133 – Réaliser des applications web en session-handing

Documentation de projet

Version 1 du mars 2025

Da Rocha Ribeiro, Simões & Blanchard

|  |  |
| --- | --- |
|  | Documentation de projet du 17.03.2025 au 15.04.2025 |

Table des matières

[1 Introduction et contexte du project 4](#_Toc194304381)

[1.1.1 Test Hello word 4](#_Toc194304382)

[2 Analyse 5](#_Toc194304383)

[2.1 Use case client et use case Rest 5](#_Toc194304384)

[2.1.1 Client 5](#_Toc194304385)

[2.1.2 Admin 6](#_Toc194304386)

[2.1.3 Entreprise 7](#_Toc194304387)

[2.2 Activity Diagram d'un cas complet navigant dans les applications avec les explications 10](#_Toc194304388)

[2.2.1 Client 10](#_Toc194304389)

[2.2.2 Admin 10](#_Toc194304390)

[2.2.3 Entreprise 11](#_Toc194304391)

[2.3 Maquettes du projet avec la navigation intégrée et ses explications 14](#_Toc194304392)

[2.4 Class Diagram complet avec les explications de chaque application 20](#_Toc194304393)

[2.5 Sequence System global entre les applications 20](#_Toc194304394)

[2.5.1 Entreprise 20](#_Toc194304395)

[2.5.2 Admin 21](#_Toc194304396)

[2.5.3 Client 23](#_Toc194304397)

[3 Conception à faire complétement avec EA -> à rendre uniquement le fichier EA 24](#_Toc194304398)

[3.1 Navigation Diagram complète avec les explications des applications 24](#_Toc194304399)

[3.2 Diagrammes de classes 25](#_Toc194304400)

[3.2.1 APIGateway 25](#_Toc194304401)

[3.2.2 Client 26](#_Toc194304402)

[3.2.3 Entreprise 27](#_Toc194304403)

[3.2.4 Admin 28](#_Toc194304404)

[4 Bases de données 29](#_Toc194304405)

[4.1 Modèles ER 29](#_Toc194304406)

[4.2 Modèles WorkBench MySQL 30](#_Toc194304407)

[5 Implémentation des applications <Le client Ap1> et <Le client Ap2> 31](#_Toc194304408)

[5.1 Une descente de code client 31](#_Toc194304409)

[6 Implémentation de l'application <API Gateway> 32](#_Toc194304410)

[6.1 Une descente de code APIGateway 32](#_Toc194304411)

[7 Implémentation des applications <API élève1> et <API élève2> 33](#_Toc194304412)

[7.1 Une descente de code de l'API REST 33](#_Toc194304413)

[8 Hébergement 34](#_Toc194304414)

[9 Installation du projet complet avec les 5 applications 35](#_Toc194304415)

[10 Outils, langages (versions, définitions, installations) 36](#_Toc194304416)

[11 Tests de fonctionnement du projet 37](#_Toc194304417)

[12 Auto-évaluations et conclusions 38](#_Toc194304418)

[12.1 Auto-évaluation et conclusion de Blanchard 38](#_Toc194304419)

[12.2 Auto-évaluation et conclusion de Simões 38](#_Toc194304420)

[12.3 Auto-évaluation et conclusion de Ribeiro 38](#_Toc194304421)

# Introduction et contexte du project

### Test Hello word

Faire le clonage avec la commande suivante et ouvrir le dossier :

git clone https://github.com/spring-guides/gs-rest-service.git

# Analyse

## Use case client et use case Rest

### Client

Définit les différentes actions et fonctionnalités réalisables par un visiteur ainsi que le changement de statut. L’application offre donc les options suivantes en tant que simple client de l’application :

* **Login** : permet de s’identifier
* **Register** : permet de s’enregister
* **Logout** : permet de se déconnecter
* **Ajouter au panier** : permet d’ajouter un menu au panier
* **Valider au panier** : permet de terminer sa commande

De plus en fond sans besoin d’actions spécifiques du client, l’application va également s’occuper de réaliser les actions :

* **Afficher menus** : permet d’afficher les menus disponibles
* **Afficher panier** : permet de mettre à jours dynamiquement le panier

Une image contenant texte, diagramme, capture d’écran, ligne

Le contenu généré par l’IA peut être incorrect.

L’APIGateway va ensuite s’occuper de distribuer au bon endroit les requêtes puis interroger la base de données ou y ajouter des entrés en fonction des actions.

Une image contenant texte, diagramme, capture d’écran, ligne

Le contenu généré par l’IA peut être incorrect.

### Admin

Le diagramme présente l'acteur principal Admin, qui interagit avec l'application à travers un acteur secondaire nommé APIGateway. Chaque interaction sera ensuite transmise à l’APIGateway qui agit comme le point d’entrée vers nos services backend. Elle centralise les requêtes, puis les redirige vers les services appropriés pour la traitée.

Les actions possibles pour l'admin sont les suivantes :

* **Show users** : Visualiser la liste des utilisateurs enregistrés.
* **Show Entreprises** : Afficher la liste des entreprises existantes dans le système.
* **Add User** : Ajouter un nouvel utilisateur au système.
* **Delete User** : Supprimer un utilisateur existant.
* **Modify User** : Modifier les informations d’un utilisateur.
* **Add Entreprise** : Ajouter une nouvelle entreprise dans la base de données.
* **Modify Entreprise** : Modifier les informations d’une entreprise existante.
* **Delete Entreprise** : Supprimer une entreprise existante.
* **Order food** : Commander de la nourriture.
* **Logout** : Se déconnecter de la plateforme.

Une image contenant texte, diagramme, ligne, capture d’écran

Le contenu généré par l’IA peut être incorrect.Une image contenant diagramme, ligne, capture d’écran, dessin

Le contenu généré par l’IA peut être incorrect.

### Entreprise

Ce Use case client coté entreprise décrit les différentes interactions que l’utilisateur peut faire depuis l’application coté entreprise. Chaque interaction sera ensuite transmise à l’APIGateway qui agit comme le point d’entrée vers nos services backend. Elle centralise les requêtes, puis les redirige vers les services appropriés pour la traitée.

L’acteur User peut effectuer les actions suivantes :

* ShowListMenu : permet d’afficher la liste des menus disponibles.
* AddMenu : permet d’ajouter un nouveau menu.
* ModifiyMenu : permet de modifier un menu existant.
* DeleteMenu : permet de supprimer un menu.
* Logout : permet de se déconnecter de la session.

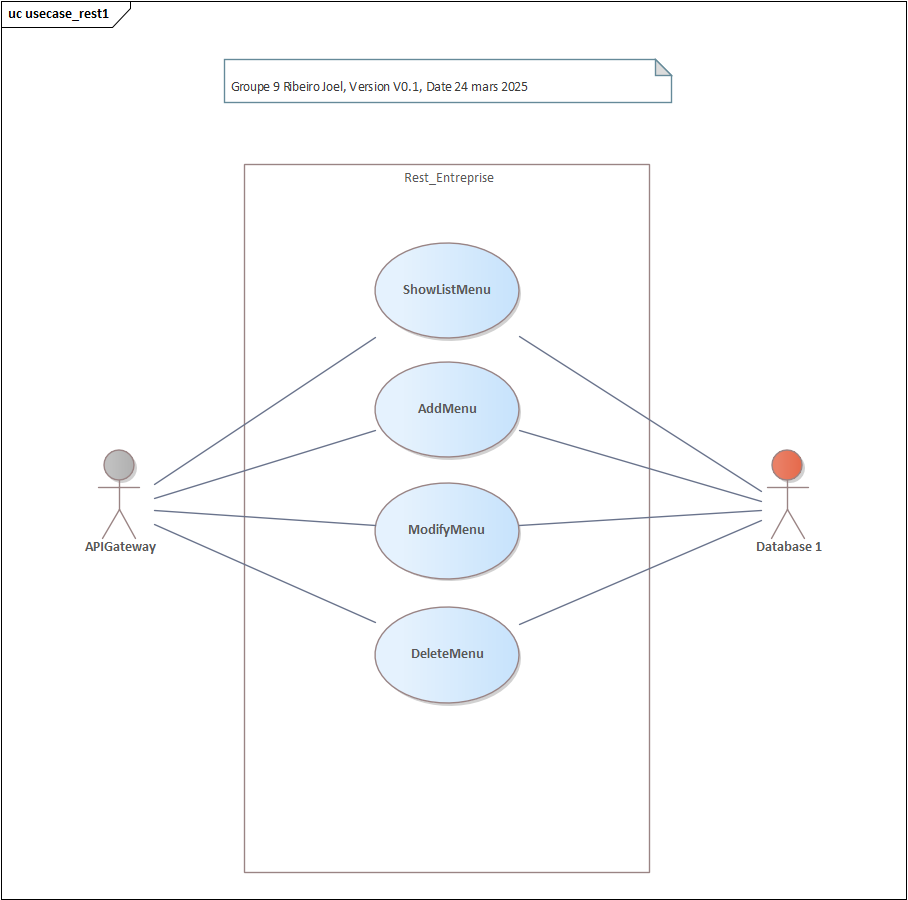
Une image contenant capture d’écran, diagramme, texte, ligne

Le contenu généré par l’IA peut être incorrect.

Ce Use case Rest coté entreprise représente les différents traitements des requêtes faite depuis l’application coté client. Chaque traitement sera ensuite acheminé à la Database qui exécutera les opérations correspondantes.

L’acteur APIGateway peut traiter les actions suivantes :

* ShowListMenu : permet de récupérer la liste des menus disponibles.
* AddMenu : permet d’insérer un nouveau menu.
* ModifiyMenu : permet de mettre à jour un menu existant.
* DeleteMenu : permet de supprimer un menu.



## Activity Diagram d'un cas complet navigant dans les applications avec les explications

### Client

Représente le déroulement de l’action register en détaillant comment se déroule la liaison entre client et la base de données.

Voici donc la suite d’actions allant se réaliser :

* Visiteur clique sur le bouton s’enregister
* Le client envoye ses infos à l’APIGateway
* L’APIGateway envoye les infos au bon API Rest
* L’API Rest récupère les infos et vérifie si le client n’existe pas à la base de données
* La base de données envoie la réponse à la requête à l’API Rest
* En fonction de la réponse l’API Rest crée ou non l’utilisateur puis envoie un message d’état à l’APIGateway
* L’APIGateway envoie ce message au client
* Le client affiche l’état du message

Une image contenant texte, capture d’écran, diagramme, Tracé

Le contenu généré par l’IA peut être incorrect.

### Admin

L'administrateur clique sur « supprimer ». Cette action déclenche une requête vers l'APIGateway, qui transmet à l'APIAdmin. Celle-ci traite la demande et envoie une requête à la base de données pour effectuer la suppression. Si l'utilisateur ou l'entreprise est supprimé avec succès, la liste mise à jour est affichée à l'administrateur. En cas d'erreur, le processus s’arrête avec un message d'erreur.

Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

Voici une description courte en texte de ton diagramme d'activité (Modification utilisateur) :

L'administrateur clique sur « modifier ». L'action déclenche une requête vers l'APIGateway, qui la transmet à l'APIAdmin. L'APIAdmin récupère les nouvelles informations de l'utilisateur et envoie la requête à la base de données pour effectuer la modification. Si la modification réussit, la liste actualisée est affichée à l'administrateur. En cas d'échec, une erreur est générée, arrêtant le processus.

Une image contenant texte, capture d’écran, diagramme, Rectangle

Le contenu généré par l’IA peut être incorrect.

### Entreprise

Ce diagramme d’activité représente le déroulement complet de l’action ShowListMenu, depuis l’initiation par le client coté entreprise jusqu’au traitement final et la réponse, en passant par les différentes couches du backend.

* + Client

L’activité débute au niveau du Client (l’utilisateur de l’application, côté entreprise). Il initie l’action avec :

* ask list menu : le client envoie une requête pour consulter la liste des menus.
  + APIGateway

L’APIGateway reçoit cette demande et agit comme point d’entrée vers les services backend. Il transmet la requête à la couche métier :

* send forward request : il redirige la requête vers l’API Rest.
  + API Rest

L’API Rest est responsable de la logique métier. Il traite la demande en interrogeant la base de données :

* request menu from database : l’API REST envoie une requête à la Database pour récupérer les menus.
  + Database

La Database exécute la requête SQL pour récupérer les menus disponibles :

* Sélectionne les menus : elle recherche tous les menus existants.
* Retourne la liste des menus : elle transmet les résultats à l’API Rest.
  + Vérification côté API Rest

L’API Rest reçoit la réponse de la base de données et vérifie son contenu :

Si la liste est vide :

* L’API REST retourne un message d'information : "liste vide".
* L’APIGateway reçoit ce message et le transmet au Client.
* Le Client affiche le message "liste vide".

Si la liste contient des menus :

* L’API REST retourne la liste des menus à l’APIGateway.
* L’APIGateway transmet la liste au Client.
* Le Client peut alors afficher les menus récupérés.

Une image contenant texte, capture d’écran, diagramme, Parallèle

Le contenu généré par l’IA peut être incorrect.

Ce diagramme d’activité représente le déroulement complet de l’action AddMenu, depuis l’initiation par le client coté entreprise jusqu’au traitement final et la réponse, en passant par les différentes couches du backend.

* + Client

L’utilisateur commence en saisissant les informations du nouveau menu :

* introduce menu information
  + Deux validations sont effectuées côté client :

information valid?

* Si non valide → afficher un message d’erreur.
* Si valide → continuer.

client valid?

* Si non valide → afficher un message d’erreur.
* Si valide → envoyer les informations à l’API Gateway.
  + APIGateway
* send menu info : transmet les données au composant suivant.
* forward menu info : envoie les informations à l’API Rest.
  + API Rest

request add menu to database : envoie la demande à la base de données.

* + Database

menu exist?

* Si oui (le menu existe déjà) :
* retourne un message d’erreur.
* L’API Rest transfère le message à l’API Gateway, qui le transmet au client.
* Le client affiche le message d’erreur.
* Si non (le menu n’existe pas) :
* le menu est ajouté dans la base.
* un message de succès est renvoyé à l’API Rest, puis à l’API Gateway.
* l’API Gateway envoie le message au client.
* le client affiche le message de succès et appelle l’action ShowListMenu pour rafraîchir la liste.

Une image contenant texte, capture d’écran, diagramme, nombre

Le contenu généré par l’IA peut être incorrect.

## Maquettes du projet avec la navigation intégrée et ses explications

ACCUEIL :

Dans cette maquette, on se trouve dans l’acceuil. Il y a le lien pour l’accueil, le menus, le panier, le register et le login. Il y a le contenu du panier le total du prix du panier. On peut pas faire grand-chose ici, il est recommander de son connecter pour accéder au fonctionnalité du projet.

Une image contenant texte, capture d’écran, conception

Le contenu généré par l’IA peut être incorrect.

MENUS :

Dans cette maquette, on peut voir la liste des menus disponibles. Il y a le lien pour l’accueil, le menus, le panier, le register et le login. On peut ajouter ou retirer une quantité de menu que l’on peut voir avec le nombre de menu sélectionner. Il y a le prix unitaire qui est afficher avec l’image.

Une image contenant texte, capture d’écran, conception

Le contenu généré par l’IA peut être incorrect.

REGISTER CLIENT :

Dans cette maquette, on peut voir le register. On doit insérer dans le champ de texte nom un nom, dans le prénom, un prénom, dans le username, un nom d’utilisateur, dans le password un mot de passe et on remet le mot de passe dans le champ re-type password. Puis on valide le register avec continuer.

Une image contenant texte, capture d’écran, reçu

Le contenu généré par l’IA peut être incorrect.

LOGIN CLIENT :

Dans cette maquette, on peut voir le login. On doit insérer dans le champ de texte username un nom d’utilisateur et dans le password un mot de passe. Puis on valide le login avec continuer, si le login est fonctionnel, on nous redirige sur la vue en fonction du status de l’utilisateur.

Une image contenant texte, capture d’écran, Police, conception

Le contenu généré par l’IA peut être incorrect.

PANIER :

Dans cette maquette, on peut voir notre panier et le valider. Il y a le lien pour l’accueil, le menus, le panier, le register et le login. Il y a le contenu du panier le total du prix du panier. Après que le panier est validé, le panier devient vide à nouveau.

Une image contenant capture d’écran, texte, conception

Le contenu généré par l’IA peut être incorrect.

ENTREPRISE :

Dans cette maquette, on peut voir, ajouter, modifier ou supprimer des menus dans une entreprise. Il y a le lien pour l’accueil, le menus, le panier, le register et le login. Il y a la liste des menus avec le nom du menu et le prix unitaire du menu. Pour continuer, on peut ajouter un menu, avec le champ de texte du menu, le prix unitaire, le bouton add pour ajouter, le bouton modify pour modifier et le bouton delete pour supprimer le menu. Il a également l’image du menu.

Une image contenant texte, capture d’écran, diagramme, conception

Le contenu généré par l’IA peut être incorrect.

ADMIN :

Dans cette maquette, on peut voir, ajouter, modifier ou supprimer des entreprises. Il y a le lien pour l’accueil, le menus, le panier, le register et le login. Il y a la liste des entreprises avec le nom de l’entreprise et son adresse. Il y a également la liste des utilisateurs avec leur nom, l’entreprise pour laquelle il travail mais il n’est pas obligé d’en avoir une et s’il est admin ou non. Pour continuer, on peut ajouter une entreprise et / ou un utilisateur, avec le champ de texte du nom, l’adresse pour l’entreprise, s’il fait partie d’une entreprise pour l’utilisateur et s’il est admin. les bouton add pour ajouter, les bouton modify pour modifier et les bouton delete pour supprimer.

Une image contenant texte, capture d’écran, nombre, Parallèle

Le contenu généré par l’IA peut être incorrect.

## Sequence System global entre les applications

### Entreprise

Voici le diagramme système coté entreprise pour showListMenu :

* + Client : Utilisateur final ou application qui initie la demande pour voir la liste des menus.
  + API Gateway : Point d'entrée unique pour les appels clients. Il transmet les requêtes vers les services internes.
  + API Rest : Composant backend qui traite les requêtes et interagit avec la base de données.
  + Database : Source des données, ici contenant les menus.
  + GetListMenu()
* Le Client envoie une requête pour obtenir la liste des menus.
* Cette requête est envoyée à API Gateway.
  + ForwarderRequest()
* L’API Gateway transmet la requête au Service API Rest.
  + RequestMenuList()
* Le Service API Rest envoie une requête à la base de données pour obtenir la liste des menus.
  + ReturnListMenu()
* La base de données retourne la liste des menus à l’API Rest.
  + Bloc alternatif (alt) — Condition : MenuListEmpty
* Ce bloc conditionnel gère deux cas selon que la liste soit vide ou non.
  + false : La liste n’est pas vide
* SendMenuList()
* L’API Rest envoie la liste des menus à l’API Gateway.
  + DisplayMenuList()
* L’API Gateway renvoie la liste au Client, qui l’affiche.
  + true : La liste est vide
* ReturnErrorMessage()
* La base de données (ou l’API) renvoie un message d’erreur au Service API Rest.
  + SendErrorMessage()
* L’API Rest envoie un message d’erreur à l’API Gateway.
  + DisplayErrorMessage()
* Le Client reçoit le message d’erreur et l’affiche à l’utilisateur.

Une image contenant texte, diagramme, capture d’écran, Parallèle

Le contenu généré par l’IA peut être incorrect.

Voici le diagramme système coté entreprise pour addMenu :

* + Client : Utilisateur de l’application (ex. via interface web)
  + API Gateway : Point d’entrée de l’API, transmet les données
  + API REST : Traitement métier (vérification, logique)
  + Database : Base de données contenant les menus
  + Informations Valid
* true : Les données envoyées par le client sont valides
* false : Les données sont incorrectes
* Si false : DisplayErrorMessage() côté client
  + Client Valid
* true : Le client est reconnu comme authentifié et autorisé
* false : Le client n’est pas valide
* Si false : DiplayErrorMessage() côté client
  + Menu Exist
* false : Le menu n’existe pas encore, donc on peut l’ajouter
* true : Le menu existe déjà
* Si true : DisplayErrorMessage() côté client
* SendMenuInfo() : Le client envoie les infos du menu à l’API Gateway.
* ForwardMenuInfo() : L’API Gateway transmet à l’API REST.
* RequestAddMenu() : L’API REST fait appel à la base de données pour l'ajout.
* AddMenu() : Ajout effectif du menu.
* ReturnMessageSuccess() : La DB confirme que l’ajout a réussi.
* ForwardMessageSuccess() : L’API REST renvoie le succès à la Gateway.
* DisplaySuccessMessage() : Message de succès côté client.
* ShowMenuList() : Mise à jour visuelle de la liste des menus
*  ForwardMenuInfo() – L’API Gateway transmet à l’API REST.
*  RequestAddMenu() – L’API REST fait appel à la base de données pour l'ajout.
*  AddMenu() – Ajout effectif du menu.
*  ReturnMessageSuccess() – La DB confirme que l’ajout a réussi.
*  ForwardMessageSuccess() – L’API REST renvoie le succès à la Gateway.
*  DisplaySuccessMessage() – Message de succès côté client.
*  ShowMenuList() – Mise à jour visuelle de la liste des menus

Une image contenant texte, diagramme, Parallèle, capture d’écran

Le contenu généré par l’IA peut être incorrect.

### Admin

Ce diagramme décrit l’interaction qui se produit lorsqu’un utilisateur avec le rôle d’Admin souhaite supprimer un utilisateur du système.

**Acteurs et composants impliqués :**

* Admin : utilisateur humain qui déclenche l'action.
* APIGateway : interface système exposée à l'Admin.
* APIAdmin : composant du système gérant l’administration des utilisateurs.
* DB (base de données) : lieu de stockage des informations sur les utilisateurs.

**Déroulement de l’interaction :**

L’Admin déclenche l’action de suppression en appelant la méthode :

* clickDelete() sur l’APIGateway.

L’APIGateway demande les informations nécessaires à l'Admin :

* Appelle la méthode GetInformation() vers l’Admin.

L’Admin renvoie les informations demandées à l’APIGateway :

* En utilisant la méthode SendInformation().

Ensuite, l’APIGateway communique avec l’APIAdmin :

* Il vérifie si l’utilisateur existe avec l’alternative conditionnelle (alt UserExist).
* L’APIAdmin envoie la requête requestToDatabase() vers la base de données (DB) pour vérifier l’existence de l’utilisateur.
* Si l’utilisateur existe, la base de données répond avec userDeleted().

Après confirmation, l’APIAdmin supprime l’utilisateur avec :

* La méthode DeleteUserFromList() vers la base de données (DB).

Après la suppression réussie, l’APIAdmin met à jour la liste d’utilisateurs via :

* La méthode RefreshList() à l’APIGateway.

L’APIGateway affiche finalement la liste mise à jour à l’Admin en appelant :

* ShowList().

**Conditions et alternatives :**

* L’interaction complète (entourée par alt deleteUser) est déclenchée uniquement lorsque l’Admin choisit de supprimer un utilisateur.
* L’autre alternative (alt UserExist) indique une vérification d’existence de l’utilisateur avant suppression.

Une image contenant texte, capture d’écran, diagramme, Parallèle

Le contenu généré par l’IA peut être incorrect.

**Description du diagramme de séquence :**

Ce diagramme décrit précisément le processus lorsqu'un administrateur (**Admin**) souhaite modifier les informations d’un utilisateur via une **API Gateway** et une **restAPI**, avec les données stockées dans une **base de données (DB)**.

**Acteurs et composants impliqués :**

* **Admin** : utilisateur humain qui initie l’action.
* **APIGateway** : interface système qui reçoit les demandes de modification.
* **restAPI** : composant qui traite les requêtes et interagit avec la base de données.
* **DB** : base de données stockant les données des utilisateurs.

**Déroulement détaillé de l'interaction :**

* **L’Admin démarre la modification** en appelant :
  + clickModify() vers l'**APIGateway**.
* L’**APIGateway** demande à l'Admin les nouvelles informations à modifier :
  + Appel de la méthode getInformation().
* **L’Admin** fournit les nouvelles données en réponse à l’APIGateway :
  + via la méthode SendNewInformation().
* L’**APIGateway** transmet ces nouvelles informations vers l’**API REST** :
  + avec une vérification préalable (alt UserExist) pour s’assurer que l’utilisateur existe bien.
* L’**API REST** vérifie dans la base de données (**DB**) si l'utilisateur existe :
  + Méthode requestToDatabase().
* Si l’utilisateur existe et que les changements sont valides (alt ChangesMade), les modifications sont effectuées :
  + Méthode ModifyInDB() appelée sur la **DB**.
* Après avoir modifié la base de données, l’**API REST** confirme la modification à l’**APIGateway** :
  + via la méthode modifyUser().
* L’**APIGateway** actualise la liste des utilisateurs :
  + via la méthode refreshList().
* Finalement, la liste mise à jour est affichée à l’Admin :
  + via la méthode ShowList().

**Conditions alternatives :**

* alt modifyUser : Indique que le processus s'applique spécifiquement lorsqu'une modification est demandée par l'Admin.
* alt UserExist : Assure que la modification ne peut avoir lieu que si l'utilisateur concerné existe déjà en base.
* alt ChangesMade : Confirme que les modifications demandées sont effectivement réalisées.

Une image contenant texte, diagramme, capture d’écran, Rectangle

Le contenu généré par l’IA peut être incorrect.

### Client

Montre les échanges entre les composants (client, passerelle API, service REST) lors d'une tentative d'inscription par un visiteur.

**Acteurs :**

* visitor : utilisateur non connecté
* client : interface web
* APIGateway : traitement des requêtes
* Rest : vérifie et crée l’utilisateur

**Scénario principal :**

1. L’utilisateur clique sur "register"
2. Le client envoie les données à APIGateway
3. Le Rest vérifie si l’utilisateur existe
4. Si tout est bon :
   * L’utilisateur est ajouté
   * Une session est créée
   * Redirection vers la page utilisateur

Une image contenant texte, diagramme, Parallèle, Plan

Le contenu généré par l’IA peut être incorrect.

# Conception à faire complétement avec EA -> à rendre uniquement le fichier EA

## Navigation Diagram complète avec les explications des applications

Ce diagramme illustre le parcours utilisateur global à travers les différentes pages de l’application web, en fonction de ses actions (login, register, logout) et de son statut (utilisateur simple, entreprise ou administrateur).

**Pages principales :**

* **index.html** : Page d’accueil avec accès aux fonctionnalités. Redirection possible vers l’inscription.
* **register.html** : Formulaire d'inscription utilisateur.
* **admin.html / entreprise.html / home.html** : Pages affichées après login selon le rôle.

**Composants logiques :**

* **APIGateway** :
  + Route les actions (submit, logout, redirections...).
  + Gère la session utilisateur.
  + Oriente vers la bonne page selon le rôle.
* **Rest** :
  + Vérifie ou crée un utilisateur.
  + Retourne les infos utilisateur au format JSON.

**Logique de session :**

1. L'utilisateur s'enregistre ou se connecte.
2. Vérification des identifiants → création session.
3. Redirection :
   * Admin → admin.html
   * Entreprise → entreprise.html
   * Autre → home.html
4. Le logout détruit la session via l’APIGateway.

Une image contenant texte, diagramme, capture d’écran, Plan

Le contenu généré par l’IA peut être incorrect.

Ce diagramme décrit la structure côté client de l'application web, en illustrant les liens HTML et les interactions JavaScript dans les processus de login, register et redirection.

**Pages HTML :**

* index.html → accueil
* register.html → inscription
* login.html → connexion
* home.html, entreprise.html, admin.html → redirections post-login selon rôle

**JavaScript :**

* registerCtrl.js & WrkRegister.js : gestion inscription
* LoginCtrl.js & WrkLogin.js : gestion connexion

**Traitement :**

* Les contrôleurs JS envoient les données (submit) à l’APIGateway
* L’APIGateway traite, valide, redirige vers la bonne page selon le rôle

Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

## Diagrammes de classes

### APIGateway

Une image contenant texte, capture d’écran, Parallèle, Police

Le contenu généré par l’IA peut être incorrect.

### Client

Gérer l'inscription, la connexion et l'ajout de commandes

**Composants clés :**

* **User** : modèle utilisateur (nom, prénom, username, password)
* **WrkDB**: accès base de données (addUser, readUser, addOrder…)
* **Wrk**: worker principal (vérifie utilisateur, transforme JSON → User)
* **ItfWrkCtrl** : interface entre logique et contrôle
* **Ctrl** : contrôleur principal (login, register, addOrder)
* **ItfCtrlRest** : interface REST

**Relations :**

* Ctrl → Wrk via ItfWrkCtrl
* Wrk → WrkDB
* Ctrl expose ses méthodes via ItfCtrlRest

Une image contenant texte, capture d’écran, Police, conception

Le contenu généré par l’IA peut être incorrect.

### Entreprise

* + wrk :WrkDB

Classe responsable de l’interaction directe avec la base de données via JDBC.

* + Attributs :
* jdbcConnection: Connection : Connexion JDBC à la base de données.
  + Méthodes :
* addMenu(): boolean : Ajoute un menu à la DB.
* dbConnect(): boolean : Ouvre la connexion DB.
* dbDisconnect(): boolean : Ferme la connexion DB.
* deleteMenu(): boolean : Supprime un menu.
* modifyMenu(): boolean : Modifie un menu.
* WrkDB() : Constructeur.
  + wrk :Wrk

Joue le rôle d’un gestionnaire métier intermédiaire, qui s’appuie sur WrkDB.

* + Attributs :
* wrkDB: WrkDB : Composition, la classe contient une instance de WrkDB.
  + Méthodes :
* addMenu(): void : Ajoute un menu (mais ne retourne rien ici).
* addUser(String): boolean : Ajoute un utilisateur.
* checkUser(String): boolean : Vérifie un utilisateur.
* jsonToUser(String): User : Convertit du JSON vers un objet User.
* Wrk() → Constructeur.
  + wrk :ItfWrkCtrl

Interface définissant les opérations que Wrk expose à la couche Ctrl.

* + Méthodes :
* addMenu(): boolean
* checkUser(String): boolean
* deleteMenu(): boolean
* modifyMenu(): boolean
  + ctrl : Ctrl

Contrôleur principal, qui utilise la couche wrk via l’interface ItfWrkCtrl.

* + Attributs :
* wrk: ItfWrkCtrl : Composition via l'interface, pour décorréler l'implémentation.
  + Méthodes :
* addMenu(): boolean : Appelle wrk.addMenu()
* deleteMenu(): boolean : Supprime un menu via wrk
* login(String): boolean : Authentifie un utilisateur
* modifyMenu(): boolean : Modifie un menu
* register(String): boolean : Inscrit un utilisateur
* Ctrl() : Constructeur
  + ctrl : ItfCtrlRest

Interface REST exposée à l’extérieur (client, API, etc.)

* + Méthodes :
* addMenu(): boolean
* deleteMenu(): boolean
* login(String): boolean
* modifyMenu(): boolean
* register(String): boolean
  + beans : menu

Classe simple représentant une entité de type menu.

* + Attributs :
* nom: String : Nom du menu
* prix: float : Prix du menu

Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.

### Admin

Ce diagramme représente une application structurée en 3 couches distinctes :

* **Couche Contrôleur (Ctrl en vert)** :
  + Reçoit les demandes utilisateur.
  + Interagit avec la couche métier à travers l'interface ItfWrkCtrl.
* **Couche Métier (Wrk en bleu)** :
  + Gère la logique applicative.
  + Communique avec la couche de persistance des données via la classe WrkDB.
* **Couche d'Accès aux données (WrkDB en bleu)** :
  + Se connecte à la base de données et exécute les opérations CRUD.
* **Beans (en rouge, User)** :
  + Simple classe représentant les données utilisateurs (nom, prénom, etc.).

Une image contenant texte, capture d’écran, Police, conception

Le contenu généré par l’IA peut être incorrect.

# Bases de données

## Modèles ER

Modéliser les relations entre utilisateurs, entreprises, menus et commandes.

**Entités principales :**

* **T\_Users** : les utilisateurs
* **T\_Entreprise** : les entreprises
* **T\_Menu** : les menus proposés
* **T\_Commande** : les commandes passées
* **T\_CommandeMenu** : les détails d’une commande (menu, quantité, etc.)



## Modèles WorkBench MySQL

Implémenter en base de données la structure logique des utilisateurs, entreprises, menus et commandes.

**Tables :**

* **T\_Users** : infos utilisateurs + rôle (isAdmin) + lien entreprise
* **T\_Entreprise** : nom + adresse
* **T\_Menu** : menu proposé par une entreprise
* **T\_Commande** : commande avec date et prix total
* **TR\_CommandeMenu** : lie menus et commandes avec la quantité

**Relations :**

* Un user ∈ une entreprise
* Une entreprise → plusieurs menus
* Une commande → plusieurs menus (via TR\_CommandeMenu)

Une image contenant texte, capture d’écran, affichage, nombre

Le contenu généré par l’IA peut être incorrect.

# Implémentation des applications <Le client>, <L’entreprise> et <L’Admin>

## Une descente de code client

## Une descente de code entreprise

## Une descente de code admin

# Implémentation de l'application <API Gateway>

## Une descente de code APIGateway

# Implémentation des applications <Rest client>, <Rest entreprise> et <Rest admin>

## Une descente de code de l'API REST

# Hébergement

# Installation du projet complet avec les 5 applications

# Outils, langages (versions, définitions, installations)

# Tests de fonctionnement du projet

# Auto-évaluations et conclusions

## Auto-évaluation et conclusion de Blanchard

## Auto-évaluation et conclusion de Simões

## Auto-évaluation et conclusion de Ribeiro