

# Rapport du projet GustatIF

B3323 (Marcin Konefal - Adrien Lepic - Quentin Vecchio)

## I - Introduction

Il nous était demandé de développer une application permettant à des utilisateurs de commander des plats dans un restaurant et de pouvoir se les faire livrer à leur domicile. Vous trouverez dans ce rapport toute l'analyse que nous avons faite pour développer cette application, appelée GustatIF. Cette analyse passe par l'élaboration du modèle métier, mais aussi par la conception des interfaces graphiques et la mise en place de services pour la communication entre IHM et base de données.

## II - Analyse du modèle de domaine

### A - Diagramme de cas d'utilisation

Un client peut s'inscrire et se connecter à l'application. Une fois connecté, il peut visualiser la liste des restaurants pour ensuite en choisir un. Une fois le restaurant choisi, il peut visualiser la liste des produits disponibles et constituer sa commande. Lorsqu'il valide sa commande, l'application choisit un livreur et il le notifie. Un livreur, une fois la commande livrée, valide la fin de la commande sur l'application. Le gestionnaire de l'application peut à tout moment consulter l'ensemble des positions (clients, livreurs, restaurants et commandes en cours de livraison) sur une carte.

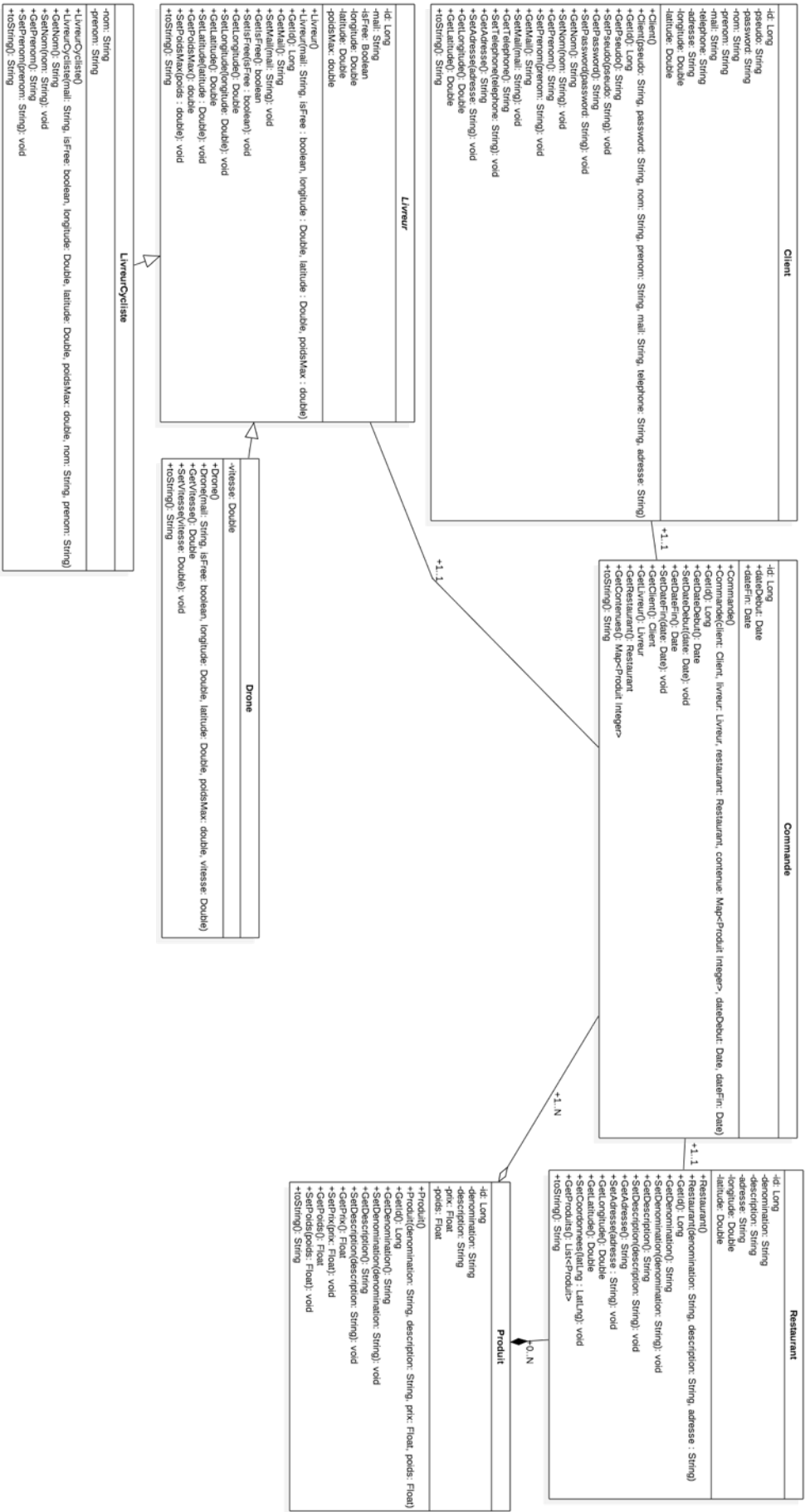
## B - Diagrammes de séquence

1 - Séquences clients

2 - Séquences livreur

3 - Séquence gestionnaire

C - Diagramme de classes



# III - Conception IHM

## A - IHM Inscription client

Afin que le client puisse interagir avec l'application il doit d'abord s'inscrire afin de transmettre ses coordonnées au système. L'interface de connexion se veut très simpliste et passe par un formulaire.

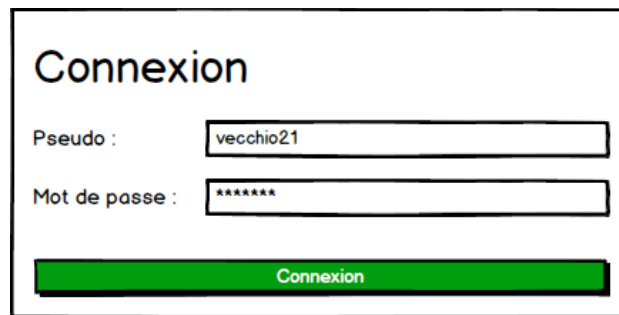
The image shows a web form titled "Inscription" (Registration). It contains several input fields, each preceded by a green checkmark icon. The fields are: "Pseudo" with the value "vecchio21", "Nom" with "Vecchio", "Prénom" with "Quentin", "Email" with "quentin.vecchio@insa-lyon.fr", "Mot de passe" (Password) with masked characters "\*\*\*\*\*", "Confirmation" (Confirmation) with masked characters "\*\*\*\*\*", "Adresse" (Address) with "20 Avenue Albert Einstein Villeurbanne 69100", and "Téléphone" (Phone) with "0689626235". At the bottom of the form are two buttons: a red "Annuler" (Cancel) button and a green "Valider" (Validate) button.

Informations : Pour plus de sécurité le champ mot de passe du formulaire ne doit être pas lisible. Il est également important de tester si le pseudo existe déjà ou non. En effet, le pseudo est unique dans la base de données. Les actions *Annuler* et *Valider* devront rediriger l'utilisateur vers une page d'accueil centrale.

Services utilisés : createClient, clientExist.

## B - IHM Connexion client

Une fois inscrit, le client peut se connecter afin de passer une commande. Pour s'identifier, l'utilisateur doit donner son pseudo et son mot de passe.



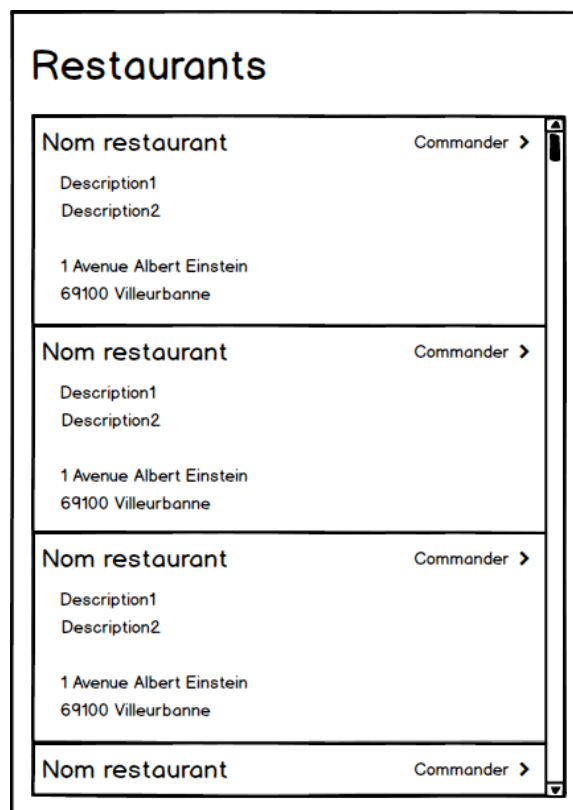
The image shows a login form titled "Connexion". It contains two input fields: "Pseudo :" with the value "vecchio21" and "Mot de passe :" with the value "\*\*\*\*\*". Below these fields is a green button labeled "Connexion".

Informations : Pour plus de sécurité le champ mot de passe du formulaire ne doit être pas lisible. L'action *Connexion* devra renvoyer l'utilisateur vers la page de choix du restaurant.

Service utilisé : connection.

## C - IHM Choix du restaurant

Une fois connecté, l'utilisateur a accès à une liste des restaurants. Pour chaque restaurant, il a aussi accès à son adresse et à une petite description. Il doit choisir un restaurant pour avoir un aperçu de la liste des produits proposée par celui ci et ainsi commencer sa commande.



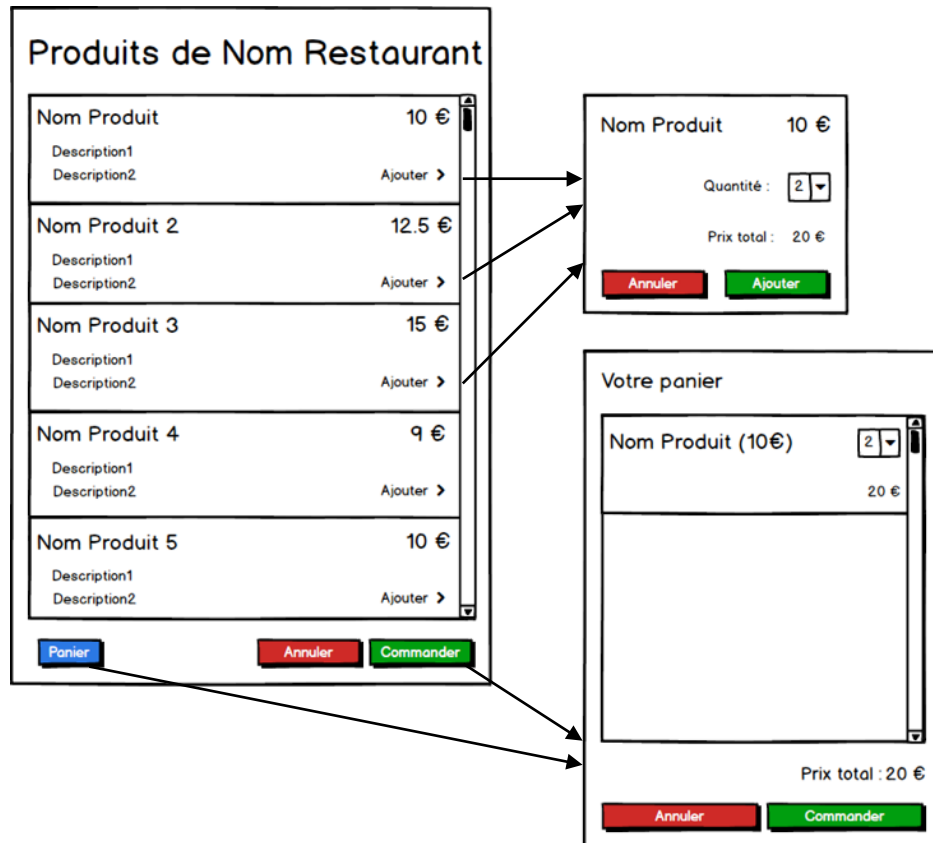
The image shows a list of restaurants titled "Restaurants". It contains four identical entries, each with a "Nom restaurant" field, a "Commander >" button, and a description. The description includes "Description1", "Description2", and the address "1 Avenue Albert Einstein, 69100 Villeurbanne".

Information : Quand l'utilisateur clique sur *Commander*, il est redirigé vers la page Choix des produits.

Service utilisé : findAllRestaurant.

## D - IHM Choix des produits (Commande)

Dès qu'un restaurant est choisi et la liste des produits proposés affichée, l'utilisateur peut commencer sa commande. Pour chaque produit, l'utilisateur a accès à une description, à un prix et à son poids. À chaque fois que l'utilisateur sélectionne un produit, on doit lui demander la quantité qu'il veut commander. L'utilisateur peut valider ou modifier à tout moment sa commande.



Informations : Lorsque l'utilisateur clique sur *Ajouter*, une fenêtre modale lui demande la quantité de produit qu'il veut commander. L'utilisateur peut à tout moment cliquer sur le bouton *Panier* pour avoir un aperçu de sa commande. Dans cette fenêtre modale il pourra également modifier sa commande. Enfin, lorsque l'utilisateur voudra valider sa commande, on lui réaffichera le détails de celle-ci avant d'aller plus loin.

Service utilisé : createCommande.

## E - IHM Fin de commande

Cette interface n'est destinée seulement qu'aux livreurs. Une fois la commande livrée, le livreur doit saisir sur l'interface son adresse mail ainsi que le numéro de commande (unique) qu'il vient de livrer. Si la commande n'existe pas, ou si le mail n'existe pas, ou si le mail donné n'est pas associé à la commande donnée, alors un message d'erreur est affiché sur l'interface. Sinon la date de fin de livraison est automatiquement modifiée et la commande est considérée comme fini.

## Validation fin de commande

Email :

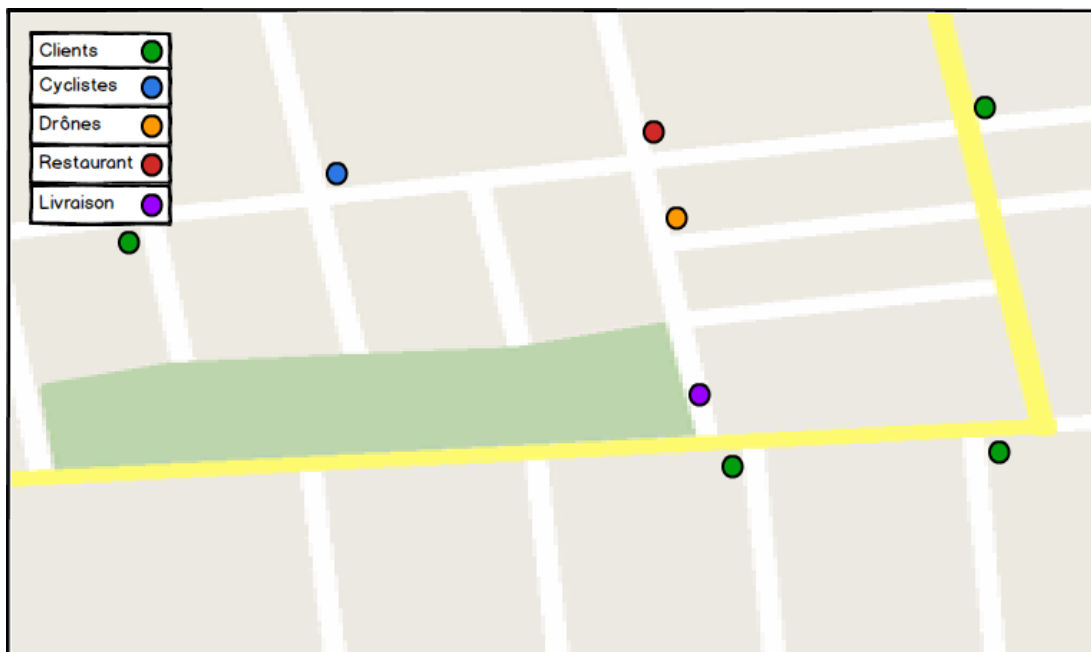
Numéro de commande :

Information : L'erreur doit être affichée dans une fenêtre modale.

Service utilisé : valideCommande.

## F - IHM Tableau de bord de l'application

Cette interface n'est destinée qu'au gestionnaire de l'application. Il a accès à une carte (google maps) où il pourra suivre ses clients, ses livreurs, ses drones et ses livraisons en cours. Il a également accès à un système de filtre qui lui permet de sélectionner ce qu'il veut afficher sur la carte.



Informations : Lorsque l'utilisateur passera son pointeur de souris sur un repère, il aura accès aux informations de celui-ci. Il faut également mettre en place un système de rafraîchissement (Ajax) pour que l'utilisateur puisse suivre le déplacement de ses livreurs. On pourra aussi mettre des tableaux en dessous de la carte pour avoir plus d'informations sur les différents acteurs de la carte.

Services utilisés : findAllClient, findAllRestaurant, findAllLivreur, findAllCommande.

## IV - Développement des services

### A - Service createClient

boolean *createClient*(Client client)

Ce service prend en paramètre un objet de type *Client* et il le crée en base de données. Attention, le client doit avoir un pseudo qui n'existe pas en base de données. Ce service envoie un mail si le client est correctement créé en base de données.

### B - Service clientExist

boolean *clientExist*(String pseudo)

Ce service prend en paramètre un objet de type *String* contenant un pseudo. Le service renvoie *true* si le pseudo existe déjà en base de données, sinon il renvoie *false*.

### C - Service connection

Client *connection*(String pseudo, String password)

Ce service prend en paramètre deux objets de type *String*, le premier contenant un pseudo, et le second contenant un mot de passe. Le service recherche ensuite si il existe un client ayant le pseudo et le mot de passe donnés en paramètre. Si il en trouve un alors il le retourne, sinon il retourne *null*.

### D - Service findAllRestaurant

List<Restaurant> *findAllRestaurant*()

Ce service retourne une liste d'objets de type *Restaurant*. La liste contient tout les restaurants présents en base de données.

### E - Service createCommande

boolean *createCommande*(Commande cmd)

Ce service prend en paramètre un objet de type *Commande*. Le service va tout d'abord assigner un livreur(cycliste ou drone) à la commande selon différents critères (poids et distance). Dans le cas où aucun livreur ne peut livrer la commande alors celle-ci n'est pas créée et le service renvoie *false*. Sinon la commande est créée et la méthode renvoie *true*. Le livreur choisit pour assurer la livraison reçoit un mail avec les différentes informations de la commande.

### F - Service valideCommande

Commande *valideCommande*(String email, Long num)

Ce service prend en paramètre un objet de type *String* qui contient l'email du livreur, ainsi qu'un objet de type *Long* qui contient le numéro de la commande. Le service cherche en base de



données s'il existe un lien entre ce livreur et la commande. S'il en trouve un, alors la commande est traitée comme étant finie et la date de fin est mise à jour (le service renvoi *true*). Si aucun lien n'est trouvé (adresse email introuvable, numéro de commande inexistant, aucun lien entre les deux), alors le service renvoi *false*.

## G - Service findAllClient

List<Client> *findAllClient()*

Ce service retourne une liste d'objets de type *Client*. La liste contient tout les clients présents en base de données.

## H - Service findAllCommandeEnCours

List<Commande> *findAllCommandeEnCours()*

Ce service retourne une liste d'objets de type *Commande*. La liste contient toutes les commandes en cours de livraison présentes en base de données.

## I - Service technique sendMail

void *sendMail*(String destinataire, String objet, String corps)

Ce service permet d'envoyer un mail. Pour les besoins de l'application il nous était seulement demandé de l'afficher dans la console. Le service prend en paramètre le destinataire, l'objet du mail, et son corps.

## J - Service technique findBestLivreur

Livreur *findBestLivreur* (List <Livreur> livreurs , Double longitude, Double latitude)

Ce service permet de trouver le meilleur livreur susceptible de livrer la commande au plus vite. Pour cela nous sommes basé sur la position des livreurs, leur vitesses, et également sur le poids maximum qu'ils pouvaient transporter. Si aucun livreur n'est disponible la méthode renvoie *null*, sinon elle renvoie l'objet de type *Livreur*.

## K - Autres services

Pour le besoin de l'application ou pour la maintenance, nous avons également rajouté d'autres services :

- Client *updateClient*(Client client)

Ce service permet de mettre à jour un client, on peut imaginer une IHM de personnalisation du profil.

- Restaurant *findRestaurantById*(Long id)

Ce service permet de retrouver un restaurant grâce à son identifiant. Nous nous en servons dans l'IHM console, cependant il ne servira à rien dans l'IHM Web.

- Produit *findProduitById*(Long id)

Ce service permet de retrouver un produit grâce à son identifiant. Nous nous en servons dans l'IHM console, cependant il ne servira à rien dans l'IHM Web.

- List<Produit> *findAllProduit*()

Ce service permet de retrouver tous les produits existant en base de données.

- void *createLivreurCycliste*(LivreurCycliste livreur)

Ce service permet la création d'un livreur de type *LivreurCycliste*. Il est utilisé pour la création de jeux d'essais pour la base de données.

- void *createDrone*(Drone drone)

Ce service permet la création d'un livreur de type *Drone*. Il est utilisé pour la création de jeux d'essais pour la base de données.

- List<Commande> *findAllCommande*()

Ce service permet de retrouver toutes les commandes présentes en base de données.

## V - Notes

Vous trouverez dans l'archive un script SQL qui permet de générer un jeu d'essai, ce script est différent et est plus complet que celui disponible sur Moodle.