RAPPORT DU PROJET

Gestion d’une entreprise de pizzas à domicile

Réalisé Par : DEHMANI Manar
CHEN MICHEL
CHEBLI WIEM
MADIODIOU DIAGNE Maguette

Encadré Par : LEFEBVRE Pierre

Année Universitaire : 2023 - 2024

Table des matières

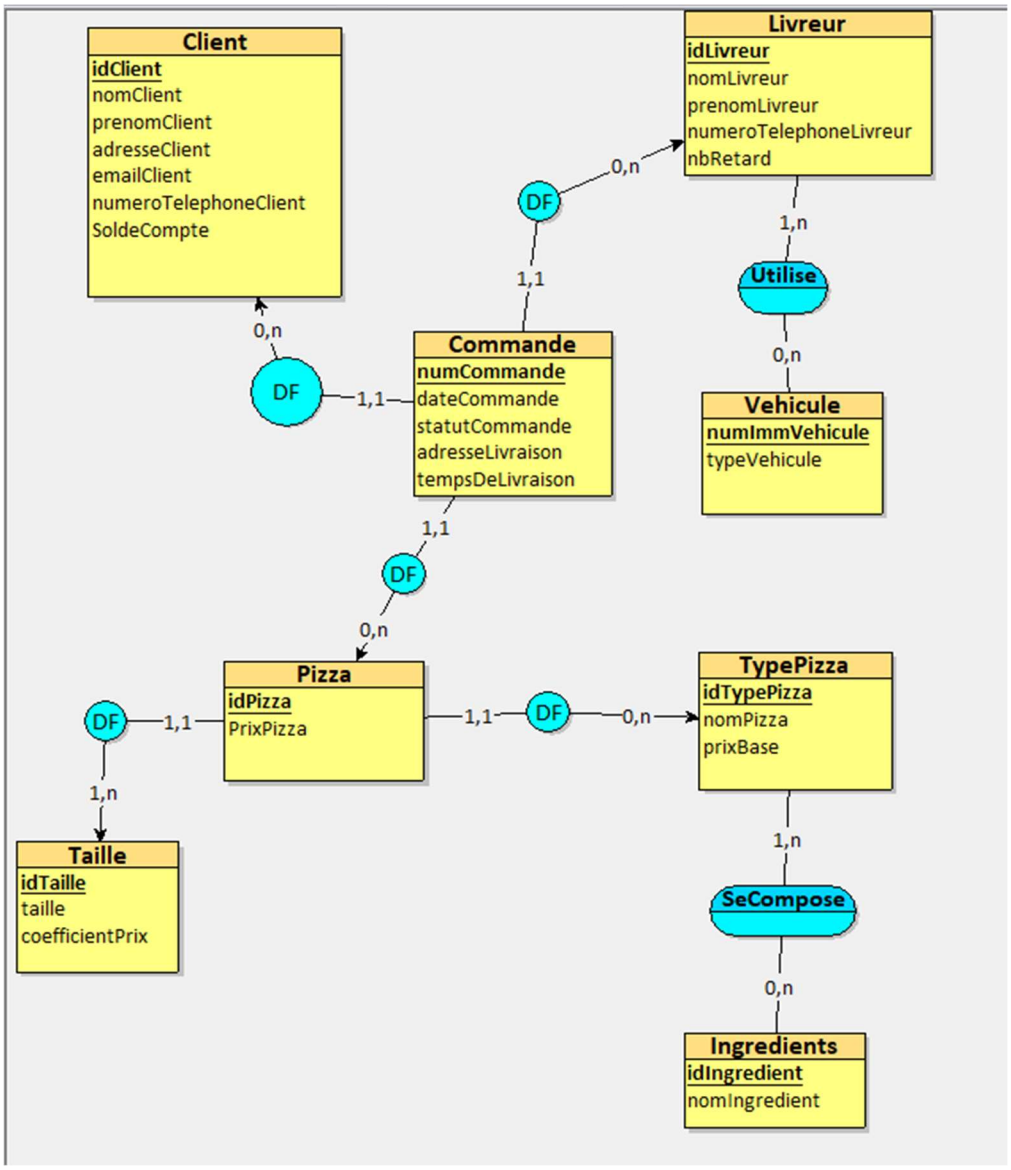
1. Introduction

Cette documentation présente la conception et le développement d'un système de gestion pour l'entreprise de fabrication et de livraison de pizzas à domicile, RaPizz. Le projet couvre la création d'une base de données pour gérer les produits, les commandes, les livreurs, et les véhicules, ainsi que des fonctionnalités statistiques et analytiques pour optimiser les opérations de l'entreprise.

2. Conception de la Base de Données

2.1 Modèle Entité-Association

Nous avons commencé par réaliser le modèle entité-association afin de représenter les principales entités du système et leurs relations.



- **Pizza** : C'est la table qui représente la pizza de la commande choisie par le client, chaque commande est caractérisée par son id et son prix qui dépend de la taille et du type de la pizza
- **Taille** : C'est la table qui représente les trois différentes tailles pour chaque pizza (naine, humaine, ogresse). Nous avons pensé à mettre en place un coefficient prix pour que la « naine » soit 1/3 moins chère que le prix de base, c'est-à-dire la taille « humaine », et l'« ogresse » soit1/3 plus chère.
- **TypePizza :** Cette table représente les différentes pizzas présentent dans la pizzeria , chaque type est caractérisée par l'id, le nom et le prix de base de la pizza sans prendre en compte sa taille.
- **Ingrédient :** Cette table représente les différents ingrédients qui composent les pizzas
- **Livreur** : C'est la table qui représente les livreurs de l'entreprise, chaque livreur est caractérisé par son id, son nom, son prénom, son numéro de téléphone et le nombre de retard effectué lors de la livraison des commandes
- **Vehicule :** C'est la table qui représente les véhicules mises à disposition aux livreurs pour qu'ils puissent assurer les livraisons, chaque véhicule est caractérisé par le numéro d'immatriculation et son type qui peut être une voiture ou une moto.
- **Client** : Cette table représente les clients de l'entreprise, chaque client est caractérisé par son id, son nom, son prénom, son adresse, son email, son numéro de téléphone et son solde de compte qui doit être suffisant pour qu'il peut passer une commande
- **Commande** : C'est la table qui représente les commandes effectuées par les clients, chaque commande est caractérisée par : la date, le statut qui peut être « Annulé – En cours – Offerte – Livrée », l'adresse et le temps de livraison

2.2 Passage au Modèle Relationnel

Après avoir réalisé le schéma entité-association, nous avons réalisé le passage relationnel pour savoir toutes les tables nécessaires avec leurs attributs et éclaircir les relations entre les tables (clé étrangère), pour qu'on puisse par la suite créer le schéma de notre base de données.

TypePizza = (idTypePizza VARCHAR (15), nomPizza VARCHAR(20), prixBase DECIMAL(5,2));

Vehicule = (numImmVehicule VARCHAR (10), typeVehicule VARCHAR(10));

Livreur = (idLivreur VARCHAR (15), nomLivreur VARCHAR(20), prenomLivreur VARCHAR(20), numeroTelephoneLivreur VARCHAR(15), nbRetard INT);

Client = (idClient INT, nomClient VARCHAR(15), prenomClient VARCHAR(15), adresseClient VARCHAR(50), emailClient VARCHAR(30), numeroTelephoneClient VARCHAR(15), SoldeCompte DECIMAL(7,2));

Ingredients = (idIngredient VARCHAR(15), nomIngredient VARCHAR(15));

Taille = (idTaille VARCHAR(15), taille VARCHAR(15), coefficientPrix DECIMAL(5,2));

Pizza = (idPizza VARCHAR(15), PrixPizza DECIMAL(5,2), #idTaille, #idTypePizza);
Commande = (numCommande INT, dateCommande DATETIME, statutCommande VARCHAR(10), adresseLivraison VARCHAR(50), tempsDeLivraison TIME, #idPizza, #idLivreur, #idClient);

Utilise = (#numImmVehicule, #idLivreur);

SeCompose = (#idTypePizza, #idIngredient);

2.3 Script de Création des Tables

```sql
create database if not exists pizza ;
use pizza ;

CREATE TABLE TypePizza(
   idTypePizza VARCHAR(15),
   nomPizza VARCHAR(20) NOT NULL,
   prixBase DECIMAL(5,2) NOT NULL,
   PRIMARY KEY(idTypePizza)
);

CREATE TABLE Vehicule(
   numImmVehicule VARCHAR(10),
   typeVehicule VARCHAR(10) NOT NULL,
   PRIMARY KEY(numImmVehicule)
);

CREATE TABLE Livreur(
   idLivreur VARCHAR(15),
   nomLivreur VARCHAR(20),
   prenomLivreur VARCHAR(20),
   numeroTelephoneLivreur VARCHAR(15),
   nbRetard INT,
   PRIMARY KEY(idLivreur)
);

CREATE TABLE Client(
   idClient INT,
   nomClient VARCHAR(15),
   prenomClient VARCHAR(15),
   adresseClient VARCHAR(50),
   emailClient VARCHAR(30),
   numeroTelephoneClient VARCHAR(15),
   SoldeCompte DECIMAL(7,2),
   PRIMARY KEY(idClient)
);

```

```sql
CREATE TABLE Ingredients(
    idIngredient VARCHAR(15),
    nomIngredient VARCHAR(15),
    PRIMARY KEY(idIngredient)
);

CREATE TABLE Taille(
    idTaille VARCHAR(15),
    taille VARCHAR(15),
    coefficientPrix DECIMAL(5,2),
    PRIMARY KEY(idTaille)
);

CREATE TABLE Pizza(
    idPizza VARCHAR(15),
    PrixPizza DECIMAL(5,2),
    idTaille VARCHAR(15) NOT NULL,
    idTypePizza VARCHAR(15) NOT NULL,
    PRIMARY KEY(idPizza),
    FOREIGN KEY(idTaille) REFERENCES Taille(idTaille),
    FOREIGN KEY(idTypePizza) REFERENCES TypePizza(idTypePizza)
);

CREATE TABLE Commande(
    numCommande INT,
    dateCommande DATETIME,
    statutCommande VARCHAR(10),
    adresseLivraison VARCHAR(50),
    tempsDeLivraison TIME,
    idPizza VARCHAR(15) NOT NULL,
    idLivreur VARCHAR(15) NOT NULL,
    idClient INT NOT NULL,
    PRIMARY KEY(numCommande),
    FOREIGN KEY(idPizza) REFERENCES Pizza(idPizza),
    FOREIGN KEY(idLivreur) REFERENCES Livreur(idLivreur),
    FOREIGN KEY(idClient) REFERENCES Client(idClient)
);

CREATE TABLE Utilise(
    numImmVehicule VARCHAR(10),
    idLivreur VARCHAR(15),
    PRIMARY KEY(numImmVehicule, idLivreur),
    FOREIGN KEY(numImmVehicule) REFERENCES Vehicule(numImmVehicule),
    FOREIGN KEY(idLivreur) REFERENCES Livreur(idLivreur)
);

CREATE TABLE SeCompose(
    idTypePizza VARCHAR(15),
    idIngredient VARCHAR(15),
    PRIMARY KEY(idTypePizza, idIngredient),
    FOREIGN KEY(idTypePizza) REFERENCES TypePizza(idTypePizza),
    FOREIGN KEY(idIngredient) REFERENCES Ingredients(idIngredient)
);
```

2.4 Insertion des Données

Avant l'insertion des données dans les tables nous avons mis en place quatre triggers qui vérifient certaines conditions.

2.4.1 Premier Trigger

Ce trigger vérifie si un livreur est disponible pour prendre en charge une commande c-à-d si le livreur a une commande dont le statut est en cours le livreur sera indisponible

```
-- Trigger qui vérifie si un livreur est disponible avant le rajout d'une commande
DELIMITER //
CREATE TRIGGER before_insert_commande
BEFORE INSERT ON Commande
FOR EACH ROW
BEGIN
    DECLARE livreur_occupe INT;

    -- On Vérifie si le livreur est déjà assigné à une commande en cours
    SELECT COUNT(*) INTO livreur_occupe
    FROM Commande
    WHERE idLivreur = NEW.idLivreur
    AND statutCommande = 'En cours';

    IF livreur_occupe > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Le livreur n\'est pas disponible.';
    END IF;
END //
DELIMITER ;

```

2.4.2 Deuxième Trigger

Ce trigger vérifie si le solde du client est suffisant pour qu'il puisse passer une commande

```
-- Trigger qui vérifie le solde du client
DELIMITER //
CREATE TRIGGER before_insert_commande_solde
BEFORE INSERT ON Commande
FOR EACH ROW
BEGIN
    DECLARE prixPizza DECIMAL(5,2);
    DECLARE soldeClient DECIMAL(7,2);

    -- Récupère le prix de la pizza commandée
    SELECT PrixPizza INTO prixPizza
    FROM Pizza
    WHERE idPizza = NEW.idPizza;

    -- Récupère le solde du client
    SELECT SoldeCompte INTO soldeClient
    FROM Client
    WHERE idClient = NEW.idClient;

```

```
41          -- Vérifie si le client a suffisamment de solde pour payer la pizza
42          IF soldeClient < prixPizza THEN
43              SIGNAL SQLSTATE '45000'
44              SET MESSAGE_TEXT = 'Le solde du compte client est insuffisant.';
45          END IF;
46      END //
```

2.4.3 Troisième Trigger

Ce trigger a pour but de mettre à jour le statut de la commande en commande Offerte si le temps de livraison est supérieur à trente minutes ou le nombre de commande du client a atteint 10 sinon on met à jour le solde du client après avoir passé la commande.

```
48  -- Trigger qui permet de gérer la mise à jour de la commande et les conditions de gratuité à la livraison
49  DELIMITER //
50  CREATE TRIGGER after_update_statut_commande
51  AFTER UPDATE ON Commande
52  FOR EACH ROW
53  BEGIN
54      DECLARE dureeLivraison INT;
55      DECLARE pizzas_livrees INT;
56      DECLARE pizza_gratuite BOOLEAN DEFAULT FALSE;
57      DECLARE prixPizza DECIMAL(5,2);
58
59      -- On Calcule la durée de livraison en minutes si la commande est livrée
60      IF NEW.statutCommande = 'Livrée' AND OLD.statutCommande != 'Livrée' THEN
61          SET dureeLivraison = TIMESTAMPDIFF(MINUTE, NEW.dateCommande, NOW());
62          UPDATE Commande
63          SET tempsDeLivraison = SEC_TO_TIME(dureeLivraison * 60)
64          WHERE numCommande = NEW.numCommande;
65
66          -- On vérifie si la durée de livraison dépasse 30 minutes
67          IF dureeLivraison > 30 THEN
68              SET pizza_gratuite = TRUE;
```

```
69                  UPDATE Livreur
70                  SET nbRetard = nbRetard + 1
71                  WHERE idLivreur = NEW.idLivreur;
72              END IF;
73
74              -- On Vérifie si le client a droit à une pizza gratuite pour fidélité
75              SELECT COUNT(*) INTO pizzas_livrees
76              FROM Commande
77              WHERE idClient = NEW.idClient AND statutCommande = 'Livrée';
78
79              IF pizzas_livrees % 10 = 0 THEN
80                  SET pizza_gratuite = TRUE;
81              END IF;
82
83              -- Si la pizza est gratuite, ne pas facturer le client
84              IF pizza_gratuite THEN
85                  -- Pas de débit du solde
86                  UPDATE Commande
87                  SET statutCommande = 'Offerte'
88                  WHERE numCommande = NEW.numCommande;
89              ELSE
```

```
90              -- Sinon On récupére le prix de la pizza commandée
91              SELECT PrixPizza INTO prixPizza
92              FROM Pizza
93              WHERE idPizza = NEW.idPizza;
94
95              -- On Débite le compte du client
96              UPDATE Client
97              SET SoldeCompte = SoldeCompte - prixPizza
98              WHERE idClient = NEW.idClient;
99          END IF;
100     END IF;
101 END //
102 DELIMITER ;
```

2.4.4 Quatrième Trigger

Ce trigger a pour but d'empêcher la modification du statut d'une commande terminée qui est déjà livrée ou annulée ou offerte.

```
104 -- Trigger qui permet d'empêcher la modification du statut d'une commande terminée
105 DELIMITER //
106 CREATE TRIGGER before_update_commande_statut
107 BEFORE UPDATE ON Commande
108 FOR EACH ROW
109 BEGIN
110     IF OLD.statutCommande IN ('Livrée', 'Annulée', 'Offerte') THEN
111         SIGNAL SQLSTATE '45000'
112         SET MESSAGE_TEXT = 'Vous ne pouvez pas modifier une commande qui est déjà Livrée,
113         Annulée ou Offerte.';
114     END IF;
115 END //
116 DELIMITER ;
```

2.4.5 Script d'Insertion des Données

Après avoir mettre en place et exécuter les triggers nécessaires pour la gestion de ventes nous avons procédé à l'insertion des données dans notre base.

```sql
-- Insertion des données dans la table véhicule
INSERT INTO Vehicule (numImmVehicule, typeVehicule)
 VALUES ('V1', 'voiture'),
        ('V2', 'moto'),
        ('V3', 'voiture'),
        ('V4', 'voiture'),
        ('V5', 'voiture'),
        ('V6', 'moto'),
        ('V7', 'moto'),
        ('V8', 'voiture'),
        ('V9', 'voiture'),
        ('V10', 'moto');
-- Insertion dans la table livreur
INSERT INTO Livreur (idLivreur, nomLivreur, prenomLivreur, numeroTelephoneLivreur, nbRetard)
VALUES  ('Livreur1', 'Dupont', 'Jean', '0612345678', 2),
        ('Livreur2', 'Martin', 'Luc', '0698765432', 5),
        ('Livreur3', 'Lopez', 'Marie', '0654321987', 0),
        ('Livreur4', 'Barbara', 'kevin', '0654321987', 10),
        ('Livreur5', 'Chen', 'Michel', '0654321987', 0),
        ('Livreur6', 'Gregori', 'Luca', '0654321987', 7),
        ('Livreur7', 'Dubois', 'Margaux', '0654321987', 3);
 -- Insertion dans la table utilise
 INSERT INTO Utilise (numImmVehicule, idLivreur)
 VALUES
     ('V1', 'Livreur1'),
     ('V2', 'Livreur1'),
     ('V3', 'Livreur2'),
     ('V4', 'Livreur3'),
     ('V5', 'Livreur3'),
     ('V6', 'Livreur3'),
     ('V7', 'Livreur4'),
     ('V8', 'Livreur5'),
     ('V2', 'Livreur5'),
     ('V5', 'Livreur6'),
     ('V4', 'Livreur7');
 -- Insertion dans la table ingrésient
 INSERT INTO Ingredients (idIngredient, nomIngredient)
 VALUES  ('I1', 'Tomate'),
         ('I2', 'Fromage'),
         ('I3', 'Pepperoni'),
         ('I4', 'Viande Hachée'),
         ('I5', 'Saucisses'),
         ('I6', 'Poulet'),
         ('I7', 'Oignons'),
         ('I8', 'Poivrons'),
         ('I9', 'Champignons'),
         ('I10', 'Basilic'),
         ('I12','Sauce Barbecue'),
         ('I13','Sauce Blanche'),
         ('I14','Sauce Tomate');
```

```sql
-- Insertion dans table TypePizza
INSERT INTO TypePizza (idTypePizza, nomPizza, prixBase) VALUES
('Type1', 'Margherita', 9.50),
('Type2', 'Pepperoni', 10.75),
('Type3', 'Cannibale', 12.35),
('Type4', 'Végétarienne', 13.80),
('Type5', 'Indienne', 10.99),
('Type6', 'Spéciale Champis', 10.99);
-- Insertion dans la table se_compose
INSERT INTO  SeCompose (idTypePizza, idIngredient) VALUES
('Type1', 'I14'), ('Type1','I2'),
('Type2', 'I14'), ('Type2', 'I2'), ('Type2', 'I3'),
('Type3', 'I12'), ('Type3', 'I2'), ('Type3', 'I4'),('Type3', 'I5'), ('Type3', 'I6'),
('Type4', 'I13'), ('Type4', 'I1'), ('Type4', 'I7'), ('Type4', 'I8'),
('Type5', 'I13'), ('Type5', 'I2'),('Type5', 'I9'),  ('Type5', 'I7'), ('Type5', 'I6'),
('Type6', 'I13'), ('Type6', 'I2'), ('Type6', 'I9'), ('Type6', 'I7'), ('Type6', 'I8');
```

```sql
-- Insertion dans la table taillePizza
INSERT INTO Taille (idTaille, taille, coefficientPrix)
VALUES  ('T1', 'Naine', 0.70),
        ('T2', 'Humaine', 1.00),
        ('T3', 'Ogresse', 1.30);
-- Insertion dans la table Client
INSERT INTO Client (idClient, nomClient, prenomClient, adresseClient, emailClient, numeroTelephoneClient,
(1, 'Derrez', 'Jonathan', '35 Rue Maurice Bécanne', 'jonathan.derrez@gmail.com', '0654321890', 200.99),
(2, 'Gregorie', 'Paul', '4 Avue du Régiment', 'paul.gregorie@gmail.com', '0612345698', 24.56),
(3, 'DEHMANI', 'Manar', '4 Avenue de Buisson', 'manar.dehmani@gamil.com', '0678912345', 127.45),
(4, 'Moreau', 'David', '2 Boulevard de la Mer', 'david.moreau@gamil.com', '0634567890', 60.78),
(6, 'CHEBLI', 'Wiem', '10 Place de la Liberté', 'wiem.chebli@gmail.com', '0687654321', 5.20),
(7, 'Roux', 'Elise', '8 Place de la République', 'elise.roux@gmail.com', '0701658921', 70.32),
(8, 'Poiteaux', 'Justine', '2 Rue de la mairie', 'justine.poiteaux@gmail.com', '0647214300', 0.00),
(9, 'Robert', 'Martine', '35 Avenue Artiside Briand', 'martine.robert@gmail.com', '0725543210', 43.15);
```

```sql
-- Insertion dans la table Pizza
INSERT INTO Pizza (idPizza, PrixPizza, idTaille, idTypePizza) VALUES
('Type1T1', 6.65, 'T1', 'Type1'),
('Type1T2', 9.50, 'T2', 'Type1'),
('Type1T3', 12.35, 'T3', 'Type1'),
('Type2T1', 7.53, 'T1', 'Type2'),
('Type2T2', 10.75, 'T2', 'Type2'),
('Type2T3', 13.98, 'T3', 'Type2'),
('Type3T1', 8.65, 'T1', 'Type3'),
('Type3T2', 12.35, 'T2', 'Type3'),
('Type3T3', 16.06, 'T3', 'Type3'),
('Type4T1', 9.66, 'T1', 'Type4'),
('Type4T2', 13.80, 'T2', 'Type4'),
('Type4T3', 17.94, 'T3', 'Type4'),
('Type5T1', 7.69, 'T1', 'Type5'),
('Type5T2', 10.99, 'T2', 'Type5'),
('Type5T3', 14.29, 'T3', 'Type5'),
('Type6T1', 7.69, 'T1', 'Type6'),
('Type6T2', 10.99, 'T2', 'Type6'),
('Type6T3', 14.29, 'T3', 'Type6');
```

```sql
-- Insertion des commandes dans la table Commande
INSERT INTO Commande (numCommande, dateCommande, statutCommande, adresseLivraison, tempsDeLivraison, idPizza, idLivreur, idClient)
VALUES
(1, '2024-06-13 12:30:00', 'Livrée', '35 Rue Maurice Bécanne', '00:30:00', 'Type1T2', 'Livreur1', 1),
(2, '2024-06-13 13:00:00', 'Livrée', '35 Rue Maurice Bécanne', '00:20:00', 'Type2T2', 'Livreur2', 1),
(3, '2024-06-13 13:15:00', 'offerte', '35 Rue Maurice Bécanne', '00:40:00', 'Type3T3', 'Livreur3', 1),
(4, '2024-06-13 14:00:00', 'Livrée', '35 Rue Maurice Bécanne', '00:25:00', 'Type4T2', 'Livreur4', 1),
(5, '2024-06-13 14:30:00', 'Livrée', '35 Rue Maurice Bécanne', '00:25:00', 'Type5T1', 'Livreur5', 1),
(6, '2024-06-13 15:00:00', 'Livrée', '4 Avue du Régiment', '00:30:00', 'Type6T3', 'Livreur6', 2),
(7, '2024-06-13 15:30:00', 'Annulée', '2 Rue de la mairie', '00:00:00', 'Type1T1', 'Livreur7', 8),
(8, '2024-06-13 16:00:00', 'Offerte', '35 Avenue Artiside Briand', '00:50:00', 'Type2T2', 'Livreur1', 9),
(9, '2024-06-13 16:30:00', 'Livrée', '35 Avenue Artiside Briand', '00:30:00', 'Type3T3', 'Livreur2', 9),
(10, '2024-06-13 17:00:00', 'En cours', '35 Avenue Artiside Briand', '00:45:00', 'Type4T2', 'Livreur3', 9),
(11, '2024-06-13 17:30:00', 'En cours', '4 Avenue de Buisson', '00:10:00', 'Type5T1', 'Livreur4', 3),
(12, '2024-06-13 18:00:00', 'Offerte', '4 Avenue de Buisson', '00:55:00', 'Type6T3', 'Livreur5', 3);
```

3. Interrogation de la Base de Données

Pour cette partie nous avons mis en place plusieurs procédures stockées pour répondre aux différents besoins.

- Menu :

```sql
-- Procédure Stockée permet d'extraire des données pour imprimer le menu (nom de chaque pizza, son prix et les ingrédients)
DELIMITER //
CREATE PROCEDURE menu()
BEGIN
    SELECT P.idPizza, TP.nomPizza, T.taille ,P.PrixPizza, GROUP_CONCAT(I.nomIngredient SEPARATOR ', ') AS ingredients
    FROM Pizza P
    JOIN TypePizza TP ON P.idTypePizza = TP.idTypePizza
    JOIN Taille T ON P.idTaille = T.idTaille
    JOIN SeCompose S ON TP.idTypePizza = S.idTypePizza
    JOIN Ingredients I ON S.idIngredient = I.idIngredient
    GROUP BY TP.nomPizza, P.PrixPizza;
END //
DELIMITER ;

```

- Fiche de la livraison :

```sql
-- Procédure Stockée permet d'imprimer la fiche de la livraison d'une commande
DELIMITER //
CREATE PROCEDURE ficheLivraison()
BEGIN
     SELECT L.nomLivreur, L.prenomLivreur, V.typeVehicule, Cl.nomClient, Cl.prenomClient, Co.dateCommande,
           CASE WHEN Co.tempsDeLivraison > 30 THEN 'Oui' ELSE 'Non' END AS retard,
           TP.nomPizza, TP.prixBase
    FROM Commande Co
    JOIN Livreur L ON Co.idLivreur = L.idLivreur
    JOIN Utilise U ON L.idLivreur = U.idLivreur
    JOIN Vehicule V ON U.numImmVehicule = V.numImmVehicule
    JOIN Client Cl ON Co.idClient = Cl.idClient
    JOIN Pizza P ON Co.idPizza = P.idPizza
    JOIN TypePizza TP ON P.idTypePizza = TP.idTypePizza
    WHERE Co.statutCommande IN ('En cours', 'livrée', 'offerte');
END //
DELIMITER ;
```

- Chiffres d’affaires :

```
-- Procédure stockée qui permet de calculer le chiffre d'affaires de la pizzeria
DELIMITER //
CREATE PROCEDURE chiffreAffaires()
BEGIN
    SELECT SUM(PrixPizza) AS chiffre_affaires
    FROM Commande
    JOIN Pizza ON Commande.idPizza = Pizza.idPizza
    WHERE statutCommande = 'Livrée';
END //
DELIMITER ;
```

- Véhicules n’ayant jamais servi :

```
-- Procédure stockée permet d'identifier la ou les véhicules non utilisé
DELIMITER //
CREATE PROCEDURE vehiculeNonUtilise()
BEGIN
    SELECT V.numImmVehicule, V.typeVehicule
    FROM Vehicule V
    LEFT JOIN Utilise U ON V.numImmVehicule = U.numImmVehicule
    WHERE U.numImmVehicule IS NULL;
END //
DELIMITER ;
```

- Nombre de commandes par client :

```
-- Procédure Stcokée permet de calculer le nombre de commande par client
DELIMITER //
CREATE PROCEDURE nbCommandeParClient(IN idClient INT)
BEGIN
    SELECT Cl.idClient, Cl.nomClient, Cl.prenomClient, COUNT(Co.numCommande) AS nombre_commandes
    FROM Client Cl
    LEFT JOIN Commande Co ON Cl.idClient = Co.idClient
    WHERE Cl.idClient = idClient
    GROUP BY Cl.idClient, Cl.nomClient, Cl.prenomClient;
END //
DELIMITER ;
```

- Le plus mauvais livreur :

```
-- Procédure Stcokée permet d'identifier le plus mauvais livreur en se basant sur le nombre de retard
DELIMITER //
CREATE PROCEDURE PlusMauvaisLivreur()
BEGIN
    SELECT *
    FROM Livreur L
    ORDER BY L.nbRetard DESC
    Limit 1;
END //
DELIMITER ;
```

- L'ingrédient Favori :

```
-- Procédure stockée permet de récupérer l'ingrédient Favori
DELIMITER //
CREATE PROCEDURE ingredientFavori()
BEGIN
    SELECT I.nomIngredient, COUNT(S.idIngredient) AS nombreUtilisations
    FROM SeCompose S
    JOIN Ingredients I ON S.idIngredient = I.idIngredient
    GROUP BY I.nomIngredient
    ORDER BY nombreUtilisations DESC
    LIMIT 1;
END //
DELIMITER ;
```

- La pizza la plus demandée

```
-- Procédure Stockée permet d'identifier la pizza la plus demandée
DELIMITER //
CREATE PROCEDURE PizzaPlusdemandee()
BEGIN
    SELECT
        Pizza.idPizza,
        TypePizza.nomPizza,
        Taille.taille,
        Pizza.PrixPizza,
        COUNT(Commande.numCommande) AS nombre_commandes
    FROM Commande
    JOIN Pizza ON Commande.idPizza = Pizza.idPizza
    JOIN TypePizza ON Pizza.idTypePizza = TypePizza.idTypePizza
    JOIN Taille ON Pizza.idTaille = Taille.idTaille
    GROUP BY Pizza.idPizza, TypePizza.nomPizza, Taille.taille
    ORDER BY nombre_commandes DESC
    LIMIT 1;
END //
DELIMITER ;
```

- Le meilleur client :

```
/* Procédure stockée qui permet de retourner le meilleur client
en se basant sur le montant depense et le nombre de commande passé */
DELIMITER //

CREATE PROCEDURE meilleurClient()
BEGIN
    SELECT
        Client.idClient,
        Client.nomClient,
        Client.prenomClient,
        SUM(Pizza.PrixPizza) AS montant_total_depense
    FROM Client
    JOIN Commande ON Client.idClient = Commande.idClient
    JOIN Pizza ON Commande.idPizza = Pizza.idPizza
    WHERE Commande.statutCommande = 'Livrée'
    GROUP BY Client.idClient, Client.nomClient, Client.prenomClient
    ORDER BY montant_total_depense DESC
    LIMIT 1;
END //

DELIMITER ;
```

- Moyenne des commandes :

```
-- Procédure Stockée permet de calculer la moyenne du prix des commandes livrées
DELIMITER //
CREATE PROCEDURE moyenneCommande()
BEGIN
    SELECT AVG(P.PrixPizza) AS moyenne_prix_commandes
    FROM Commande Co
    JOIN Pizza P ON Co.idPizza = P.idPizza
    WHERE Co.statutCommande = 'Livrée';
END //
DELIMITER ;
```

4. Programmation

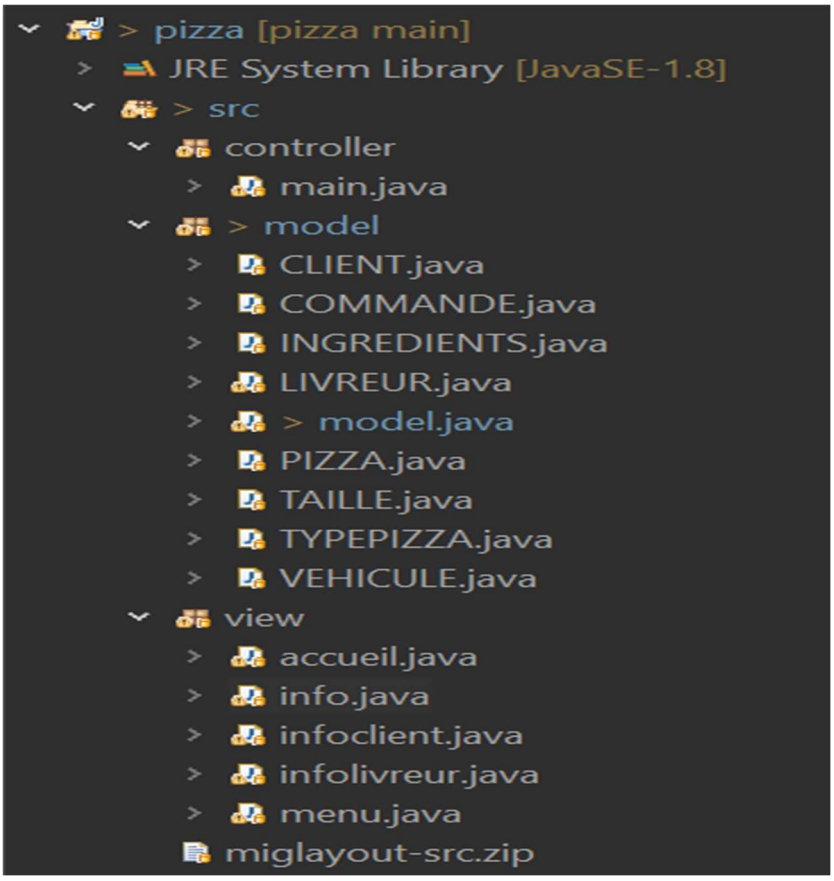
4.1. Architecture du Système

Pour le côté serveur, nous avons utilisé Java comme langage de programmation avec l'API JDBC pour interagir avec la base de données. Nous avons structuré notre application en utilisant le modèle MVC (Modèle-Vue-Contrôleur), ce qui permet de séparer les préoccupations et de rendre le code plus modulaire et maintenable.

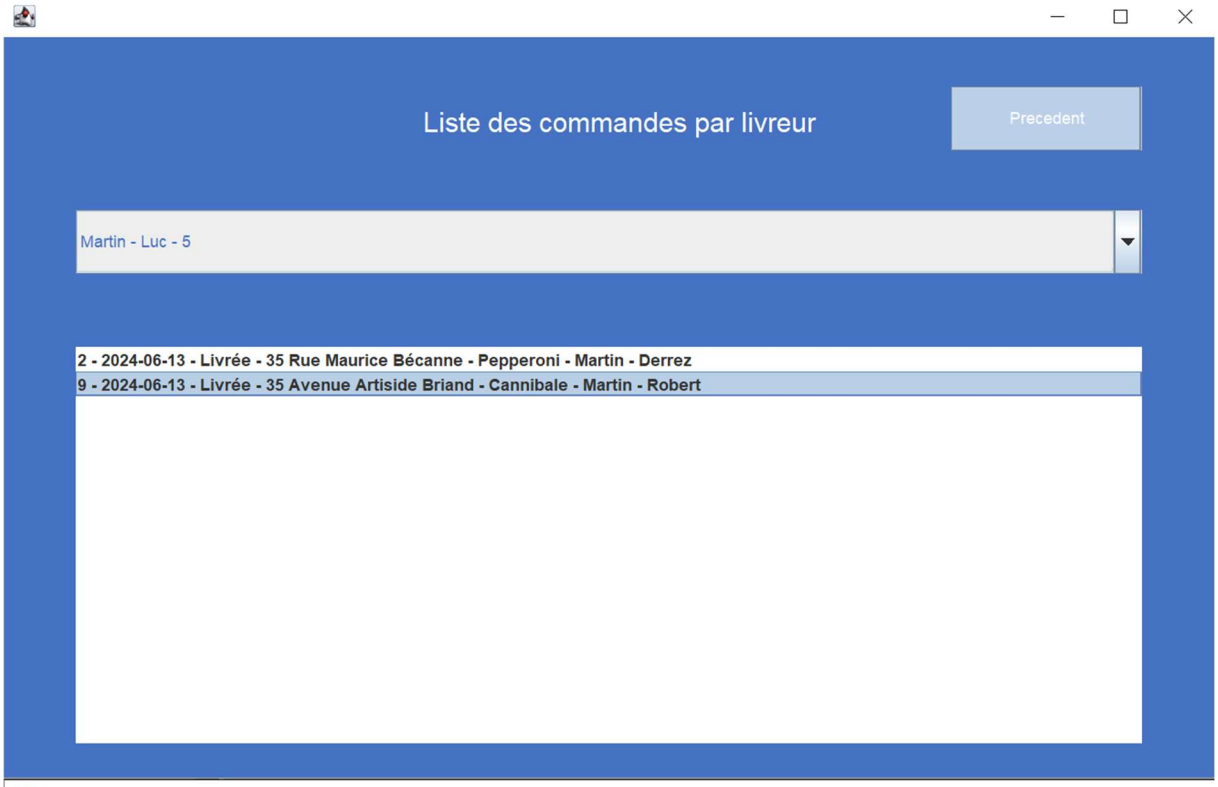
4.2. Structure du Projet

La structure du projet est organisée comme suit :

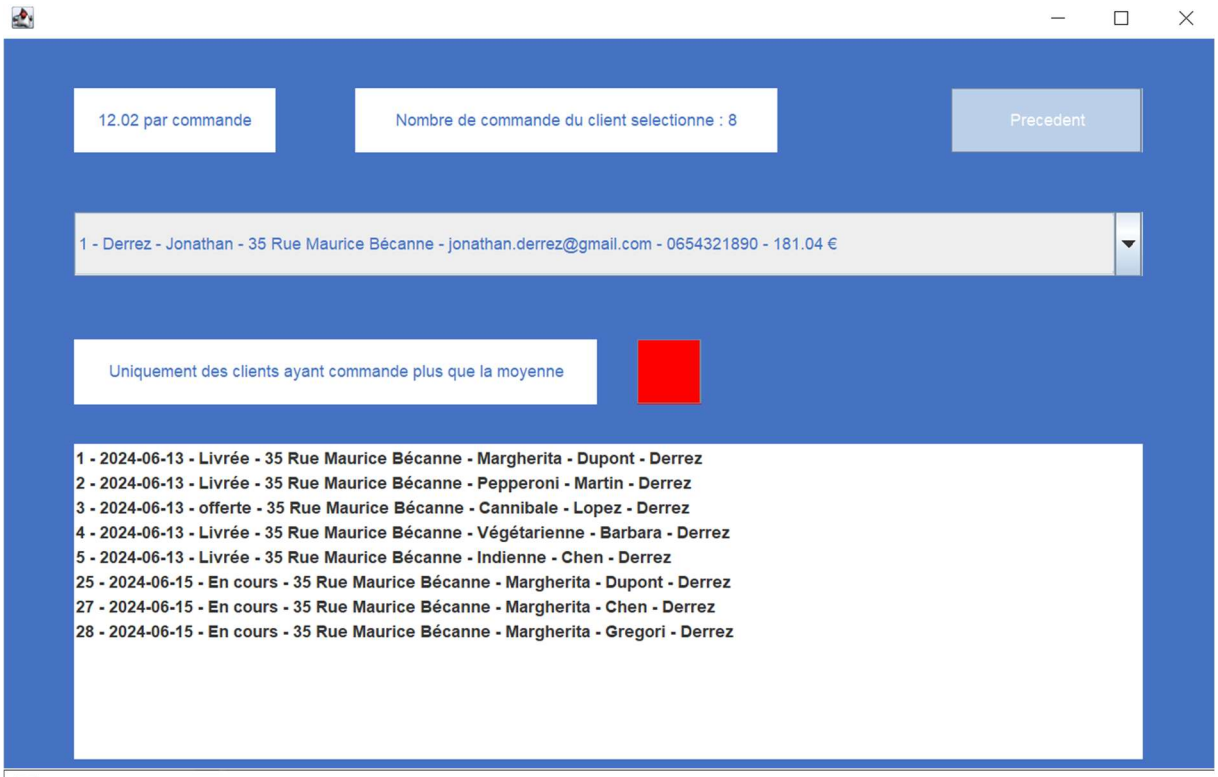
- **Controller** : Le contrôleur gère les interactions entre la vue et le modèle. Il contient la logique de l'application et répond aux actions de l'utilisateur. Le contrôleur utilise JDBC pour accéder à la base de données et exécuter des requêtes SQL.
    - main.java : Point d'entrée de l'application.
- **Model** : Le modèle contient les classes représentant les entités de la base de données ainsi que la logique de manipulation des données. Chaque classe correspond à une table de la base de données et contient des attributs et des méthodes pour accéder et manipuler les données.
    - CLIENT.java : Classe représentant un client.
    - COMMANDE.java : Classe représentant une commande.
    - INGREDIENTS.java : Classe représentant les ingrédients.
    - LIVREUR.java : Classe représentant un livreur.
    - PIZZA.java : Classe représentant une pizza.
    - TAILLE.java : Classe représentant les tailles de pizzas.
    - TYPEPIZZA.java : Classe représentant les types de pizzas.
    - VEHICULE.java : Classe représentant un véhicule.
    - model.java : Classe gérant les interactions avec la base de données via JDBC.
- **View** : L'interface utilisateur a été développée en utilisant Swing pour offrir une application facile à utiliser. Swing est une bibliothèque graphique de Java qui permet de créer des interfaces utilisateur avec une grande flexibilité.
    - accueil.java : Interface d'accueil de l'application.
    - info.java : Interface pour afficher les informations générales.
    - infoclient.java : Interface pour afficher les informations des clients.
    - infolivreur.java : Interface pour afficher les informations des livreurs.
    - menu.java : Interface pour afficher le menu des pizzas.

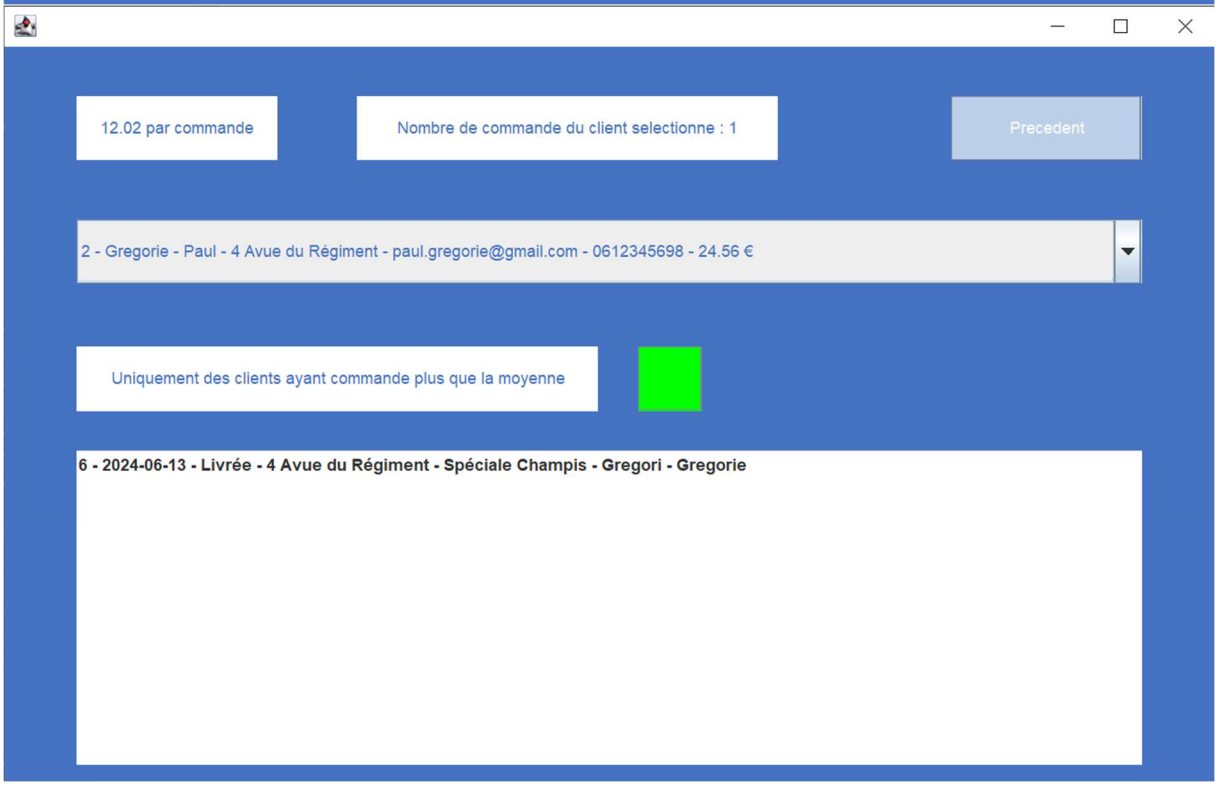

> pizza [pizza main]
JRE System Library [JavaSE-1.8]
> src
controller
main.java
> model
CLIENT.java
COMMANDE.java
INGREDIENTS.java
LIVREUR.java
> model.java
PIZZA.java
TAILLE.java
TYPEPIZZA.java
VEHICULE.java
view
accueil.java
info.java
infoclient.java
infolivreur.java
menu.java
miglayout-src.zip

5. Résultats Obtenus


Liste des commandes par livreur
Precedent
Martin - Luc - 5
2 - 2024-06-13 - Livrée - 35 Rue Maurice Bécanne - Pepperoni - Martin - Derrez
9 - 2024-06-13 - Livrée - 35 Avenue Artiside Briand - Cannibale - Martin - Robert


Precedent
Type3T1 - Cannibale - Naine - 8.65€ - Ingrédients: Poulet, Sauce Barbecue, Fromage, Viande Hach...
Type3T2 - Cannibale - Humaine - 12.35€ - Ingrédients: Poulet, Saucisses, Viande Hachée, Fromage,...
Type3T3 - Cannibale - Ogresse - 16.06€ - Ingrédients: Sauce Barbecue, Poulet, Saucisses, Viande H...
Type5T1 - Indienne - Naine - 7.69€ - Ingrédients: Sauce Blanche, Fromage, Poulet, Oignons, Champ...
Type5T2 - Indienne - Humaine - 10.99€ - Ingrédients: Sauce Blanche, Fromage, Poulet, Oignons, Ch...
Type5T3 - Indienne - Ogresse - 14.29€ - Ingrédients: Champignons, Sauce Blanche, Fromage, Poule...
Type1T1 - Margherita - Naine - 6.65€ - Ingrédients: Sauce Tomate, Fromage
Type1T2 - Margherita - Humaine - 9.5€ - Ingrédients: Sauce Tomate, Fromage
Type1T3 - Margherita - Ogresse - 12.35€ - Ingrédients: Sauce Tomate, Fromage
Type2T1 - Pepperoni - Naine - 7.53€ - Ingrédients: Fromage, Pepperoni, Sauce Tomate
Type2T2 - Pepperoni - Humaine - 10.75€ - Ingrédients: Pepperoni, Sauce Tomate, Fromage
Type2T3 - Pepperoni - Ogresse - 13.98€ - Ingrédients: Sauce Tomate, Pepperoni, Fromage
Type6T1 - Spéciale Champis - Naine - 7.69€ - Ingrédients: Oignons, Poivrons, Champignons, Sauce ...
Type6T2 - Spéciale Champis - Humaine - 10.99€ - Ingrédients: Fromage, Champignons, Poivrons, Sa...
Type6T3 - Spéciale Champis - Ogresse - 14.29€ - Ingrédients: Champignons, Oignons, Fromage, Sau...
Type4T1 - Végétarienne - Naine - 9.66€ - Ingrédients: Tomate, Oignons, Sauce Blanche, Poivrons
Type4T2 - Végétarienne - Humaine - 13.8€ - Ingrédients: Poivrons, Sauce Blanche, Tomate, Oignons
Type4T3 - Végétarienne - Ogresse - 17.94€ - Ingrédients: Tomate, Sauce Blanche, Oignons, Poivrons
Veuillez saisir votre identifiant client :
6
Validation
Votre solde est insuffisant


12.02 par commande
Nombre de commande du client selectionne : 8
Precedent
1 - Derrez - Jonathan - 35 Rue Maurice Bécanne - jonathan.derrez@gmail.com - 0654321890 - 181.04 €
Uniquement des clients ayant commande plus que la moyenne
1 - 2024-06-13 - Livrée - 35 Rue Maurice Bécanne - Margherita - Dupont - Derrez
2 - 2024-06-13 - Livrée - 35 Rue Maurice Bécanne - Pepperoni - Martin - Derrez
3 - 2024-06-13 - offerte - 35 Rue Maurice Bécanne - Cannibale - Lopez - Derrez
4 - 2024-06-13 - Livrée - 35 Rue Maurice Bécanne - Végétarienne - Barbara - Derrez
5 - 2024-06-13 - Livrée - 35 Rue Maurice Bécanne - Indienne - Chen - Derrez
25 - 2024-06-15 - En cours - 35 Rue Maurice Bécanne - Margherita - Dupont - Derrez
27 - 2024-06-15 - En cours - 35 Rue Maurice Bécanne - Margherita - Chen - Derrez
28 - 2024-06-15 - En cours - 35 Rue Maurice Bécanne - Margherita - Gregori - Derrez


12.02 par commande
Nombre de commande du client selectionne : 1
Precedent
2 - Gregorie - Paul - 4 Avue du Régiment - paul.gregorie@gmail.com - 0612345698 - 24.56 €
Uniquement des clients ayant commande plus que la moyenne
6 - 2024-06-13 - Livrée - 4 Avue du Régiment - Spéciale Champis - Gregori - Gregorie

6. Conclusion

En guise de conclusion, il paraît clair que ce projet a été très enrichissant. Il nous a permis de renforcer nos compétences en conception et gestion de bases de données ainsi qu'en développement d'applications en Java. La modélisation des processus de gestion d'une entreprise de livraison de pizzas a été une expérience pratique qui nous a permis d'appliquer des concepts théoriques à un cas concret. Les défis rencontrés et surmontés au cours de ce projet ont contribué à notre développement professionnel et nous ont préparés à aborder des projets plus complexes à l'avenir.

7. Annexes

Lien GitHub : https://github.com/michel-ch/pizza