HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

My amazing fitness

Départements : TIC

Unité d'enseignement : BDR

Auteurs : Valentin Bugna, Theodros Mulugeta et Massimo Stefani

Professeur : Nastaran Fatemi

Assistant : Christopher Meier

Date : 21.01.2024

Présentation du projet

Avant de plonger dans les détails du projet, veuillez noter que les instructions complètes pour le déploiement et l'utilisation du système sont fournies dans le fichier "readme.txt" accompagnant ce document.

Le projet vise à créer une base de données pour une salle de fitness, avec pour objectif d'améliorer l'expérience des utilisateurs et d'optimiser la gestion administrative liée à l'exploitation de la salle de fitness. Les fonctionnalités majeures comprennent :

Inscription et Gestion des Membres : Implémentation d'un formulaire d'inscription et d'un système de connexion avec gestion de cookies.

Gestion des abonnements : Suivi et gestion des abonnements des membres.

Gestion des Paiements : Intégration d'un système de paiement avec historique des transactions.

Planification des Cours : Intégration d'un calendrier interactif.

Statistique des Passages : Outil de suivi de la progression pour les membres.

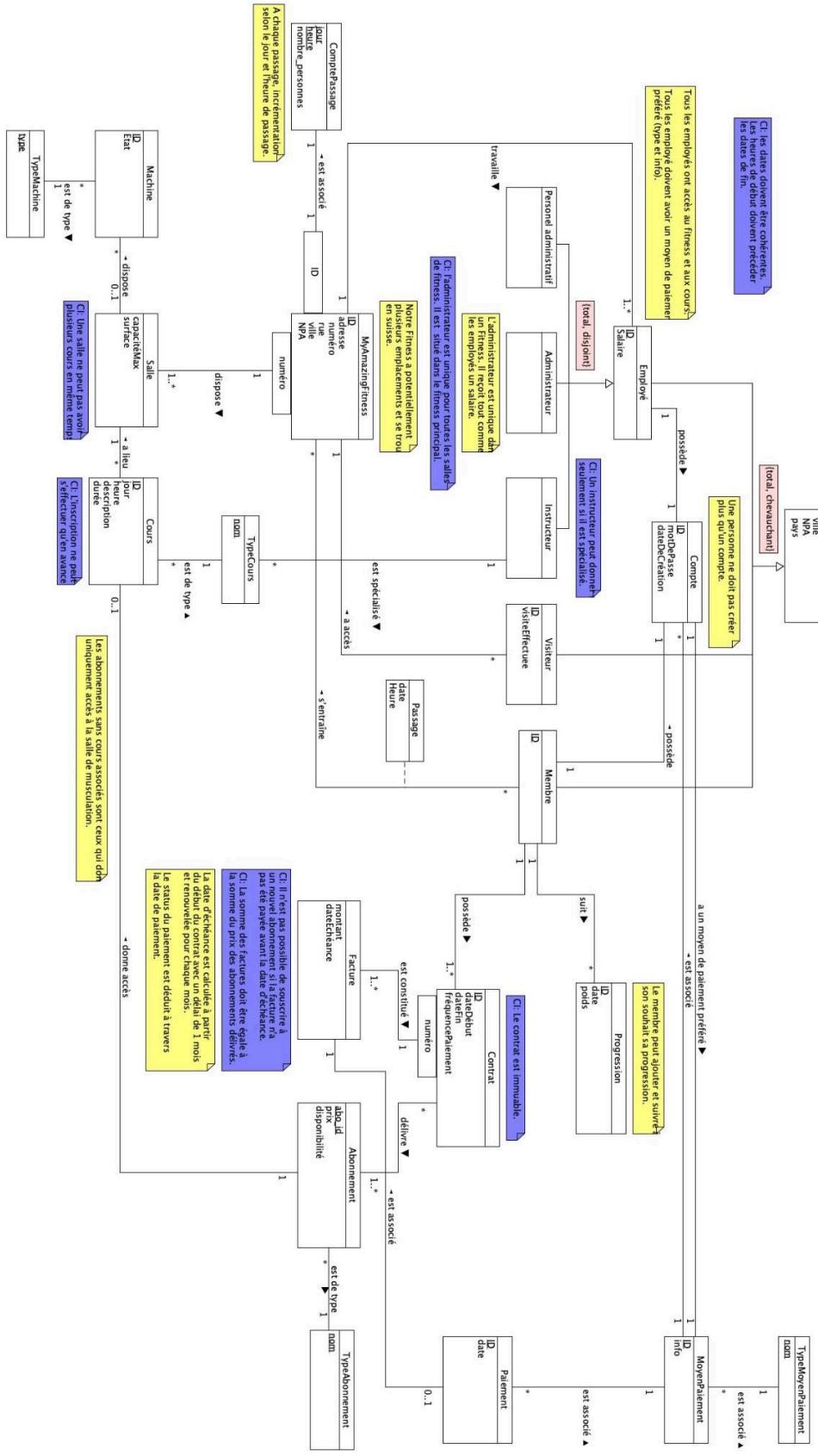
Administrateur : Création, mise à jour, et coordination des cours sur le site web.

Le cahier des charges complet est annexé pour une référence détaillée.

Le développement du projet se déroule dans l'environnement Intellij IDEA, avec l'utilisation de la conteneurisation Docker. La configuration de la connexion pour l'utilisateur est définie avec les paramètres suivants : localhost comme hôte, le port 5432, la base de données "bdr", et les identifiants utilisateur bdr avec le mot de passe bdr.

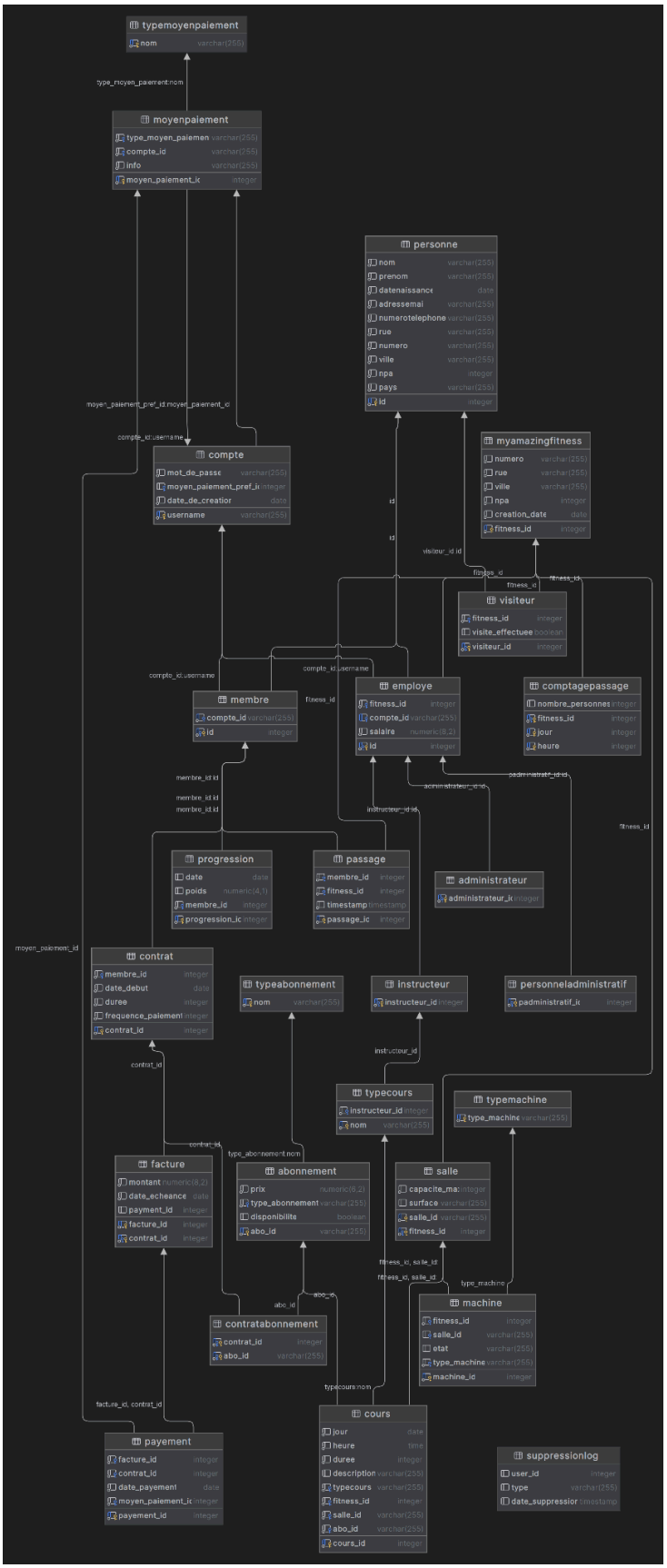
Dans la page suivante, le schéma UML de la base de données mise à jour, accompagné des contraintes d'intégrité et des notes nécessaires garantissant la cohérence des données, sera présenté. Cette représentation visuelle offrira une vue d'ensemble claire de la structure de la base de données et de l'interaction entre les différentes entités. Explorez le schéma UML pour mieux comprendre l'organisation du système et les relations entre ses composants.

Modélisation conceptuelle (mise à jour)



HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Modélisation relationnelle (mise à jour)



Enumération des Vues dans l'Application

AccountView : combine toutes les informations des comptes (Membre, Employé, Personnel Administratif, Administrateur, Instructeur) avec leurs détails personnels.

CourseWeekView : affiche tous les cours de la semaine avec leurs détails, y compris la date, l'heure, la description, l'instructeur, le type de cours, le centre de fitness, la salle, et l'abonnement associé.

MemberCourseWeekView : présente tous les cours de la semaine avec des détails spécifiques aux membres, incluant l'identifiant du membre associé.

MembreAbonnementView : donne un aperçu des membres et de leurs contrats et abonnements, incluant l'identifiant du membre, l'identifiant du contrat, la date de début et de fin du contrat, l'identifiant de l'abonnement, le prix, et le type d'abonnement.

MembreFactureView : affiche les membres et leurs factures, comprenant l'identifiant du membre, l'identifiant de l'abonnement, l'identifiant de la facture, le montant, la date d'échéance, et l'identifiant du paiement.

IntructeurTypeCoursView : présente les instructeurs et leurs types de cours, avec le nom complet de l'instructeur et l'identifiant du compte associé.

MoyennePersonnesParHeureCeJourDeSemaineView : donne la moyenne des personnes par heure pour chaque centre de fitness, en fonction du jour de la semaine.

HoraireCoursView : offre un aperçu de l'horaire des cours, organisé par jour de la semaine, avec le type de cours, l'instructeur, et la salle associés.

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Enumération des Triggers dans l'Application

create_account_trigger_membre : Trigger avant insertion d'un nouveau membre pour créer un compte associé.

create_account_trigger_employe : Trigger avant insertion d'un nouvel employé pour créer un compte associé.

create_factures_trigger : Trigger après insertion d'un nouveau contrat d'abonnement pour créer les factures associées.

increment_comptage_passage_trigger : Trigger après insertion d'un nouveau passage pour incrémenter le comptage de personnes dans le fitness center.

suppression_trigger_membre : Trigger après suppression d'un membre pour enregistrer les informations dans les journaux d'enregistrement.

suppression_trigger_employe : Trigger après suppression d'un employé pour enregistrer les informations dans les journaux d'enregistrement.

cours_overlapping : Ce Trigger vise à contrôler si les horaires des cours se chevauchent dans la même salle à la même date.

Mise en oeuvre

1. Package db

En ce qui concerne la connexion à la base de données, deux implémentations ont été mises à disposition. Une approche avec Hibernate et une autre avec le DriverManager.

La classe SQLManager joue un rôle fondamental dans la gestion des connexions à la base de données avec le DriverManager. Ses principales démarches incluent :

Établissement de la connexion : La classe SQLManager est responsable de l'établissement de la connexion à la base de données. Elle utilise les paramètres de connexion, tels que le nom d'utilisateur, le mot de passe et l'URL de la base de données, pour créer une connexion avec le système de gestion de base de données (SGBD) sous-jacent.

Exécution des requêtes SQL : Une fois la connexion établie, la classe SQLManager permet l'exécution de requêtes SQL. Elle offre des méthodes pour effectuer des opérations telles que l'insertion, la mise à jour, la suppression et la sélection de données.

Gestion des Ressources : La classe gère également les ressources liées à la connexion, garantissant la fermeture appropriée de la connexion après son utilisation pour éviter les fuites de ressources.

2. Package entity

Le package englobe des classes, dont la classe Abonnement, qui représentent des entités persistantes dans le contexte d'une application Java utilisant la Java Persistence API (JPA). Le but de ces entités est d'empêcher une exécution de l'application si notre base de données n'est pas cohérente avec notre application.

Par exemple, la classe "Abonnement" est annotée avec "@Entity", ce qui indique qu'elle est stockée dans une base de données. Elle est associée à une table nommée "Abonnement". Les attributs de cette entité sont mappés sur des colonnes de la table, avec une clé primaire grâce à la notation @Id.

3. Package structure

Le package englobe la classe Account, qui agrège diverses informations provenant de différentes tables de la base de données. De plus, dans l'optique de simplifier l'utilisation des vues et des tables, d'autres classes ont été créées dans ce package, principalement dans un rôle d'énumération.

4. Package controller

Les classes Controller agissent comme des middlewares entre l'interface utilisateur et la base de données. Leurs principales démarches comprennent :

Gestion des requêtes utilisateur : Par exemple, la classe GeneralController intercepte les requêtes provenant de l'interface utilisateur. Elle valide et traite ces requêtes avant de les transmettre à la classe SQLManager pour exécution.

Traitement des réponses de la BDD : Par exemple, après l'exécution des requêtes par la classe SQLManager, GeneralController traite les résultats et prépare les données pour une présentation adéquate à l'interface utilisateur.

5. Package form

Le package de notre application Java contient la classe "Field", conçue pour gérer les champs de formulaire de manière structurée. L'objectif est de simplifier la manipulation des formulaires en regroupant les informations associées aux champs.

6. Package view

Le package rassemble plusieurs classes qui ont pour fonction de récupérer les informations de la base de données et de stocker les vues définies dans le fichier .sql associé au système. Ces vues servent de composants essentiels pour des requêtes complexes. Les vues sont utilisées pour agréger des données, fournir des résultats spécifiques, et simplifier la complexité des requêtes.

7. Package web

La classe CookieManager offre des fonctionnalités de gestion des cookies pour les applications web basées sur Jakarta Servlet. Ses principales responsabilités incluent :

Authentification par Cookies : Les méthode isLogged permet de déterminer si l'utilisateur doit se connecter ou s'il est déjà connecté en vérifiant la présence des cookies "username" dans la requête HTTP.

Génération et Gestion des Cookies de Connexion : Les méthodes setCookie sont responsables de la création de cookies de connexion avec le nom, la valeur, et la durée de vie spécifiés. Ces cookies sont utilisés pour indiquer l'état de connexion de l'utilisateur au serveur. La méthode deleteCookie permet de supprimer ces cookies lorsqu'ils ne sont plus nécessaires.

Récupération de Cookies : La méthode getCookie facilite la récupération d'un cookie spécifique à partir de la requête HTTP.

8. Package freemarker

Le package contient une classe nommée "FreeMarkerConfig" qui a pour objectif de configurer et initialiser l'environnement FreeMarker. Ce dernier est un moteur de modèle Java qui permet de générer des contenus dynamiques, tels que des pages HTML, à partir de modèles.

9. Package pages

Le package englobe plusieurs classes conçues pour gérer les requêtes doPost() et doGet() dans le contexte d'une application web basée sur Jakarta Servlet. Ces classes représentent les pages accessibles sur le serveur local (localhost) et jouent un rôle crucial dans l'interaction entre l'utilisateur et le système. Voici une vue d'ensemble des responsabilités de ce package :

Gestion des requêtes HTTP : Chaque classe dans le package pages est responsable de la gestion des requêtes HTTP spécifiques, distinguant entre les requêtes POST et GET. Cela permet de traiter les formulaires soumis (POST) et les requêtes d'information (GET) de manière appropriée.

Logique de page spécifique : Chaque classe encapsule la logique spécifique à une page particulière de l'application. Cela peut inclure la manipulation de données, l'appel de méthodes de la couche de contrôle, et la préparation des réponses à renvoyer au client.

Présentation des Données : Les classes du package sont responsables de la préparation des données pour l'affichage dans les pages web correspondantes. Cela peut inclure la récupération d'informations depuis la base de données et la transmission de ces données aux pages pour être présentées à l'utilisateur.

10. Package api

Le package est dédié à la gestion des interfaces de programmation d'application. Ces API constituent des points d'entrée permettant d'effectuer des requêtes, notamment des requêtes HTTP de type POST. Cela permet l'envoi de données depuis le client vers le serveur, souvent utilisé pour la création ou la modification de ressources.

11. Package components

Le package components se consacre à la gestion des composants web. Ces composants sont des éléments HTML personnalisés et réutilisables qui contribuent à la création d'interfaces graphiques.

Fonctionnalités du Package :

Gestion des templates HTML : Le package intègre un mécanisme pour récupérer des templates HTML. Ces templates servent de base à la création des composants web.

Création de balises HTML personnalisées : Les composants web permettent de créer des balises HTML personnalisées. Cela offre une modularité et une réutilisabilité accrues dans le développement de l'interface utilisateur.

Parsing des données : Une fois les templates récupérés, le package assure le parsing des données associées aux composants. Cette étape est cruciale pour dynamiser le contenu des balises HTML.

Front-End : Les composants web sont principalement liés au front-end de l'application. Ils contribuent à l'amélioration de l'expérience utilisateur en permettant la création de widgets interactifs.

Vue d'ensemble sur les fonctionnalités

Sur le port 8080 du localhost, la plateforme offre une gamme complète de fonctionnalités permettant de gérer la salle de fitness.

En tant qu'utilisateur, vous avez la possibilité de procéder à un enregistrement facilité par un formulaire d'inscription.

En tant qu'administrateur, les capacités sont encore plus vastes, vous permettant d'effectuer des modifications cruciales. Vous avez le pouvoir de créer des cours, définir des plannings d'entraînement, et intégrer de nouvelles machines pour un environnement d'entraînement optimal. De plus, en tant qu'administrateur, vous pouvez effectuer des mises à jour sur les comptes des membres, garantissant une gestion précise des informations. Enfin, vous avez la flexibilité de supprimer des éléments, que ce soit des cours, des machines, ou des comptes, offrant ainsi un contrôle complet sur la configuration et la maintenance de la salle de fitness via cette interface locale.

Limitations et Points Non Implémentés

Notre mise en œuvre a été élaborée de manière à autoriser l'accès à /myAccount ou /myAccountAdmin uniquement aux membres et aux administrateurs. Faute de temps pour l'implémentation, les autres employés ne pourront pas y accéder.

En conclusion, la plateforme locale sur le port 8080 offre une solution complète et conviviale pour la gestion optimale de la salle de fitness, offrant aux utilisateurs et administrateurs un contrôle total sur leurs interactions et configurations.

Critique

L'application Web que nous avons conçue n'exploite pas toutes les fonctionnalités que peut offrir l'implémentation de la base de données. La structure de la base de données, étant assez robuste et flexible pourrait réellement être exploitée par une compagnie désirant créer un fitness.

Nous avons cherché à mettre en œuvre différentes technologies pour accéder à notre base de données afin d'explorer les diverses possibilités. Que ce soit en utilisant le modèle API ou un modèle plus "traditionnel", tout en travaillant avec le DriverManager, d'une part, ou en optant pour des solutions comme Hibernate, d'autre part, nous avons rencontré certains défis constants, tandis que d'autres ont évolué grâce à l'aide de différents frameworks. Cependant, ce qui joue un rôle capital, c'est la structure de notre base de données, car elle détermine la facilité avec laquelle nous pouvons gérer efficacement les données.