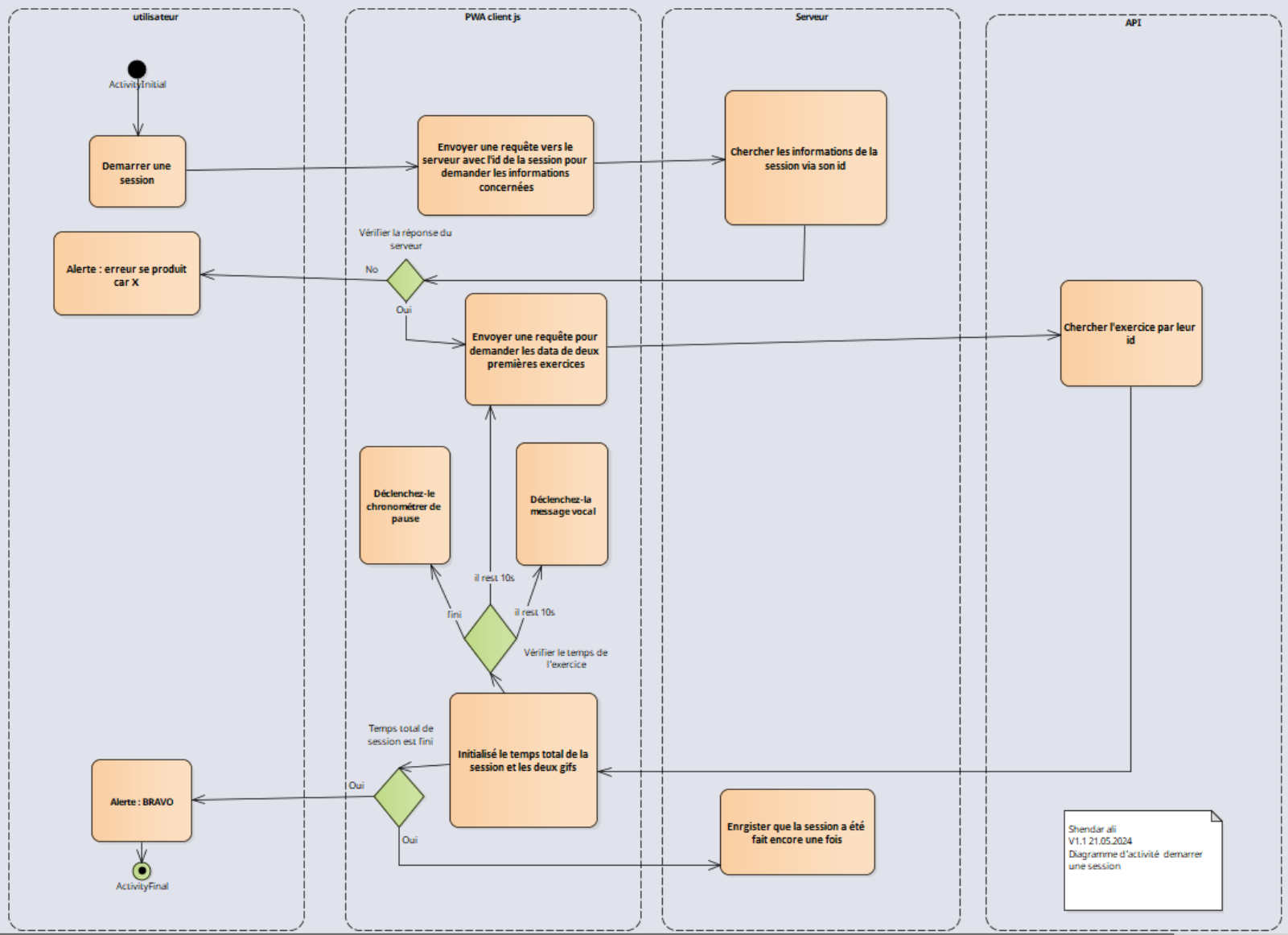


Diagramme activité entreprise architecteur

Ce diagramme d'activité illustre le processus de démarrage et d'exécution d'une session d'entraînement dans une application.

Le processus commence lorsque l'utilisateur initie l'activité en démarrant une session d'entraînement. Une requête est alors envoyée depuis le Frontend PWA/JS vers le Backend avec l'ID de la session pour demander les informations nécessaires. Le Backend cherche les informations de la session via son ID.

Une fois que le Backend a récupéré les informations nécessaires, la réponse est vérifiée par le Frontend. Si la réponse du Backend n'est pas satisfaisante, une alerte est générée pour informer l'utilisateur de l'erreur survenue. Si la réponse est positive, le Frontend PWA/JS envoie une nouvelle requête pour demander les données des deux premiers exercices de la session, depuis l'API.

Après réception des données, le Frontend déclenche le chronomètre pour la session et le premier exercice. Lorsque le temps de cet exercice est terminé, un chronomètre de pause est déclenché, suivi du second exercice. Si l'intervalle de temps restant pour l'exercice atteint 10 secondes, une vérification est effectuée pour préparer le prochain exercice ou la prochaine pause. Un message vocal sera déclenché pour indiquer la prochaine étape et chronométrer les neuf dernières secondes.

Tout au long du processus, le temps total de la session et les animations GIF des exercices sont initialisés et mis à jour pour guider l'utilisateur. Lorsque le temps total de la session est écoulé, une alerte "BRAVO" est générée pour féliciter l'utilisateur. Enfin, le système enregistre que la session a été effectuée une nouvelle fois, clôturant ainsi le processus.

2.1.3 Processus d'entreprise concernés

Tout d'abord, il y a la gestion des utilisateurs et des administrateurs, qui inclut la création et la gestion des comptes utilisateurs, l'attribution et la gestion des statuts administratifs, ainsi que le suivi des connexions des utilisateurs.

Ensuite, la personnalisation des sessions d'entraînement est essentielle, permettant la sélection des groupes musculaires à travailler, la définition des temps d'exercice et de pause, et la génération automatique des séquences d'exercice sans répétition excessive des mêmes muscles. Il est également possible de copier et modifier les sessions d'entraînement existantes.

En outre, le suivi et les notifications jouent un rôle important, avec le chronométrage des exercices et des pauses, l'envoi de notifications pour l'hydratation tout au long de la journée, et la gestion des statistiques des exercices.

Enfin, l'infrastructure technique et les sauvegardes sont cruciales, avec l'utilisation de GitLab pour le suivi des versions, la sauvegarde des données sur divers supports (FTP, OneDrive, local, disque dur externe), et la gestion des versions de l'application, incluant des phases de base et avancées avec notifications. Ces processus assurent que l'application offre une expérience utilisateur fluide et personnalisée, tout en permettant une gestion efficace et sécurisée des données et des utilisateurs.

2.1.4 Objectifs

Exigence fonctionnelle	Description	Priorité
Lecture du cahier des charges	Il faut comprendre l'énoncé, la portée et la complexité du projet avant toute autre action.	Haute
Analyse des besoins	Réalisation des diagrammes UML d'analyse sur la base des maquettes présentées dans ce document. Réalisation d'une documentation d'analyse.	Haute
Planning	Réalisation d'une planification détaillée ainsi qu'un suivi, représentant les différentes tâches, de l'analyse, de la conception, de l'implémentation, des tests et de la documentation, à réaliser dans ce projet.	Haute
Conception	Réalisation des diagrammes UML de conception permettant de représenter	Haute
Création et implémentation de la base de données	Création et implémentation de la base de données ainsi que les services utiles sur le domaine.	Haute
Implémentation de l'application	Implémentation de l'application et ses échanges avec les services de la base de données.	Haute
Documentation	Tout au long du projet, la documentation sera mise à jour régulièrement et finalisée après chaque phase de développement terminée. Le dernier jour permettra de finaliser les différentes documentations du projet avant de les rendre selon les consignes demandées.	Haute

Tests fonctionnels	Réalisation des tests fonctionnels de l'application et des accès à la base de données en situation de production.	Moyenne
Implémentation de l'application (notifications et calcul de l'eau)	Les notifications et le calcul de l'eau consommée par le client.	Basse

2.2 Variantes

2.2.1 MVC2

L'application selon un modèle implique l'utilisation de classes et d'objets pour représenter les différentes entités et fonctionnalités de l'application, telles que les utilisateurs, les sessions d'entraînement, et les notifications. Ce qui peut faciliter la gestion de projet.

Avantages :

- **Performance** : adopter un modèle orienté objet peut améliorer la performance en structurant le code de manière plus organisée et modulaire, ce qui facilite la maintenance et la réutilisabilité des composants. Les classes et objets permettent une encapsulation des données et des fonctions, réduisant ainsi les erreurs et les redondances. Enfin, permettent d'étendre facilement les fonctionnalités existantes sans impacter les performances globales du système.
- **Débogage facile** : l'orientation objet facilite le débogage en encapsulant les données et les comportements dans des classes distinctes, ce qui permet d'isoler et de corriger plus facilement les erreurs. Les méthodes et les attributs sont bien définis et structurés, ce qui rend le suivi des bugs plus clair et organisé. De plus, la réutilisation de code bien testé réduit les risques d'introduire de nouveaux bugs.

Désavantages :

- **Complexité Initiale** : la conception et la mise en œuvre d'une architecture orientée objet peuvent être plus complexes et prendre plus de temps initialement. Il faut bien planifier les classes et leurs relations.
- **Apprentissage et adoption** : pour les personnes qui ne sont pas familières avec l'OJ, il peut y avoir une courbe d'apprentissage. La transition d'une programmation fonctionnelle ou procédurale à une programmation orientée objet peut nécessiter du temps et des efforts.

2.2.2 Système de fonction

L'application Frontend serait développée en utilisant un modèle de programmation fonctionnelle au lieu d'une approche orientée objet. Cela signifie que l'application serait construite

principalement autour de fonctions pures qui manipulent des données immuables, plutôt que des objets et des classes.

Avantages :

- **Moins de Bugs** : En évitant les états changeables et en favorisant les fonctions pures, on réduit les risques de bugs liés à l'état et aux effets de bord, ce qui peut améliorer la fiabilité du code.
- **Simplicité et Lisibilité** : les fonctions pures sont souvent plus simples et plus faciles à comprendre que les classes et les objets. Cela peut rendre le code plus lisible et plus facile à maintenir, surtout pour les développeurs qui sont déjà familiers avec la programmation fonctionnelle.

Désavantages :

- **Performance** : Les opérations fonctionnelles sur des structures de données immuables peuvent parfois être moins performantes que les manipulations directes d'objets mutables, notamment pour des applications intensives en ressources.
- **Modularité et Organisation** : La programmation orientée objet offre des structures bien définies pour organiser le code (classes, objets, héritage, etc.). Dans une approche fonctionnelle, il peut être plus difficile de structurer des projets complexes de manière claire et modulaire.

2.3 Choix de variante

J'ai choisi le modèle MVC2 pour faciliter la prise en main de l'application. Bien que l'initialisation de ce système puisse être difficile, le débogage devient ensuite beaucoup plus simple grâce à l'encapsulation et à la modularité offertes par l'orientation objet. De plus, il est crucial de gagner en performance, car l'application doit fonctionner en continu pour gérer les notifications, ce qui nécessite une structure robuste et efficace. Une application efficace est essentielle pour éviter tout problème de performance, surtout dans un contexte où elle doit tourner 24/24.

2.4 Rentabilité

Actuellement, le processus de gestion des demandes de sessions d'entraînement par En Méga Forme est long et inefficace. Voici les points-clés de la situation actuelle et l'utilité économique du projet proposé :

- **Réduction du Temps de Traitement** :
 - a. L'automatisation de la génération des sessions via l'application réduira considérablement le temps de traitement, permettant de gérer plus de demandes en moins de temps. Cela augmentera l'efficacité opérationnelle et réduira les coûts de main-d'œuvre.
- **Amélioration de la Gestion des Sessions** :
 - a. L'application permet un suivi centralisé et automatisé des sessions, garantissant une plus grande variété et une personnalisation accrue des programmes d'entraînement. Cela peut améliorer la satisfaction et la fidélisation des clients, réduisant

ainsi les coûts liés à la perte de clients et au besoin constant de trouver de nouveaux clients.

- Optimisation des Ressources Humaines :
 - a. En libérant du temps pour les coachs grâce à l'automatisation, ceux-ci peuvent se concentrer sur des tâches à plus forte valeur ajoutée, comme l'interaction directe avec les clients et la conception de programmes personnalisés. Cela optimise l'utilisation des ressources humaines et améliore la qualité du service.
- Augmentation de la Capacité de Service :
 - a. L'application peut traiter simultanément un grand nombre de demandes, augmentant ainsi la capacité de service sans nécessiter une augmentation proportionnelle du personnel. Cela peut conduire à une croissance des revenus sans une augmentation significative des coûts fixes.
- Amélioration de la Satisfaction Client :
 - a. Une application qui fournit rapidement des sessions variées et personnalisées peut améliorer la satisfaction des clients, augmentant ainsi la fidélité et les recommandations, ce qui contribue à une croissance durable des revenus.

En conclusion, l'utilité économique du projet de développement de l'application de coaching personnel est significative. Elle permet de réduire les coûts opérationnels, d'optimiser les ressources humaines, d'augmenter la capacité de service, et d'améliorer la satisfaction et la fidélisation des clients. Ces avantages se traduisent par une amélioration globale de l'efficacité et de la rentabilité de l'entreprise En Méga Forme.

2.5 Analyse de risque

Inefficacité Persistante :

Le processus manuel actuel continuera de prendre entre 1 et 2 heures pour chaque demande de session d'entraînement. Ce qui peut conduire à des conséquences comme limiter la capacité de l'entreprise à gérer efficacement les demandes, entraînant des coûts opérationnels élevés et une utilisation inefficace des ressources humaines.

Perte de Compétitivité :

L'entreprise risque de perdre son avantage concurrentiel face à des concurrents qui offrent des solutions technologiques plus avancées et plus efficaces. Cela peut entraîner une diminution des parts de marché et des revenus.

Investissement Perdu :

Les ressources financières, humaines et temporelles investies dans le projet seront gaspillées. Cela représentera un investissement et du temps perdu de mon côté, ainsi que du côté d'En Méga Forme.

Réputation Impactée :

Pour ma part, ma réputation en sera impactée car je me suis lancé dans un projet que je n'ai pas terminé correctement, et du côté d'En Méga Forme, cela peut réduire la confiance des clients actuels et potentiels dans l'entreprise.

2.6 Sécurité de l'information et protection des données

Accès Non Autorisé

Des personnes non autorisées pourraient accéder aux données sensibles des utilisateurs. Pour prévenir cela, un système de login sécurisé avec mot de passe est en place.

Violation de Données

Des attaquants pourraient voler ou altérer les données des utilisateurs. Pour prévenir cela, la gestion des sessions sera mise en œuvre, garantissant que le Backend ne répond qu'aux requêtes provenant de sessions authentifiées. Les données seront chiffrées en transit (SSL/TLS) et au repos. Pour se protéger contre les injections SQL, l'utilisation de « Prepared Statements » en PHP est recommandée, permettant de passer toutes les requêtes à la base de données en texte et empêchant l'exécution de requêtes non définies.

Faibles de Sécurité dans le Code

Des vulnérabilités dans le code de l'application pourraient être exploitées par des attaquants. Pour contrer ces effets, des tests rigoureux seront effectués pour chaque source potentielle de vulnérabilité. Des revues de code régulières et des tests de pénétration seront également réalisés pour identifier et corriger les faibles de sécurité.

Conformité à la Protection des Données

Dans le cadre de ce projet, les données personnelles des utilisateurs se limitent principalement à leur nom. L'équipe d'En Méga Forme doit obtenir l'autorisation des clients. L'application n'est pas censée collecter d'autres données sensibles. Une fois qu'un compte doit être supprimé, toutes les données liées seront définitivement effacées de la base de données. Pour assurer un haut niveau de qualité et de transparence, l'application est hébergée chez un partenaire suisse de confiance : Alphosting.

3 Concept

3.1 Exigences du système

Frontend :

Pour utiliser l'application, un smartphone ou un iPhone est requis. Il est également nécessaire d'avoir un navigateur internet tel que Google Chrome, Firefox, Safari, ...etc.

Et par fonctionnalité :

Fonctionnalités	Navigateur	Supporter
Local Notifications	Chrome	OK
	Safari (iOS)	KO, pas trop
	Firefox	OK
Texte to speech	Chrome	OK
	Safari (iOS)	OK
	Firefox	OK

Il est recommandé d'utiliser Google Chrome pour une expérience optimale. Une connexion internet est indispensable pour télécharger et accéder aux données nécessaires au fonctionnement de l'application.

Backend :

Un serveur Apache2 est nécessaire pour exécuter le Backend PHP. Une base de données MySQL est également requise. Ces deux éléments sont fournis et gérés par notre partenaire Alphosting.

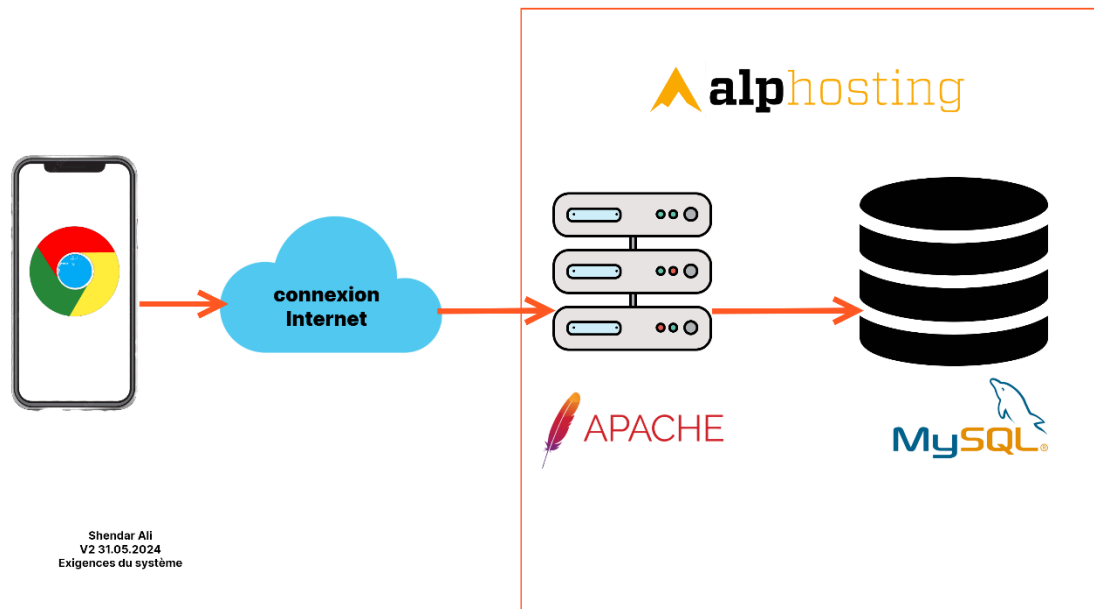


Schéma interaction globale Lunacy

Base de données :

La liaison entre la table T_User et T_Session a été modifiée, l'explication de la modification a été faite dans le chapitre Réalisation-> design de système -> problème

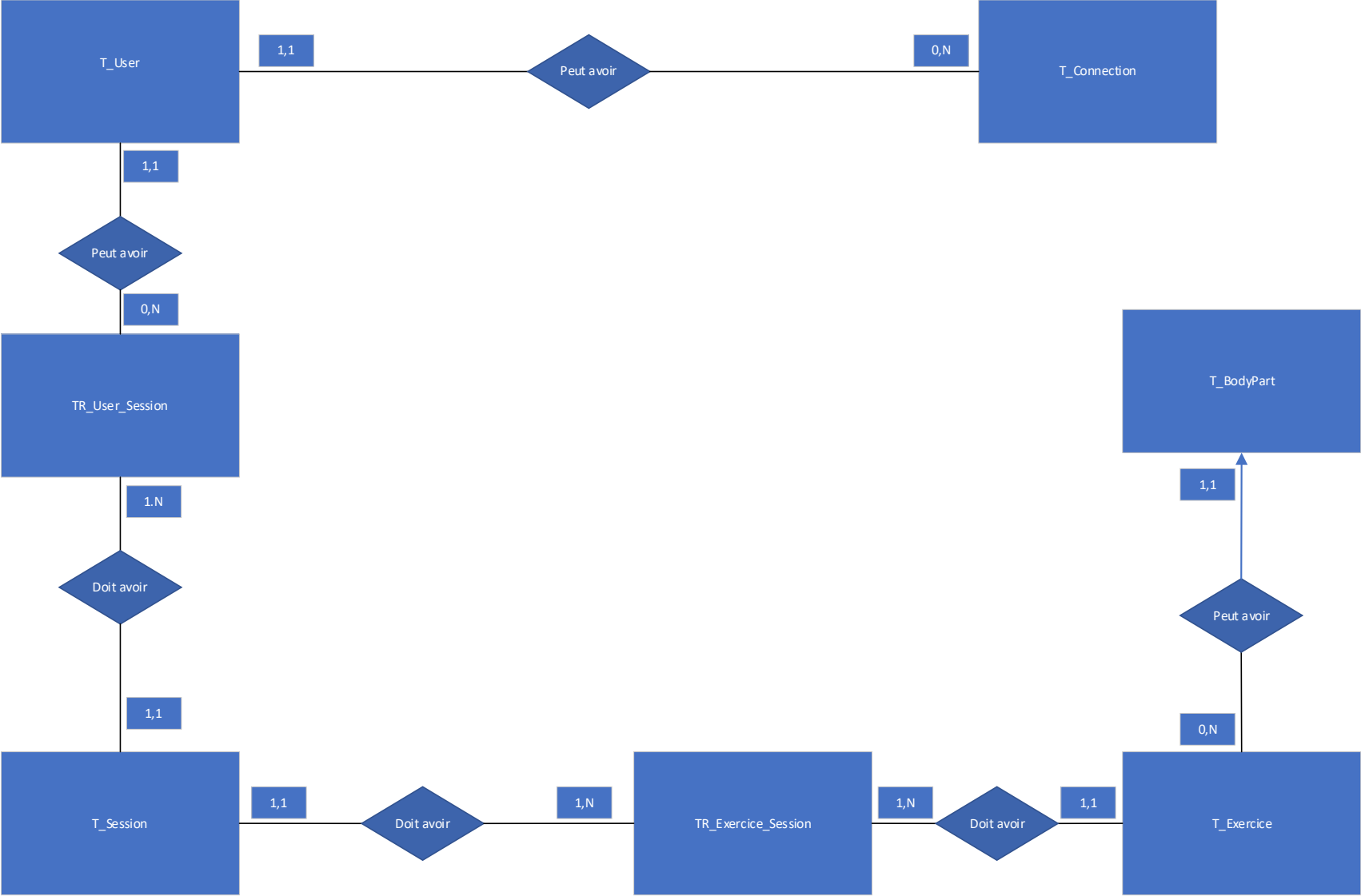


Schéma de base de données visio

J'ai enregistré les données essentielles de l'application, pour limiter les requêtes vers l'API ExerciceDB, et aussi pour la confidentialité des données d'application.

Table : T_User :

Il contient des informations sur les utilisateurs. Chaque utilisateur est identifié de manière unique par pk_user, une clé primaire entière. Les autres colonnes incluent un nom d'utilisateur de type varchar (50), un password, un mot de passe haché de type varchar (255), status, un entier représentant le statut de l'utilisateur admin ou utilisateur (il faut utiliser 0 pour utilisateur et 1 pour admin), et un litresByDay, un float indiquant la consommation quotidienne d'eau en litres par l'utilisateur.

Liaison :

- T_Connection via une relation un-à-plusieurs
- TR_User_Session via une relation zéro-à-plusieurs également.

Table : T_Connection :

La table T_Connection enregistre les temps de connexion des utilisateurs. Elle est identifiée par pk_connection, une clé primaire entière. La colonne fk_User est une clé étrangère qui fait référence à pk_user dans la table T_User, indiquant l'utilisateur associé à la connexion. La colonne connection_time est de type datetime et enregistre le moment de la connexion. Cette table a une relation plusieurs-à-un avec la table T_User.

Liaison :

- T_User via une relation plusieurs-à-un

Table : T_Session :

La table T_Session contient des informations sur les sessions. La clé primaire pk_session est un entier unique pour chaque session. La colonne name est un varchar (100) qui stocke le nom de la session et t_temps_total est un entier qui indique la durée totale de la session.

Liaison :

- TR_User_Session par une relation zéro -à-plusieurs
- TR_Exercice_Session par une relation un-à-plusieurs

Table : T_Exercice :

La table T_Exercice contient des informations sur les exercices. La clé primaire pk_exercice est un entier unique pour chaque exercice. La colonne exercice_id est varchar (255) identifiant l'exercice.

Liaison :

- TR_Exercice_Session : une relation un-à-plusieurs

Table : T_muscle :

La table T_muscle contient des informations sur les muscles impliqués dans les exercices. La clé primaire pk_muscle est un entier unique pour chaque muscle et la colonne name est un varchar (50) qui stocke le nom du muscle.

Liaison :

- T_Exercice : une zéro un-à-plusieurs

Table : TR_Exercice_Session :

La table TR_Exercice_Session représente la relation entre les exercices et les sessions. La colonne fk_exercice est une clé étrangère référant à T_Exercice(pk_exercice) et fk_session est une clé étrangère référant à T_Session(pk_session). Les colonnes time_of_work et time_of_rest sont des entiers indiquant respectivement le temps de travail et le temps de repos pour l'exercice dans la session, nbreExerciceByMuscle représente le nombre des exercices par muscle.

Liaison :

- T_Exercice : une relation plusieurs-à-un
- T_Session : une relation plusieurs-à-un

TR_User_Session :

La table TR_User_Session représente la relation entre les utilisateurs et les sessions. Les colonnes fk_user et fk_session sont des clés étrangères faisant référence respectivement à T_User(pk_user) et T_Session(pk_session). La colonne doneNbr est un entier indiquant le nombre de fois que l'utilisateur a complété la session.

Liaison :

- T_User : relations plusieurs-à-un
- T_Session : relations plusieurs-à-un

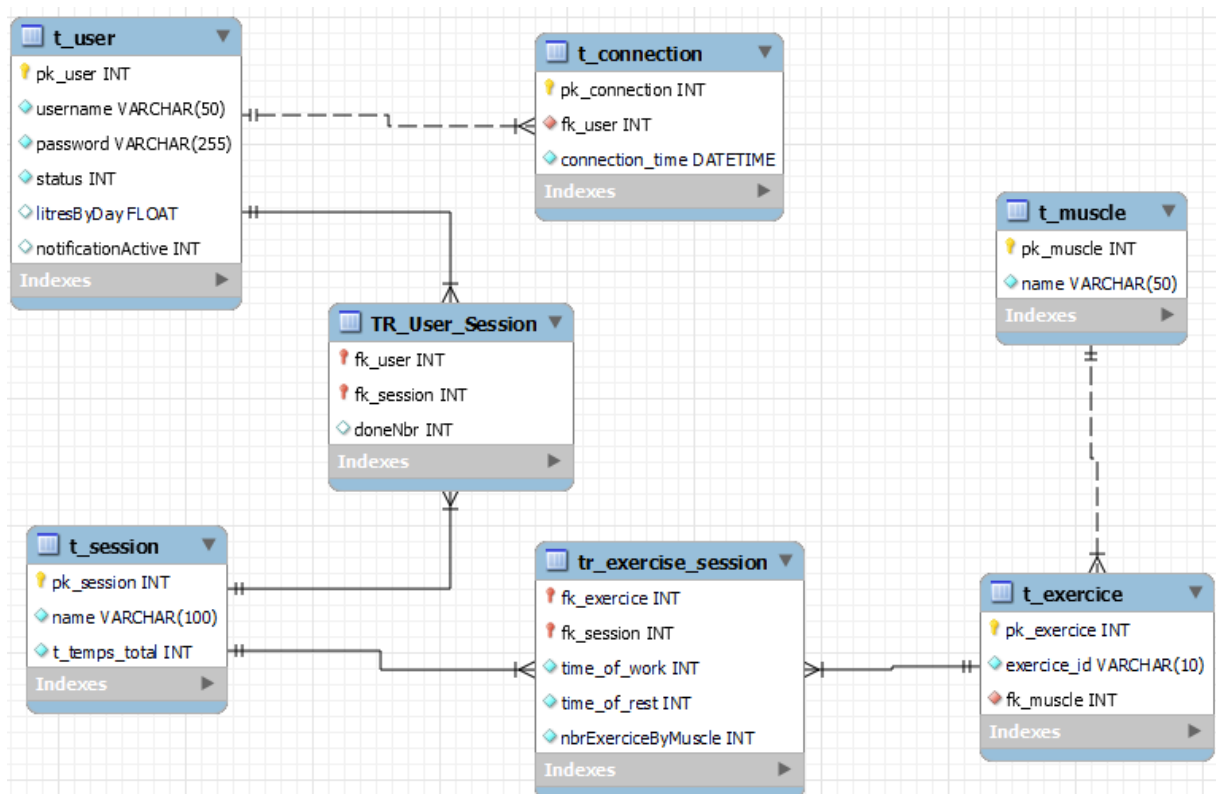


Schéma de base de données MYSQL workbench

3.2 Architecture du système

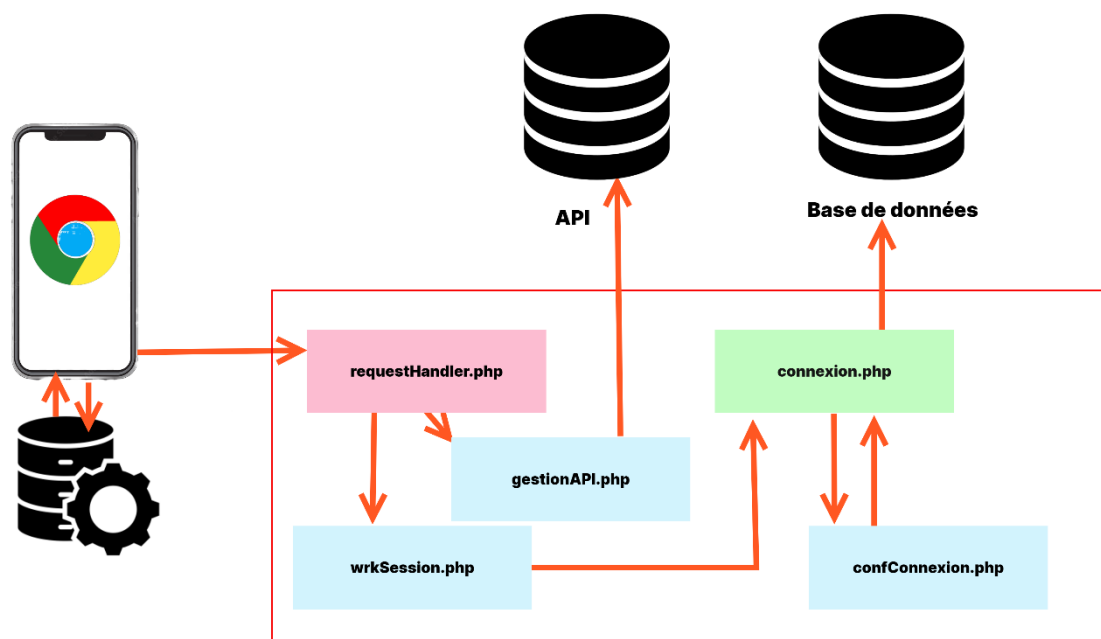


Schéma globale Lunacy

L'architecture d'un système d'interaction pour la création d'une session comprend plusieurs composants. L'utilisateur interagit avec un Backend PHP et une base de données (DB) et avec une API externe.

Lorsque l'utilisateur crée une session, une autre requête est dirigée vers le Backend PHP, spécifiquement vers le fichier requestHandler. Ce fichier est chargé de gérer les requêtes des utilisateurs en traitant les informations nécessaires.

L'utilisateur qui va faire passer la demande des target vers la gestionAPI va chercher les data depuis l'API et ensuite passer les data vers le Frontend qui est chargé d'extraire les id d'exercices et les passer aux Backend php.

Dans le backend les données passent ensuite vers le fichier gestionOfSession, qui est responsable de la gestion des sessions. Il crée, modifie, supprime ou recherche les données d'une session. Ce fichier prépare la requête SQL et la transmet à ConnectionDB, qui utilise les configurations de connexion à la base de données provenant du fichier ConnectionConf. Enfin, la requête est envoyée à la base de données

La deuxième étape consiste à transmettre la réponse de la base de données à gestionOfSession, qui à son tour la transmet à requestHandler pour finalement envoyer cette information à l'utilisateur.

3.3 Plan d'intégration des systèmes

Le système va s'entre communiquer via le protocole http, donc le frontend communique avec le backend en utilisant les méthodes spécifiques de protocole http comme :

- POST : pour insérer des nouvelles données
- GET : pour demander les données
- PUT : pour modifier un enregistrement
- DELETE : pour supprimer un enregistrement

3.4 Concept d'implémentation

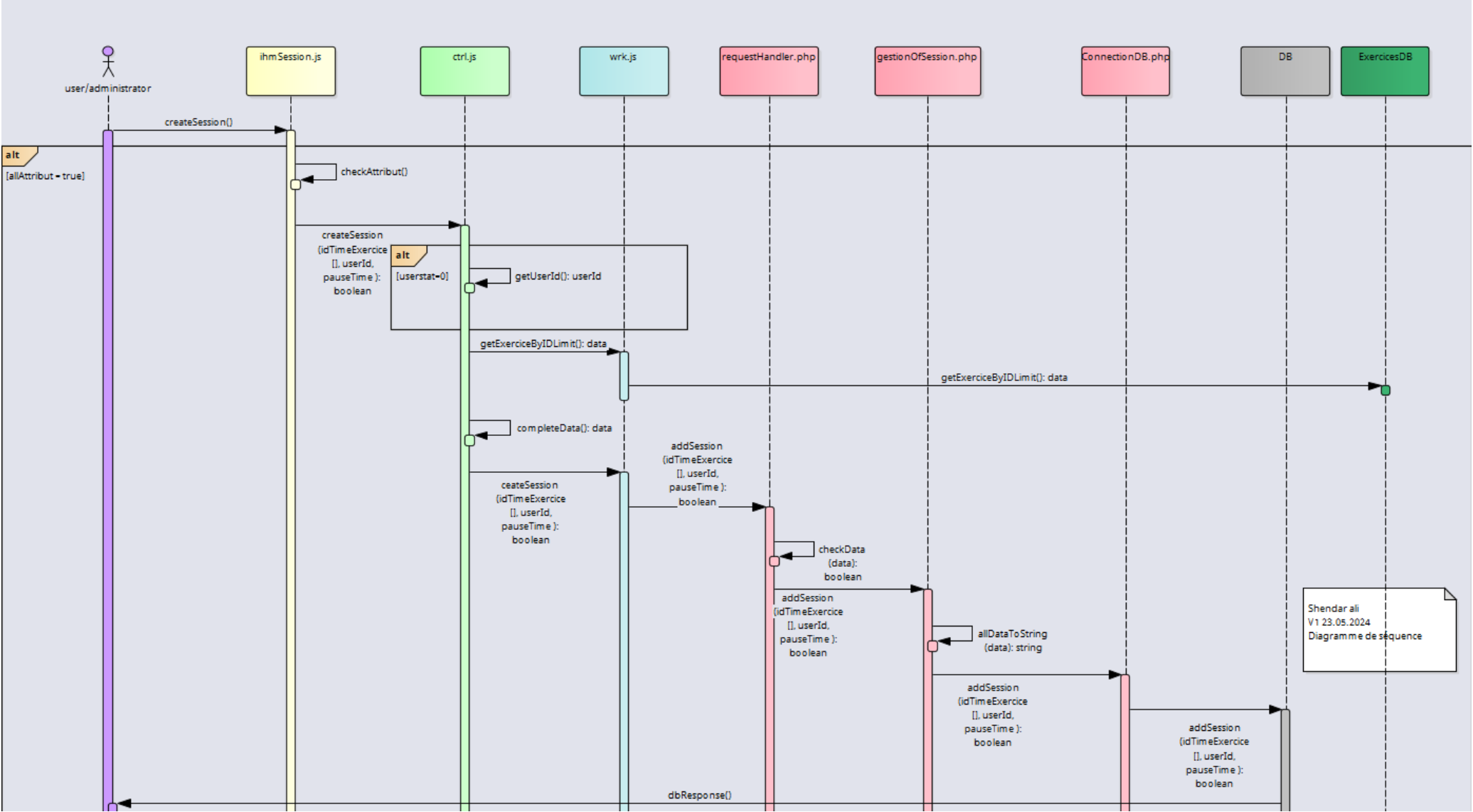


Diagramme de séquence Entreprise architecteur

Shendar ali
V1 23.05.2024
Diagramme de séquence

Créations des sessions :

Le diagramme de séquence illustre le processus de création d'une session par un utilisateur ou un administrateur. L'utilisateur initie la création de la session en appelant la méthode `createSession()`, laquelle est reçue par `ihmSession`. Ce fichier vérifie d'abord les attributs via la méthode `checkAttribut()`. Si tous les attributs sont corrects (`allAttribut = true`), `ihmSession` appelle la méthode `createSession()` de `ctrl`.

Ensuite, `ctrl` reçoit la requête et complète les données nécessaires avec la méthode `getUserId()`, en cas d'administrateur les data seront déjà complets, en cas d'utilisateur la méthode va chercher l'id d'utilisateur connecté actuellement et complète les data. Ensuite il va passer le muscle sélectionné et le nombre d'exercice souhaité aux `wrk` qui lui va se charger de chercher les data depuis l'API, et les passer au `ctrl`, qui lui se charge d'extraire les id des exercices via la méthode `completeData()`. Finalement on ajoute l'id d'user, puis on appelle `createSession()` sur `wrk`. Le fichier `wrk` reçoit la requête et procède à l'ajout de la session en appelant la méthode `addSession()` sur `requestHandler`. À ce stade, `requestHandler` vérifie les données avec `checkData()` et, si elles sont correctes, appelle la méthode `addSession()` sur `gestionOfSession`.

Le fichier `gestionOfSession` convertit toutes les données en chaîne de caractères avec `allDataToString()`. Ensuite il va passer la requête SQL déjà prête à `ConnectionDB`. Ce dernier se connecte à la base de données (DB) et ajoute la session. Une réponse (`dbResponse`) est ensuite renvoyée de la DB à `ConnectionDB`, puis transmise à `gestionOfSession`, `requestHandler`, `wrk`, `ctrl`, et enfin `ihmSession`, qui envoie la réponse finale à l'utilisateur ou à l'administrateur. Ce processus détaillé montre les interactions entre les différents composants du système pour accomplir la création d'une session, depuis la demande initiale jusqu'à l'enregistrement dans la base de données et le retour d'information à l'utilisateur.

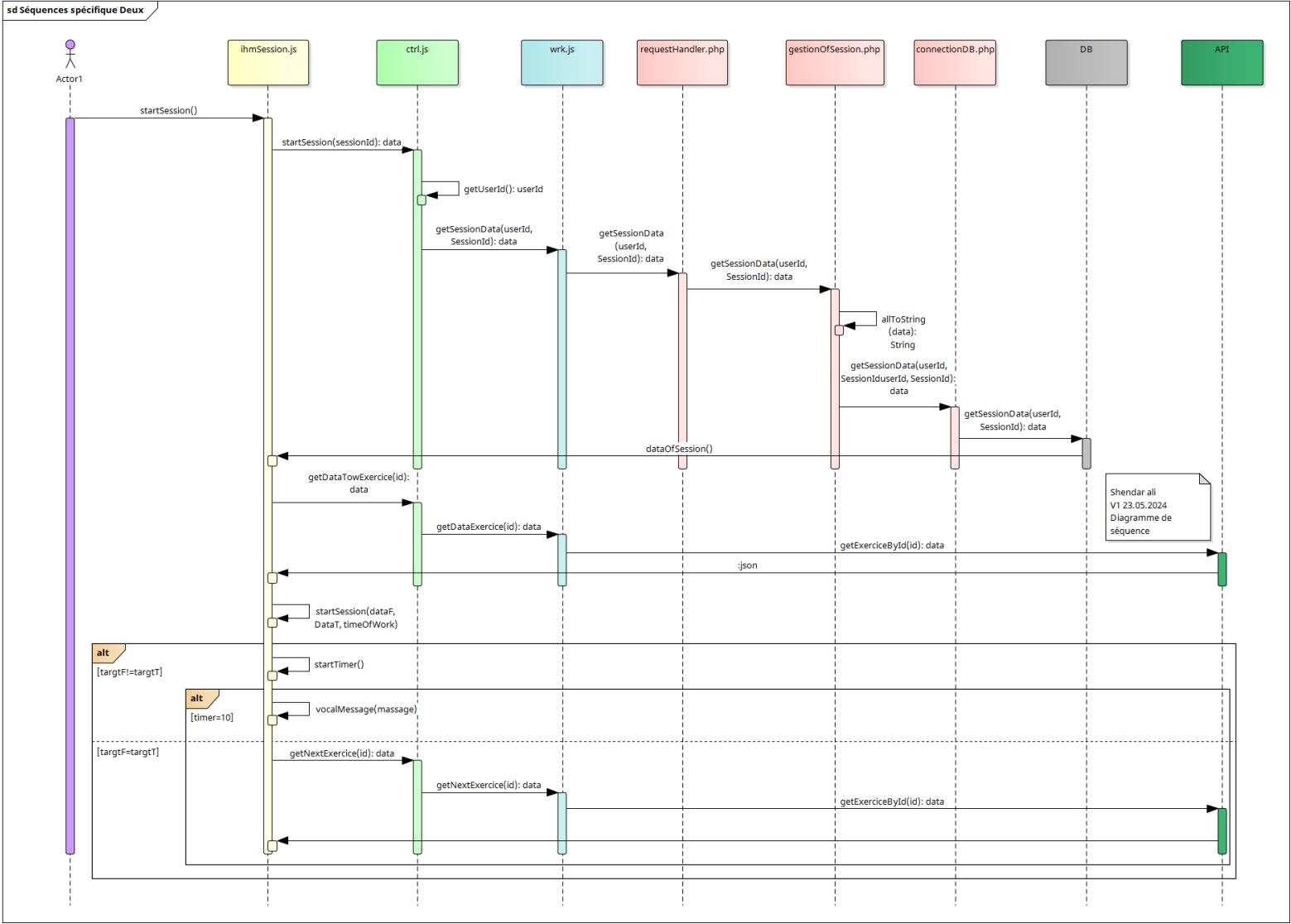


Diagramme de séquence Entreprise architecteur

Démarrer une session :

Le diagramme de séquence présente le processus de démarrage d'une session d'exercice, en illustrant les interactions entre l'acteur (utilisateur), le frontend, le backend et les composants externes tels que la base de données (DB) et l'API.

Tout commence en appelant la méthode `startSession()` dans `ihmSession`, qui à son tour appelle `startSession(sessionId)` sur `ctrl` en lui passant l'id de la session, `ctrl` va chercher l'id utilisateur `getUserId()`, finalement il va passer les data à `wrk`.

Celui-ci traite ces requêtes en obtenant les données nécessaires et en les transmettant à `requestHandler`. Ce dernier appelle `gestionOfSession` pour récupérer les données de session spécifiques en utilisant la méthode `getSessionData(userId, SessionId)`. `gestionOfSession` convertit les données en chaîne de caractères avec `allToString()` avant d'envoyer la requête à `ConnectionDB`, qui interagit avec la base de données pour récupérer les données de session.

Une fois les données récupérées, elles sont renvoyées à `gestionOfSession`, puis à `requestHandler`, `wrk`, et enfin à `ctrl`. `ctrl` appelle alors `getDataTowExercice(id)` pour obtenir les données de deux premières l'exercices, depuis l'API.

Dans le cas où les muscles des deux exercices sont différents, la session d'exercice est démarrée avec `startSession(dataT, DataT, timeOfWork)`. Un chronomètre sera déclenché avec le temps de travail du premier exercice, si la valeur du chronomètre est de 10s, la fonction `vocalMessage(message)` sera appelée pour faire l'annonce vocale. Et la fonction `getNextExercice(id)`, sera appelée, ce qui déclenchera une chaîne de requêtes similaires pour obtenir les données de l'exercice suivant.

3.5 Concept de formation

Le concept de formation pour "En Méga Forme" se concentre sur plusieurs volets essentiels pour garantir une utilisation optimale et efficace de l'application par tous les utilisateurs, qu'ils soient administrateurs ou utilisateurs.

Objectifs de la formation

- **Familiarisation avec l'interface utilisateur :** Permettre aux utilisateurs de naviguer facilement dans l'application, de se connecter, de créer et de gérer des sessions d'entraînement, et d'utiliser les fonctionnalités de suivi et de notification.
- **Gestion des utilisateurs et des sessions :** Former les administrateurs à ajouter de nouveaux utilisateurs, à promouvoir des utilisateurs au statut d'administrateur, et à générer et copier des sessions d'entraînement pour les utilisateurs.

3.6 Concept de tests

L'objectif de tests pour l'application "En Méga Forme" est de garantir la fiabilité, la performance et l'utilisabilité de l'application avant son déploiement final. Pour atteindre cet objectif il faut plusieurs types de tests, chacun visant à identifier et corriger les éventuels défauts et à vérifier que l'application répond aux exigences spécifiées. De manière générale il faut faire un mélange de test entre blackbox et whitebox, whitebox pour voir en détails ce qui se passe au moment où l'action est appelée, jusqu'à l'envoi de la requête https, et une méthode blackbox pour tester le Backend PHP, pour vérifier que le Backend répond correctement.

Un test full blackbox pour tester la sécurité d'application les attaques XSS, et la sécurité des données sensibles, telles que les informations de connexion des utilisateurs.

Backend			
Nr.	Objet testé	Description du test	Attente
1	getAllUser	Chercher tous les utilisateur	Liste JSON avec tous les utilisateur
2	getSession-Data	Avoir toutes les informations d'une session d'entraînements	Liste JSON avec toutes les informations d'une session
3	getAllSession-ByUser	Avoir toutes les sessions d'un utilisateur	Liste JSON avec le nome, temps total, nombre de fois que la session a été fait et id d'utilisateur
4	getAll-Muscles	Chercher tous les muscles possibles dans la DB	Liste JSON avec tous les muscles dans la base de données
5	getExercice-ByTarget	Chercher les exercices possibles par muscles depuis l'API	Liste JSON avec l'exercice depuis de l'API
6	getStartSession	Retourne une session avec tous les détails	Liste JSON avec les informations d'une session depuis la nôtre base de données et le nom et l'URL de gif depuis l'API
7	getFinishSession	Retourné une boolean	Une boolean pour dire que le nombre de fois de la session est incrémenter d'un
8	createSession	Crée une session avec les données que j'ai envoyé,	Un flux JSON avec l'id de la session qu'il a créée
9	updateSession	Modifier la session sélection selon les données envoyer	Retournée une boolean, si la modification a été fait ou pas
10	login	Demander si le nom d'utilisateur et mot de passe sont correct	Soit retourné une bollean false en cas d'erreur, soit un flux JSON avec message de succès, la status d'utilisateur, l'id, litre d'eau par jour et si les notification sont activé ou pas, et finalement ajouter dans la table de connexion la date et leur actuelle
11	disconnect	Déconnecter l'utilisateur et supprimée la session de connexion depuis le Backend	Message de déconnexion et index.html pour reloader la page de index
12	addUser	Ajouter un nouvel utilisateur dans la table utilisateur	Un message de succès

13	updateUserStatus	Changer le statut d'utilisateur à admin et l'inverse	Une message succès
14	notificationUser	Changer si l'utilisateur veut avoir les notifications ou pas, et ajoute le nombre des litres qu'il souhaite de boire par jour	Message de succès
15	copySession	Copier la session d'un utilisateur pour un autre	L'id de la nouvelle session
16	Session	Supprimer la session avec l'id dans la requête	Boolean avec si la session a été supprimé ou pas
17	User	Supprimer l'utilisateur avec l'id dans la requête	Boolean avec si l'utilisateur a été supprimé ou pas
18	Injection PHP	Injecter une script PHP dans la requête	Il ne faut pas que l'application n'exécute pas le scripte PHP
19	Injection JS	Injecter une scripte JS, pour change dans la JS de l'application	Il ne faut pas que l'application n'exécute pas le scripte JS
20	Injection HTML	Injecter une scripte HTML, pour change dans la HTML de l'application	Il ne faut pas que l'application n'exécute pas le scripte HTML
21	Injection SQL	Injecter une requête SQL pour supprimer une table	Il enregistre la requête dans la base de données comme une string

WhiteBox			
Nr.	Objet testé	Description du test	Attente
22	Afficher tous les utilisateurs	Il faut que le frontend chercher la liste d'utilisateur et leurs données les afficher aux admins connecter	Liste d'utilisateur avec leurs statuts et leurs dernière connexion
23	Changer le statut d'un utilisateur	En cliquant sur le check box	Il faut demander le backend de changer le statut d'utilisateur ensuite afficher le résultat dans une toast
24	Ajouter un utilisateur	Demander le backend pour ajouter un utilisateur	Il faut afficher la réponse de backend en toast
25	Supprimer un utilisateur	Demander le backend pour supprimer un utilisateur	Il faut afficher la réponse de backend en toast
26	Crée une session	Créer une session pour plusieurs utilisateurs et plusieurs muscle	Il faut que l'application demander le Backend les exercices par muscle ensuite faire la limitation par pour les nombres d'exercice demander, et envoyer la nouvelle session aux Backend, finalement afficher la réponse du Backend

27	Chercher et afficher les muscle disponible	Demander le Backend tous les muscle possible	Afficher les exercices dans le drop-down
28	Chercher toutes les sessions par utilisateur	Demander toutes les sessions par utilisateur	Afficher toutes les sessions l'utilisateur sélectionné et les afficher dans dropdown
29	Copier la session pour une autre utilisateur	Demander le Backend de copier la session chez une autre utilisateur	Afficher l'id de la nouvelle session dans une toast
30	Chercher toutes les données des sessions liées à un utilisateur	Demander le Backend d'envoyer toutes les sessions et leur donnée par utilisateur	Afficher les sessions dans une table
31	Enregistrer litre par jours	Demander le Backend d'enregistrer le nombre de litre que l'utilisateur souhaite de boire par jours	Afficher le message de succès dans une toast
32	L'activation de notification	Demander le Backend d'enregistrer si l'utilisateur souhaite avoir des notifications pour boire d'eau	Afficher le message de succès dans une toast
33	Crée une session depuis le compte d'utilisateur	Demander le Backend de crée une session	Afficher un toast avec l'id de session
34	Modifier une session déjà existante	Demander le Backend de modifier une session déjà existante	Afficher un message de succès
35	Supprimer une session	Demander le Backend de supprime une session depuis la base de données	Afficher un message de succès
36	Démarre une session	Demander le Backend toutes les données d'une session et les url de gif dedans	Démarrer une session et les chronos et afficher les deux première gif
37	Stocké tous les gifs	Il faut charge tous les gifs dans le cache de la navigateur	Dans le cache de navigateur il faut voir tout le gif de la session
38	En cas d'offline	Il faut stock les requête : getFinishSession,	Dans le cache de navigateur il faut voir toutes les requêtes, et une fois connecter il faut les envoyer

		createSession, updateSession, copySession, updateUserStatus, addUser, Dans le cache et une fois connecter il faut l'envoyer	
--	--	--	--

3.7 Moyens nécessaires

Divers moyens matériels et logiciels sont nécessaires pour garantir le bon déroulement du développement, des tests et de la mise en œuvre de l'application. Voici une description détaillée des ressources requises :

Matériel :

- Une PC de systèmes Windows en préférence
- Un smartphone de préférence Android
- Une connexion internet
- Un serveur d'hébergement

Logiciels et outils de développement :

- PhpStorm pour le développement
- Un compte gitlab
- Cpanel pour gestion de serveur
- Phpmyadmin pour gestion de base de données
- Chrome pour voir l'avancement de projet
- Postman pour tester les fonctionnements du Backend

Documentation :

- Word pour la documentation
- Excel pour le planning et le journal de travail

Réalisation

3.8 Spécifications détaillées

Pour assurer le bon fonctionnement de l'application, les prérequis suivants sont nécessaires :

- Plateformes supportées :
 - Android
 - iOS
- Navigateur recommandé :
 - Google Chrome : Il est fortement recommandé d'utiliser Google Chrome comme navigateur, car la fonctionnalité de notification est optimisée pour ce navigateur.

Ces spécifications sont essentielles pour garantir une expérience utilisateur optimale et la pleine compatibilité des fonctionnalités de l'application, en particulier la gestion des notifications.

3.9 Design du système

3.9.1 Backend

Diagramme de classe suivant représente les classes de la partie backend de l'application.

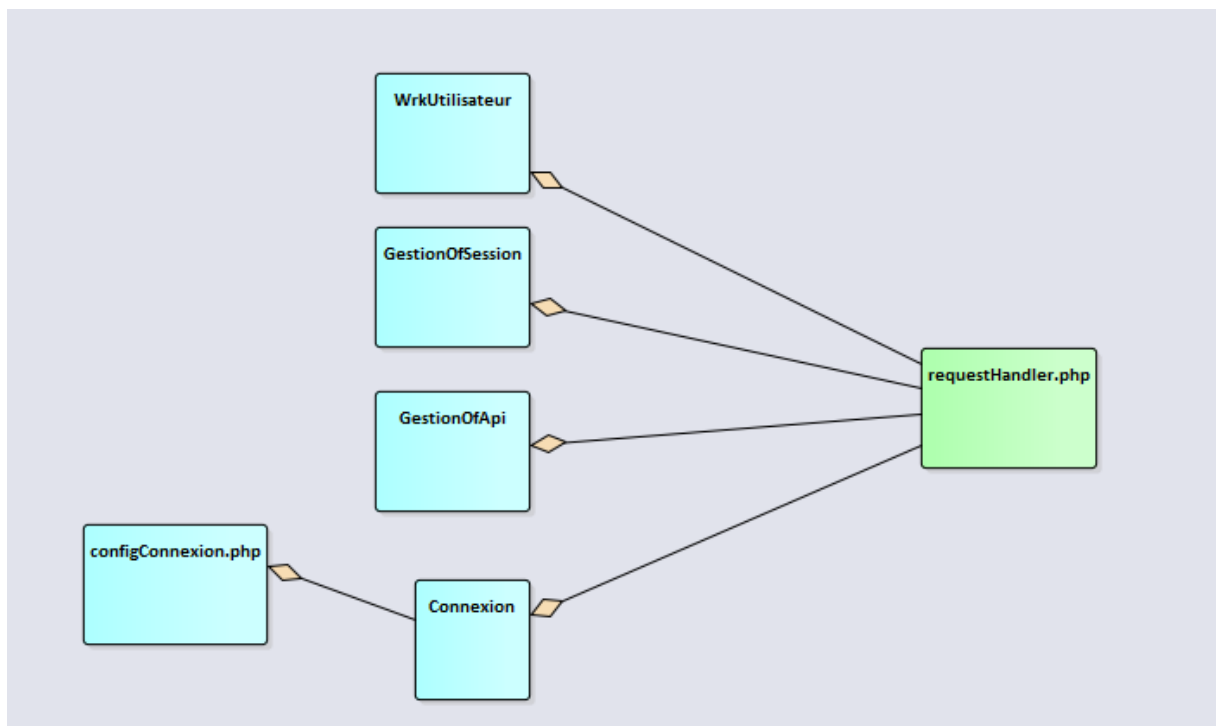


Diagramme de classe Entreprise architecteur

Le diagramme de classes ci-dessous représente la structure principale du Backend, illustrant les relations entre plusieurs classes clés : Connexion, requestHandler, WrkUtilisateur, GestionOfSession, GestionOfApi, et configConnexion. Voici une explication détaillée de chacune de ces classes et de leurs rôles :

- **requestHandler**: Ce script joue le rôle de contrôleur principal. Il reçoit les requêtes du frontend, traite les données, puis les passe aux différentes classes worker pour une gestion spécifique. Notez que requestHandler est un script PHP et non une classe, mais il orchestre les interactions entre les classes.
- **WrkUtilisateur** : Cette classe est responsable de la gestion des utilisateurs. Elle fournit des fonctionnalités pour ajouter, modifier, supprimer, et récupérer des informations utilisateur.
- **GestionOfSession** : Cette classe gère les sessions des utilisateurs. Elle inclut des méthodes pour créer, mettre à jour, supprimer et récupérer des sessions.
- **GestionOfApi** : Cette classe est utilisée pour interfacier avec diverses API externes. Elle peut inclure des méthodes pour récupérer des données spécifiques nécessaires au fonctionnement de l'application.
- **configConnexion**: Ce fichier contient les configurations nécessaires pour établir la connexion à la base de données. Il définit les paramètres de connexion comme l'hôte, le nom d'utilisateur, le mot de passe et le nom de la base de données.
- **Connexion** : Cette classe gère la connexion entre le Backend et la base de données. Elle utilise les configurations définies dans configConnexion pour établir et maintenir la connexion.

Chaque composant de ce système joue un rôle crucial pour assurer une interaction fluide et efficace entre le frontend et le backend, en traitant les données de manière sécurisée et en fournissant les fonctionnalités nécessaires aux utilisateurs de l'application.

3.9.1.1 Implémentation

Pour le Backend, il y a deux parties distinctes que l'on peut identifier : le WRK (Worker) et le CTRL (Contrôleur). Dans cette documentation, je vais me concentrer sur les deux fonctionnalités les plus importantes et spécifiques :

Création de session :

Tout commence dans requestHandler qui une fois qu'il reçoit la requête http de type PUT avec l'action createSession : commence le processus. Il faut vérifier que les paramètres nécessaires sont présents dans la requête en utilisant la méthode isset() de PHP. Si tous les paramètres requis sont présents, les données sont transmises à la méthode createSession() via la variable globale « \$gestionOfSession » qui est une instance de la classe GestionOfSession.

```
case "createSession":
    if (isset($data['name'], $data['t_temps_total'], $data['user_id'],
    $data['exercises'])) {
        try {
            $session_id = $gestionOfSession->createSession($data['name'],
            $data['t_temps_total'], $data['user_id'], $data['exercises']);
            sendResponse(['session_id' => $session_id, 'message' => 'Ses-
            sion created successfully']);
        } catch (Exception $e) {
            sendResponse(['error' => $e->getMessage()], 400);
        }
    } else {
        sendResponse(['error' => 'Missing parameters'], 400);
    }
    break;
```

Code Backend phpStorm

Une fois arrivé dans la classe GestionOfSession et dans la méthode creatSession

Toutes les données reçues dans les paramètres sont passées dans la méthode validateParameters(). Cette méthode vérifie que le nom et le temps total ne sont pas vides, que la liste des exercices n'est pas vide et que chaque exercice contient un ID d'exercice, un temps de travail et un temps de pause. Si l'une de ces données est incorrecte, une erreur est renvoyée.

```
private function validateParameters($name, $t_temps_total, $user_id, $exercises)
{
    if (empty($name) || empty($t_temps_total) || empty($user_id) || empty($exercises)) {
        throw new Exception('message: 'Tous les paramètres sont obligatoires.');
```

```
    }
    if (!is_array($exercises) || count($exercises) == 0) {
        throw new Exception('message: 'La liste des exercices doit être un tableau non vide.');
```

```
    }
    foreach ($exercises as $exercice) {
        if (empty($exercice['exercice_id']) || empty($exercice['time_of_work']) || empty($exercice['time_of_rest'])) {
            throw new Exception('message: 'Chaque exercice doit avoir un ID, un temps de travail et un temps de repos.');
```

```
        }
    }
}
```

Code Backend phpStorm

Si tous les paramètres sont correctement vérifiés, une série de requêtes SQL sera créée et exécutée :

- La première requête ajoute une nouvelle session dans la table T_Session. Ensuite, l'ID du dernier enregistrement est récupéré via la méthode getLastId.

```
// Insérer la session
$query = 'INSERT INTO T_Session (name, t_temps_total) VALUES (:name, :t_temps_total)';
$params = array('name' => $name, 't_temps_total' => $t_temps_total);
$result = $this->pdo->executeQuery($query, $params);
$session_id = $this->pdo->getLastId('table: "T_Session");
```

Code Backend phpStorm



- La deuxième requête ajoute un enregistrement dans TR_User_Session en utilisant l'ID de l'utilisateur reçu dans les paramètres et l'ID de la dernière session ajoutée.

```
// Associer l'utilisateur à la session
$query = 'INSERT INTO TR_User_Session (fk_user, fk_session) VALUES (:fk_user, :fk_session)';
$params = array('fk_user' => $user_id, 'fk_session' => $session_id);
$result = $this->pdo->executeQuery($query, $params);
```

Code Backend phpStorm

- Ensuite, une boucle itère sur chaque exercice de la liste à ajouter dans la base de données. Pour chaque exercice, il vérifie s'il existe déjà dans la base de données. Si c'est le cas, il ne fait rien ; sinon, il l'ajoute. Finalement, les exercices et leurs temps de travail et de pause sont ajoutés à la liste \$params pour la dernière requête.

```
// Préparer les valeurs pour l'insertion des exercices
$values = [];
$params = [];
foreach ($exercices as $index => $exercice) {
    $values[] = "(:exercise_id{$index}, :session_id, :time_of_work{$index}, :time_of_rest{$index})";
    $params["exercise_id{$index}"] = $exercice['exercise_id'];
    $params["time_of_work{$index}"] = $exercice['time_of_work'];
    $params["time_of_rest{$index}"] = $exercice['time_of_rest'];
}
$params['session_id'] = $session_id;
```

Code Backend phpStorm

- Enfin, après avoir ajouté chaque exercice, son temps de travail et son temps de pause à une liste, ces informations sont passées à la requête pour TR_Exercice_Session afin d'ajouter la liste des exercices et leurs paramètres à la session liée.

```
// Insérer les exercices dans la session
$query = 'INSERT INTO TR_Exercice_Session (exercise_id, session_id, time_of_work, time_of_rest) VALUES ' . implode('separator: ', $values);
$result = $this->pdo->executeQuery($query, $params);
// Valider la transaction
$this->pdo->commitTransaction();
```

Code Backend phpStorm

- La méthode commitTransaction() valide la transaction. Si tout se passe bien, l'ID de la session est retourné.

```
// Valider la transaction
$this->pdo->commitTransaction();
```

Code Backend phpStorm

- En cas de problème, une exception est levée, déclenchant un rollback de la transaction et retournant une réponse d'erreur.

```
} catch (\Exception $e) {  
    // Annuler la transaction en cas d'erreur  
    $this->pdo->rollbackTransaction();  
    throw $e;  
}
```

Code Backend phpStorm

Et voici un vison général de la méthode :



```

public function createSession($name, $t_temps_total, $user_id, $exercices)
{
    try {
        // Vérifier les paramètres
        $this->validateParameters($name, $t_temps_total, $user_id, $exercices);
        // Insérer la session
        $query = 'INSERT INTO T_Session (name, t_temps_total) VALUES (:name, :t_temps_total)';
        $params = array('name' => $name, 't_temps_total' => $t_temps_total);
        $result = $this->pdo->executeQuery($query, $params);
        $session_id = $this->pdo->getLastId( table: "T_Session");

        // Associer l'utilisateur à la session
        $query = 'INSERT INTO TR_User_Session (fk_user, fk_session) VALUES (:fk_user, :fk_session)';
        $params = array('fk_user' => $user_id, 'fk_session' => $session_id);
        $result = $this->pdo->executeQuery($query, $params);
        // Préparer les valeurs pour l'insertion des exercices
        $values = [];
        $params = [];
        foreach ($exercices as $index => $exercice) {
            $values[] = "(:exercise_id{$index}, :session_id, :time_of_work{$index}, :time_of_rest{$index})";
            $params["exercise_id{$index}"] = $exercice['exercise_id'];
            $params["time_of_work{$index}"] = $exercice['time_of_work'];
            $params["time_of_rest{$index}"] = $exercice['time_of_rest'];
        }
        $params['session_id'] = $session_id;
        // Insérer les exercices dans la session
        $query = 'INSERT INTO TR_Exercise_Session (exercise_id, session_id, time_of_work, time_of_rest) VALUES ' . implode( separator: ', ', $values);
        $result = $this->pdo->executeQuery($query, $params);
        // Valider la transaction
        $this->pdo->commitTransaction();

        return $session_id;
    } catch (\Exception $e) {
        // Annuler la transaction en cas d'erreur
        $this->pdo->rollbackTransaction();
        throw $e;
    }
}

```

Code Backend phpStorm

Supprime une session :

Pour supprimer une session, j'ai changé le type de requête HTTP à DELETE. Tout commence encore une fois avec le contrôleur requestHandler. Lorsque ce dernier reçoit une requête de type DELETE, il vérifie si celle-ci contient deux paramètres : l'objet à supprimer et l'ID de l'élément à supprimer.

<https://alis.emf-informatique.ch/test-total/server/ctrl/requestHandler.php/user/8>

Code Backend phpStorm

```
function doDelete()
{
    global $gestionOfSession, $gestionOfUtilisateur;

    // Lire les données DELETE de l'URL
    $request = explode(separator: '/', trim($_SERVER['REQUEST_URI'], characters: '/'));
    $entity = $request[count($request) - 2];
    $id = end($request);

    if (is_numeric($id)) {
        $id = (int)$id;
        try {
            switch ($entity) {
                case 'session':
                    $result = $gestionOfSession->deleteSession($id);
                    sendResponse($result);
                    break;
            }
        }
    }
}
```

Code Backend phpStorm

Il faut aussi vérifier si la deuxième entité est numérique. Si les vérifications sont réussies, les données sont envoyées à wrkSession dans la fonction deleteSession, où une série de requêtes est exécutée :

- La première requête supprime tous les enregistrements de la table TR_Exercise_Session liés à l'ID de session reçu.
- La deuxième requête supprime tous les enregistrements de la table TR_User_Session liés à l'ID de session reçu.
- Enfin, la requête supprime la session ciblée de la table T_Session.

Pour valider la transaction, la méthode commitTransaction est utilisée. Si tout se passe bien, le résultat de la dernière requête est retourné. Sinon, une rollback est effectuée.



```

public function deleteSession($session_id)
{
    try {

        // Supprimer les exercices de la session
        $query = 'DELETE FROM TR_Exercice_Session WHERE session_id = :session_id';
        $params = array('session_id' => $session_id);
        $this->pdo->executeQuery($query, $params);

        // Supprimer les associations utilisateur-session
        $query = 'DELETE FROM TR_User_Session WHERE fk_session = :session_id';
        $this->pdo->executeQuery($query, $params);

        // Supprimer la session
        $query = 'DELETE FROM T_Session WHERE pk_session = :session_id';
        $result = $this->pdo->executeQuery($query, $params);

        // Valider la transaction
        $this->pdo->commitTransaction();

        return $result;
    } catch (Exception $e) {
        // Annuler la transaction en cas d'erreur
        $this->pdo->rollbackTransaction();
        throw $e;
    }
}

```

Code Backend phpStorm

3.9.2 Frontend

Le diagramme de classe représente les classes de Frontend : couleur jaune pour IHM vert pour contrôleur et bleu pour le worker.

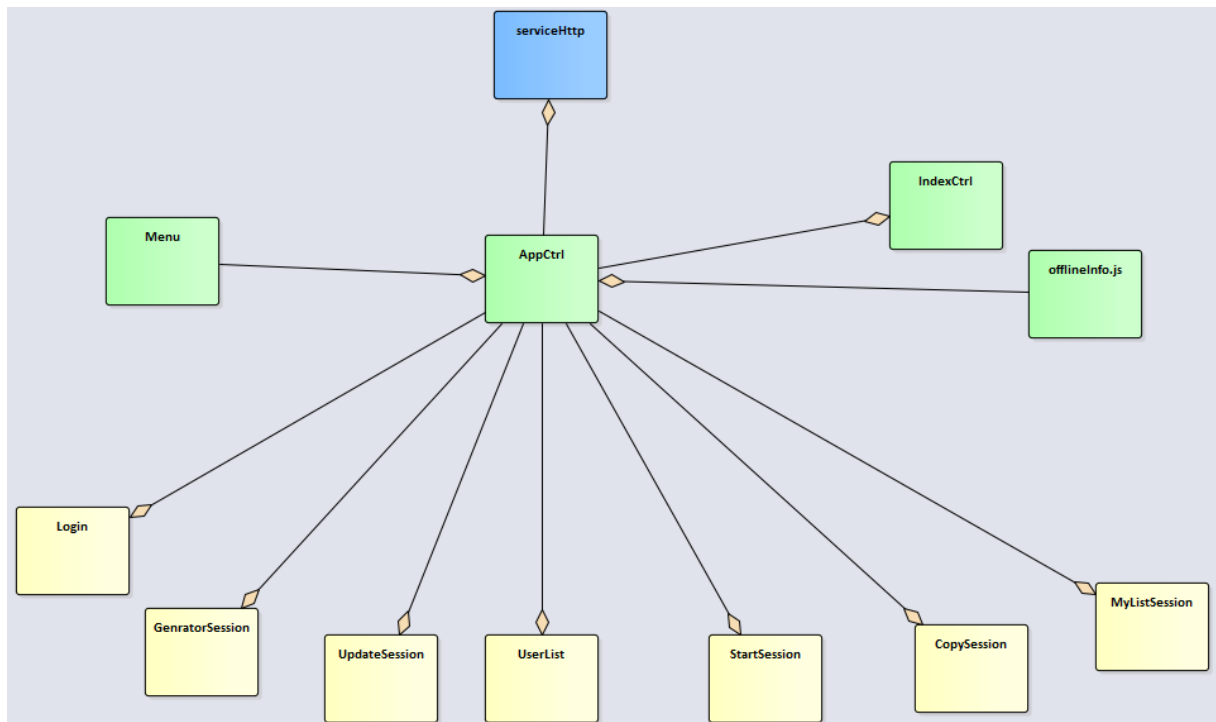


Diagramme de classe Entreprise architecteur

Ce diagramme de classes représente la structure principale du frontend, illustrant les classes clés : ServiceHttp, AppCtrl, IndexCtrl, Menu, Login, GenratorSession, UpdateSession, UserList, StartSession, CopySession et MyListSession. Voici une explication détaillée de chacune de ces classes et de leurs rôles :

- **ServiceHttp** : Cette classe gère les requêtes HTTP entre le frontend et le backend de l'application.
- **AppCtrl** : Cette classe agit comme un contrôleur central qui gère la navigation entre différentes vues et la coordination entre les différentes classes de l'application.
- **IndexCtrl** : Cette classe initialise l'application en chargeant la vue de connexion.
- **Menu** : Cette classe gère le menu de navigation de l'application, permettant de charger différentes sections de l'application.
- **Login** : Cette classe gère la connexion des utilisateurs.
- **GenratorSession** : Cette classe permet de générer de nouvelles sessions pour les utilisateurs, en sélectionnant des exercices et des utilisateurs.
- **UpdateSession** : Cette classe permet de mettre à jour les sessions existantes.
- **UserList** : Cette classe gère l'affichage et la gestion de la liste des utilisateurs.
- **StartSession** : Cette classe gère le démarrage des sessions d'entraînement.
- **CopySession** : Cette classe permet de copier des sessions d'un utilisateur à un autre.
- **MyListSession** : Cette classe gère l'affichage de la liste des sessions de l'utilisateur.

Dans cette descend de code je vais prendre la fonctionnalité principale de côté frontend c'est démarrer une session

3.9.2.1 Implémentation

Démarrer une session :

Tous commencé quand l'utilisateur choisi une session depuis sa liste de session, dans cette moment la classe startSession sera créé une dans le constructeur la fonction getSession sera appeler, qui lui va chercher les données de la session depuis le Backend et les passés aux fonction start success qui lui va distribuer les données vers la fonction concernée.

```
/**
 * Callback de succès pour la récupération des données de la session.
 * @param {Array} data - Données de la session.
 */
Show usages new *
getstartSuccess(data : (any)[] ) : void {
    if (data.length > 0) {
        this.data = data;
        this.totalTime = parseInt(this.data[0].total_time, radix: 10);
        this.startTotalTimeTimer(this.totalTime);
        this.displayExercise(this.currentExerciseIndex);
    }
}
```

Code Frontend phpStorm

Donc la fonction de timer qui sont chargé de déclencher le timer total d'entraînement avec le bon temp

```
/**
 * Démarre le timer pour le temps total de la session.
 * @param {number} duration - Durée totale en secondes.
 */
Show usages new *
startTotalTimeTimer(duration : number ) : void {
    this.remainingTotalTime = duration; // Mémoire le temps total restant
    const display : HTMLElement = document.getElementById( elementId: 'countdown');

    this.intervalTotal = setInterval( handler: () : void => {
        const minutes : number = parseInt( string: this.remainingTotalTime / 60, radix: 10);
        const seconds : number = parseInt( string: this.remainingTotalTime % 60, radix: 10);

        this.valueMMT = minutes < 10 ? "0" + minutes : minutes;
        this.valueSST = seconds < 10 ? "0" + seconds : seconds;

        display.textContent = `${this.valueMMT}:${this.valueSST}`;
        if (--this.remainingTotalTime < 0) {
            clearInterval(this.intervalTotal);
            this.toastSuccess('La session est terminée. ');
            this.sessionGetFinish();
        }
    }, this.interval);
}
```

Code Frontend phpStorm

Et la fonction displayExercise qui est chargé de chercher les gifs depuis le cache en cas que le gif n'est pas présente dans le cache il va le chercher depuis internet, une fois trouver il va les déplace dans des bons endroits. Le gif de premier exercice dans le placement un « grand gif », le deuxième dans placement deux « petite gif », et il va aussi déclencher le timer de travail « startTimer ».

startTime est la fonction chargé pour démarrer les chronomètre pour le time de travail pour l'exercice ou le time de pause, et comme je reçoit le temps en seconde je le convertir en première temps en minute et la reste de la division en seconde

```

this.remainingWorkPauseTime = duration; // Mémoire le temps de travail/pause restant
this.type = type;
const display : HTMLElement = type === 'work' ? document.getElementById( 'elementId: 'worktimespan' ) : document.getElementById( 'elementId: 'PauseTime' );

this.intervalWP = setInterval( handler : () : void => {
    const minutes : number = parseInt( string: this.remainingWorkPauseTime / 60, radix: 10 );
    const seconds : number = parseInt( string: this.remainingWorkPauseTime % 60, radix: 10 );

    this.valueWPMPT = minutes < 10 ? '0' + minutes : minutes;
    this.valueWPSST = seconds < 10 ? '0' + seconds : seconds;
}, this.interval);

```

Code Frontend phpStorm

Ensuite, il va aussi contrôler l'avancement de temps et une fois arriver à 12 secondes avant la fin pour temp de travail il va appeler la fonction de speak « je vais l'explique après », et en cas de pause avant 15 secondes de la fin de temp il va appeler la fonction speak. Dans les deux cas si le temps est plus petit que 10 et plus grand 0 il va envoyer la valeur actuelle du temps à la fonction speak.

```

if (type === "work") {
    if (this.valueWPMPT == "00" && this.valueWPSST == "12") {
        this.speak("Pause dans ");
    } else if (this.valueWPMPT == "00" && this.valueWPSST < "10" && this.valueWPSST > "00" ) {
        let nbr : number = this.valueWPSST
        this.speak(parseInt(nbr));
    }
} else if (type === "rest") {
    if (this.valueWPMPT == "00" && this.valueWPSST == "15") {
        this.speak(`Exercice suivant ${this.nextExercise.name} dans`);
    } else if (this.valueWPMPT == "00" && this.valueWPSST < "10" && this.valueWPSST > "00" ) {
        let nbr : number = this.valueWPSST

        this.speak(parseInt(nbr));
    }
}
}

```

Code Frontend phpStorm

Et une fois le temps est fini il va supprimer l'intervalle et switcher entre la valeur de type de temp de travail à temp de pause et l'inverse.

```

if (--this.remainingWorkPauseTime < 0) {
    clearInterval(this.intervalWP);
    if (type === 'work') {
        this.handleWorkComplete();
    } else if (type === 'rest') {
        this.handleRestComplete();
    }
}
}, this.interval);

```

Code Frontend phpStorm

La fonction speak est la fonction chargée de faire le textToSpeech il prendre le message depuis le paramètre et le son déjà sélection dans le cas de « En Méga Forme » : 'Microsoft

Hortense - French (France)', ensuite va créer un nouvel objet SpeechSynthesisUtterance, Finalement il a passé les deux informations à cet objet pour que l'application puisse prononcer vocalement l'information de message

```
/**
 * Lit un message à l'aide de la synthèse vocale.
 * @param {string} message - Message à lire.
 */
Show usages new *
speak(message) : void {
    let selectedVoiceName : string = 'Microsoft Hortense - French (France)';
    let toSpeak : SpeechSynthesisUtterance = new SpeechSynthesisUtterance(message);

    this.voices.forEach((voice) : void => {
        if (voice.name === selectedVoiceName) {
            toSpeak.voice = voice;
        }
    });

    this.synth.speak(toSpeak);
}
```

Code Frontend phpStorm

3.9.3 Problème

Appelle de l'API :

Une problème que j'ai découvert en codant la partie frontend c'est été que je ai l'appeler à l'API depuis le frontend pour gagner en vitesse mais la problème qui se posé c'était que je stocké la TOKEN de l'API de côté Frontend ce qui n'est pas une bon manière pour la partie sécurité donc j'ai dû déplacer l'appelle de là l'API vers le Backend et changé la logique, donc actuellement la création ou l'update ou la chercher des information de une exercice passe tous le temps depuis le Backend, ensuite traité depuis le Frontend, finalement envoyer vers le Backend pour les enregistrer dans la base de données,

Pourquoi une partie de calcul se fait depuis le frontend ? parce que quand j'ai déplacé tous vers le backend ça prend trop du temps donc j'ai devisé pour qu'il soit le plus rapide possible.

Donc actuellement la création des sessions se fait comme ça :

Shendar ali



La relation entre la table session et user :

J'ai constaté que la liaison entre la table T_User et T_session « N à N » n'était pas trop correct car quand l'admin duplique copy la session d'un utilisateur pour un autre se qui se passé cette que l'application ajouter un autre enregistrement dans la table TR_User_Session. Donc la session resté la même, cette méthodologie pose le problème que si un utilisateur modifie dans la session la session est modifier pour tout le monde :

Exemple : si l'utilisateur zana et l'utilisateur colella ont une session de biceps en commune et le temp total de la session est 6 minutes et que l'utilisateur colella change le temps pour 12 minute l'utilisateur zana sera effectuer aussi et sa session de biceps deviendra aussi 12 minutes ce qui lui ne veut pas.

La solution est de faire une liaison directe entre les deux tables pour avec une liaison que une session doit avoir une utilisateur et utilisateur n'est pas obligé d'avoir une session se qui veut dire que quand l'administrateur copy une session la session est publiques dans la table de session et donc les modification d'un utilisateur ne peut pas touche l'autre.

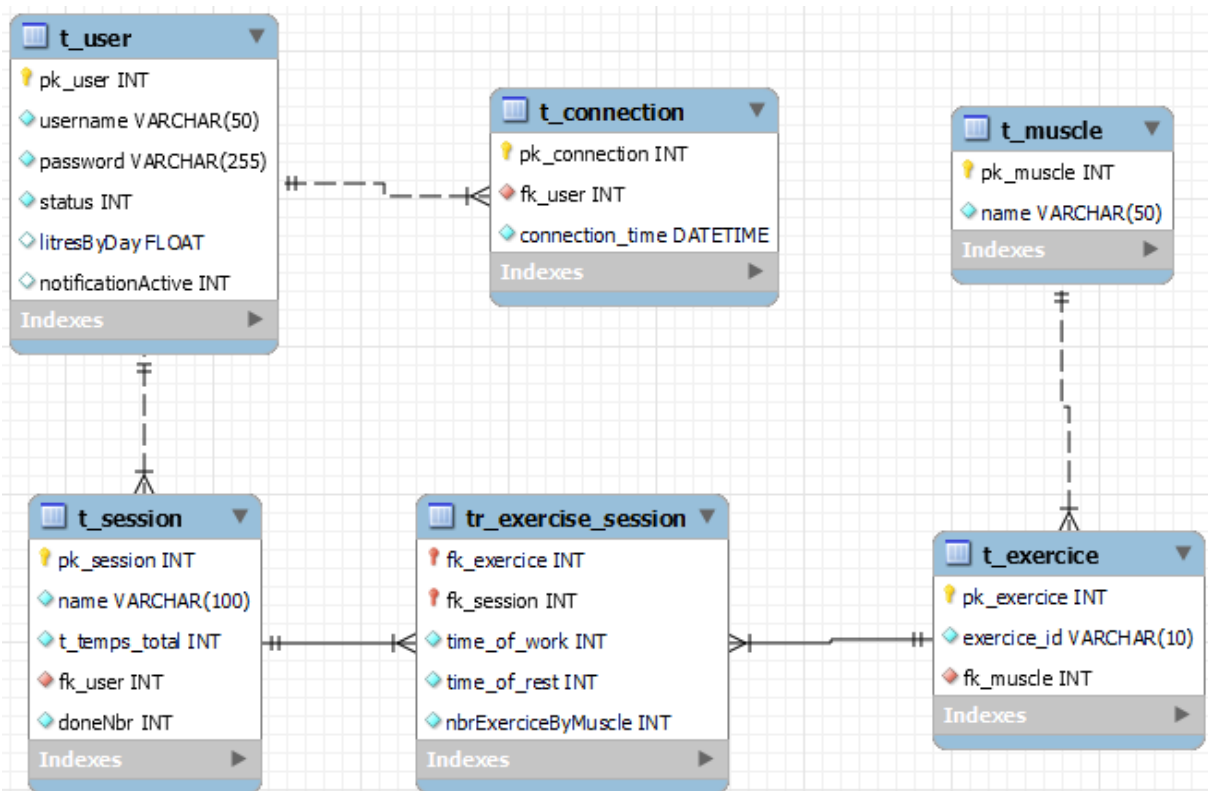


Schéma de base de données MSQWorkbench

3.10 Configuration

Dans le cadre de ce projet plusieurs technologie ont été utilisé avec des version et des dé-taille qu'il faut bien respecter, les technologies :

- PHP : version 7.3
- JQuery : version 3.7.0

- Bootstrap : version 5.3.3
- Sweetalert2 : 11
- datatables : 1.11.5
- bootstrap-icons : 1.9.1
- popper 1.16.0
- MYSQL :5.7

4 Test

4.1 Procédure de test

Les tests seront faits sur deux méthodologies :

- **Blackbox** : pour le backend via Postman pour tester toutes les fonctionnalités possibles et les tests de sécurité contre les injections.
- **Whitebox** : pour tester la partie frontend pour voir ce qui se passe à l'entraîneur de l'application et si le flux respecte bien le modèle MVC2, et tester la sécurité de l'application via le navigateur Chrome.

Accès :

<https://alis.emf-informatique.ch/test-total/>

Username	Mot de passe	Rôle
admin	Emf12345	Administrateur
colella	Emf12345	Utilisateur
Denervaud	Emf12345	Utilisateur
Hans	Emf12345	Utilisateur

4.2 Protocol de test

Dans ce protocole de test, il faut tester toutes les fonctionnalités de l'application ainsi que la sécurité. Tous les tests possibles doivent être effectués sans en ignorer aucun.

Blackbox					
Nr.	Objet testé	Description du test	Attente	Résultat	Visa
1	getAllUser	Chercher tous les utilisateurs	Liste JSON avec tous les utilisateurs	OK	SA
2	getSession-Data	Avoir toutes les informations d'une session d'entraînements	Liste JSON avec toutes les informations d'une session	OK	SA
3	getAllSession-ByUser	Avoir toutes les sessions d'un utilisateur	Liste JSON avec le nom, temps total, nombre de fois que la session a été faite et id d'utilisateur	OK	SA



4	getAll-Muscles	Chercher tous les muscles possibles dans la DB	Liste JSON avec tous les muscles dans la base de données	OK	SA
5	getExercice-ByTarget	Chercher les exercices possibles par muscles depuis l'API	Liste JSON avec l'exercice depuis de l'API	OK	SA
6	getStartSession	Retourne une session avec tous les détails	Liste JSON avec les informations d'une session depuis la base de données et le nom et l'URL de gif depuis l'API	OK	SA
7	getFinishSession	Retourné une boolean	Une boolean pour dire que le nombre de fois de la session est incrémenter d'un	OK	SA
8	createSession	Crée une session avec les données que j'ai envoyé,	Un flux JSON avec l'id de la session qu'il a créée	OK	SA
9	updateSession	Modifier la session sélection selon les données envoyer	Retournée une boolean, si la modification a été fait ou pas	OK	SA
10	login	Demander si le nom d'utilisateur et mot de passe sont correct	Soit retourné une boolean false en cas d'erreur, soit un flux JSON avec message de succès, la status d'utilisateur, l'id, litre d'eau par jour et si les notification sont activé ou pas, et finalement ajouter dans la table de connexion la date et leur actuelle	OK	SA
11	disconnect	Déconnecter l'utilisateur et supprimée la session de connexion depuis le Backend	Message de déconnexion et index.html pour reloader la page de index	OK	SA
12	addUser	Ajouter un nouvel utilisateur dans la table utilisateur	Un message de succès	OK	SA
13	updateUserStatus	Changer le statut d'utilisateur à admin et l'inverse	Une message succès	OK	SA
14	notificationUser	Changer si l'utilisateur veut avoir les notifications ou pas, et ajoute le nombre des litres	Message de succès	OK	SA



		qu'il souhaite de boire par jour			
15	copySession	Copier la session d'un utilisateur pour un autre	L'id de la nouvelle session	OK	SA
16	Session	Supprimé la session avec l'id dans la requête	Boolean avec si la session a été supprimé ou pas	OK	SA
17	User	Supprimé l'utilisateur avec l'id dans la requête	Boolean avec si l'utilisateur a été supprimé ou pas	OK	SA
18	Injection PHP	Injecter une script PHP dans la requête	Il ne faut pas que l'application n'exécute pas le scripte PHP	OK	SA
19	Injection JS	Injecter une scripte JS, pour change dans la JS de l'application	Il ne faut pas que l'application n'exécute pas le scripte JS	OK	SA
20	Injection HTML	Injecter une scripte HTML, pour change dans la HTML de l'application	Il ne faut pas que l'application n'exécute pas le scripte HTML	OK	SA
21	Injection SQL	Injecter une requête SQL pour supprimer une table	Il enregistre la requête dans la base de données comme une string	OK	SA

WhiteBox					
Nr.	Objet testé	Description du test	Attente	Résultat	Visa
22	Afficher tous les utilisateurs	Il faut que le frontend cherche la liste d'utilisateur et leurs données les afficher aux admins connecter	Liste d'utilisateur avec leurs statuts et leurs dernière connexion	Affiche dans la table d'utilisateur tous les utilisateurs	OK-SA
23	Changer le statut d'un utilisateur	En cliquant sur le check box	Il faut demander le backend de changer le statut d'utilisateur ensuite afficher le résultat dans une toast	Changement dans la table d'utilisateur et un toast	OK-SA
24	Ajouter un utilisateur	Demander le backend pour ajouter un utilisateur	Il faut afficher la réponse de backend en toast	Afficher l'utilisateur dans la table et un toast	OK-SA
25	Supprimer un utilisateur	Demander le backend pour supprimer un utilisateur	Il faut afficher la réponse de backend en toast	Supprimer l'utilisateur dans la table et un toast	OK-SA



26	Crée une session	Créer une session pour plusieurs utilisateurs et plusieurs muscle	Il faut que l'application demander le Backend les exercices par muscle, ensuite faire la limitation par pour les nombres d'exercice demander, et envoyer la nouvelle session à le Backend, finalement afficher la réponse du Backend	Un toast avec la réponse de Backend. L'utilisateur peut voir la nouvelle session dans son liste	OK-SA
27	Chercher et afficher les muscle disponible	Demander le Backend tous les muscle possible	Afficher les exercices dans le dropdown	Dans la dropdown voir ^les muscles disponible	OK-SA
28	Chercher toutes les sessions par utilisateur	Demander toutes les sessions par utilisateur	Afficher toutes les sessions l'utilisateur sélectionné et les afficher dans dropdown	Afficher le nom des sessions dans la dropdown	OK-SA
29	Copier la session pour une autre utilisateur	Demander le Backend de copier la session chez une autre utilisateur	Afficher l'id de la nouvelle session dans une toast	Un toast avec l'id de la nouvelle session	OK-SA
30	Chercher toutes les données des sessions liées à un utilisateur	Demander le Backend d'envoyer toutes les sessions et leurs données par utilisateur	Afficher les sessions dans une table	Afficher toutes les sessions de l'utilisateur dans la table	OK-SA
31	Enregistrer litre par jours	Demander le Backend d'enregistre le Backend de litre que l'utilisateur souhaite de boire par jours	Afficher le message de succès dans une toast	Un toast avec la réponse de Backend	OK-SA
32	L'activation de notification	Demander le Backend d'enregistre si l'utilisateur souhaite avoir des notifications pour boire d'eau	Afficher le message de succès dans une toast	Un toast avec la réponse de Backend	OK-SA
33	Crée une session depuis le compte d'utilisateur	Demander le Backend de crée une session	Afficher un toast avec l'id de session	Un toast avec l'id de la nouvelle session, et la nouvelle liste est ajouter dans la liste des sessions	OK-SA



34	Modifier une session déjà existante	Demander le Backend de modifier une session déjà existante	Afficher un message de succès	Un toast avec la réponse de Backend et si le temps de la session est changé l'utilisateur doit le voir dans la liste de session	OK-SA
35	Supprimer une session	Demander le Backend de supprimer une session depuis la base de données	Afficher un message de succès	Un toast avec la réponse du Backend et dans la liste des sessions la session est supprimée	OK-SA
36	Démarrer une session	Demander le Backend toutes les données d'une session et les url de gif dedans	Démarrer une session et les chronos et afficher les deux premières gif	Changer la page vers la page de démarrage de session et le chrono total est commencé et les gifs sont affichés et le chrono de le premier exercice est démarré	OK-SA
37	Stocké tous les gifs	Il faut charger tous les gifs dans le cache de la navigateur	Dans le cache de navigateur il faut voir tout le gif de la session	OK	OK-SA
38	En cas d'offline	Il faut stocker les requêtes : getFinishSession, createSession, updateSession, copySession, updateUserStatus, addUser, Dans le cache et une fois connecté il faut l'envoyer	Dans le cache de navigateur il faut voir toutes les requêtes, et une fois connecté il faut les envoyer	Rien ne va se passer jusqu'à la connexion et une fois en ligne va afficher le toast avec la réponse de Backend	OK-SA

Conclusion :


Les tests effectués sur l'application de coaching personnel "En Méga Forme" montrent que celle-ci répond aux exigences spécifiées dans le cahier des charges. Elle est fonctionnelle, performante et sécurisée. Aucune amélioration majeure n'est nécessaire à ce stade.

Recommandations :

- Continuer les tests de performance pour des charges encore plus élevées pour garantir la scalabilité.

- Effectuer des tests de sécurité par quelqu'un externes

4.3 Signature du protocole de test

Date	Nom	Signature
03.06.2024	Shendar ali	

5 Conclusion

L'objectif global du projet a été atteint avec succès. Toutes les étapes prévues ont été exécutées conformément au plan initial. L'application est pleinement fonctionnelle et répond aux objectifs spécifiés dans le cahier des charges, le tout dans les délais impartis. En résumé, le projet est achevé de manière satisfaisante et répond aux attentes établies.

Personnellement, je suis très satisfait du travail accompli. Ce projet m'a permis de développer de nouvelles compétences techniques, notamment en gestion de projet agile. J'ai également apprécié les défis rencontrés et la manière dont ils ont été surmontés, ce qui a renforcé ma capacité à résoudre des problèmes complexes. Toutefois, je reconnais qu'il y a toujours des possibilités d'amélioration, et je suis ouvert aux feedbacks pour continuer à m'améliorer dans mes futurs projets.

5.1 Améliorations possibles

L'amélioration graphique de l'application est essentielle pour améliorer l'expérience utilisateur. Voici quelques idées concrètes :

- Mise en place de GIFs en fond d'écran : Intégrer des GIFs en fond d'écran pour chaque exercice pourrait rendre l'application plus dynamique et engageante.
- Thèmes et Personnalisation : Ajouter des options de thèmes et de personnalisation afin que les utilisateurs puissent modifier l'apparence de l'application selon leurs préférences.

Achat d'une Licence pour l'API :

Actuellement, l'application utilise une API token gratuit pour récupérer les données d'exercice, mais elle est limitée par des quotas de requêtes mensuelles. Acheter une licence pour cette API présente plusieurs avantages :

- Suppression des Limites de Requête : Une licence payante éliminerait les restrictions sur le nombre de requêtes, assurant une disponibilité constante des données pour tous les utilisateurs.

Calcul des Calories Dépensées :

Ajouter une méthodologie pour calculer les calories dépensées par session est une fonctionnalité qui pourrait grandement enrichir l'application. Voici comment cela pourrait être implémenté :

- Collecte de Données Utilisateur : Demander aux utilisateurs des informations nécessaires telles que l'âge, le poids, le sexe, et le niveau d'activité physique. Ces données sont essentielles pour des calculs précis.
- Algorithmes de Calcul des Calories : Utiliser des formules établies pour estimer les calories brûlées. Par exemple, des formules comme la formule de Harris-Benedict ou l'équation de Mifflin-St Jeor pourraient être utilisées pour calculer le métabolisme de base, puis ajustées en fonction de l'intensité et de la durée de l'exercice.

- **Intégration avec les Sessions** : Intégrer ces calculs dans les sessions d'exercice. Par exemple, chaque type d'exercice pourrait avoir un coefficient de dépense calorique, et l'application pourrait multiplier ce coefficient par le temps passé sur l'exercice et les données de l'utilisateur pour obtenir une estimation des calories brûlées.

Autres Améliorations :

- **Notifications en Temps Réel** : Ajouter des notifications en temps réel pour rappeler aux utilisateurs de commencer leurs séances, prendre des pauses ou atteindre leurs objectifs quotidiens.
- **Rapports et Statistiques** : Créer des tableaux de bord de statistiques pour que les utilisateurs puissent suivre leur progression au fil du temps. Cela pourrait inclure des graphiques sur les calories brûlées, les séances terminées, et les améliorations de performance.
- **Intégration Sociale** : Permettre aux utilisateurs de partager leurs progrès et leurs séances sur les réseaux sociaux pour augmenter la motivation et l'engagement.
- **Support Multilingue** : Ajouter des options multilingues pour rendre l'application accessible à un public plus large.
- **Une token par utilisateur** : pour des raisons de sécurité et de rapidité, cette solution est idéale pour garantir une interaction plus rapide avec l'API. De plus, chaque utilisateur disposera de son propre token, ce qui permet au développeur d'identifier facilement la source de tout problème éventuel en traçant le jeton concerné.
- **Offline notification** : Il est de stocker localement le nombre de litres à boire par jour et la quantité déjà consommée. Ainsi, même si l'utilisateur est hors ligne, il continuera à recevoir des notifications pour l'encourager à rester hydraté.

En mettant en œuvre ces améliorations, l'application non seulement répondra mieux aux besoins actuels des utilisateurs.

5.2 Auto-évaluation

Aujourd'hui, je ressens une grande satisfaction et un profond sentiment d'accomplissement. J'ai consacré 80 heures à ce projet, durant lesquelles j'ai mobilisé toutes mes capacités. En examinant le travail réalisé au cours de cette période, je suis fier de mes réalisations.

Je tiens à exprimer ma gratitude envers monsieur Colella, monsieur Denervaud et monsieur Wilhelm Schwendimann pour leur soutien inestimable. Leur aide a été cruciale pour le succès de ce projet. Un autre élément clé de cette réussite a été notre planification rigoureuse. Bien que le planning ait été ambitieux, la méthodologie efficace et les compétences enseignées par l'école m'ont permis de gérer efficacement cette charge de travail intense.

En somme, cette expérience a non seulement été enrichissante sur le plan professionnel, mais elle m'a également permis de développer des compétences précieuses en gestion de projet et en résolution de problèmes. Je suis reconnaissant pour l'opportunité de travailler sur ce projet et pour tout ce que j'ai appris au cours de ce processus.

6 Bibliographie: liste des sources et références

Nome de la sources	Cadre d'utilisation	Lien
What web can do	Pour l'utilisation des fonctionnalités des smart-phones	https://whatwebcando.today/
Bootstrap	Pour les éléments visuels de partie Frontend	https://getbootstrap.com/docs/5.0/getting-started/introduction/
Wikipedia	Pour le glossaire	https://fr.wikipedia.org/
ChatGPT	Pour la correction orthographe de la documentation et pour aide pour le débuge	https://chatgpt.com/

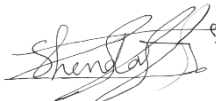
7 Glossaire

Terme	Signification
PWA	Progressive Web App, application web progressive.
Bootstrap	Bootstrap (framework) est une collection d'outils utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web.
Apache2	Un serveur linux c'est une serveur http qui est capable d'exécuté de scripte PHP
MySQL	MySQL : est un système de gestion de bases de données relationnelles
XSS	Injection de code malveillant, injection SQL, JS, HTML, et CSS
Worker	C'est le type de classe qui fait les calculs et le communication soit avec la base de données soit avec le Backend en cas de frontend
Contrôleur	C'est le type de classe plus proche de surface qui lui a une roll de les données resu de la frontend en cas de Backend et IHM en cas de frontend
HTTP	Protocole de communication internet « Hypertext Transfer Protocol »
PUT	Cette méthode de http permet de remplacer ou d'ajouter une ressource sur le Backend. L'URI fourni est celui de la ressource en question.
isset()	Méthode de PHP pour détermine si une variable est déclarée et est différente de null
DELETE	Cette méthode de http permet de supprimer. L'URI fourni est celui de la ressource en question.
HTML	Le HyperText Markup Language : langage de programmation pour afficher les éléments visuels de site web ou application ...etc
CSS	Cascading Style Sheets, langage de programmation pour styliser les éléments visuels des sites web.
JavaScript (JS)	Langage de programmation utilisé pour créer des contenus web interactifs.
MySQL	Système de gestion de bases de données relationnelles.
API	Une backend externe

UML	Unified Modeling Language, langage de modélisation utilisé pour la conception de systèmes.
GitLab	Plateforme de gestion de dépôts Git, offrant des outils de DevOps.
FTP	File Transfer Protocol, protocole de transfert de fichiers entre un client et un serveur sur un réseau informatique.
OneDrive	Service de stockage en ligne de Microsoft.
VSCode	Visual Studio Code, éditeur de code source développé par Microsoft.
Enterprise Architect	Outil de modélisation pour la conception de logiciels.
FileZilla	Client FTP libre et open-source pour le transfert de fichiers.
SQL	Structured Query Language, langage de programmation utilisé pour gérer et manipuler des bases de données.
Sweetalert2	Librairie JavaScript pour créer des pop-ups interactives et élégantes.
Postman	Outil utilisé pour effectuer des tests des backend
datatables	Plugin jQuery utilisé pour créer des tables dynamiques avec des fonctionnalités avancées telles que la pagination, le tri et la recherche.
jQuery	Bibliothèque JavaScript rapide et concise qui simplifie la manipulation de documents HTML, la gestion d'événements, l'animation et les interactions Ajax.
bootstrap-icons	Une bibliothèque d'icônes vectorielles libres conçue pour fonctionner avec Bootstrap.
popper	Bibliothèque JavaScript utilisée pour gérer les pop-ups et les info-bulles dans les interfaces web.

8 Signatures

Je soussigné déclare que les informations contenues dans ce rapport de travail pratique individuel rendu ce jour le 04.06.2024 dans le cadre de la procédure de qualification de mon CFC d'informaticien, ne sont pas plagiées. Toutes les informations de sources extérieures ainsi que les informations fournies par des tiers durant le déroulement du travail sont consignées.

Date	Nom	Signature
04.06.2024	Shendar ali	

9 Annexes

• Nom	• Placement
Base de données	/Base de données/alis_en_Mega_forme.sql
Schéma de base de données workbanch	/Base de données/schéma.mwb
Maquette admin	/Maquette/copie de session.pdf
Maquette admin	/Maquette/crée une session admin.pdf
Maquette admin	/Maquette/liste d'utilisateur.pdf
Maquette admin	/Maquette/login.pdf
Maquette utilisateur	/Maquette/crée un session.pdf
Maquette utilisateur	/Maquette/liste de session.pdf
Maquette utilisateur	/Maquette/login.pdf
Maquette utilisateur	/Maquette/session en cours.pdf
Planning	/Planning+Journal/Planning_EnMega-Forme_148444_ALI.xlsx
Journal	/Planning+Journal/Journal_EnMega-Forme_148444_ALI.xlsx
Projet Frontend	/Projet/client/
Projet Backend	/Projet/server/
Diagramme use case	/UML/use case/
Diagramme séquence	/UML/sequence/
Diagramme classe	/UML/classe/
Diagramme d'activité	/UML/activite/