RaPizz

Cadre du projet

Dans le cadre de l'unité **Base de données**, dispensée en 1ère année de cycle ingénieurs à l'ESIEE Paris en filière **Informatique et applications**, nous avons été amené à développer l'application **RaPizz**.

	Données	Commentaire
Nom du projet	`"RaPizz"`	Application de gestion de commandes de pizzas
Date de rendu	20 juin 2021	Date de rendu du projet
Technologies	Java, JavaFX, MySQL, JDBC	-

Description du projet

On veut modéliser la gestion d'une entreprise de fabrication et de **livraison de pizzas** à domicile : la société **RaPizz**. Il s'agit d'une société en franchise qui utilise des formats et des compositions de pizzas normalisés à partir d'un ensemble **d'ingrédients déterminés**. En d'autres termes, le client n'a pas la liberté de composer lui-même une pizza personnalisée ; il doit choisir dans le **catalogue** proposé.

Lien du sujet

Membres du projet

- Lucas BILLARD
- Ewen BOUQUET
- Fabien COURTOIS
- Loic FOURNIER

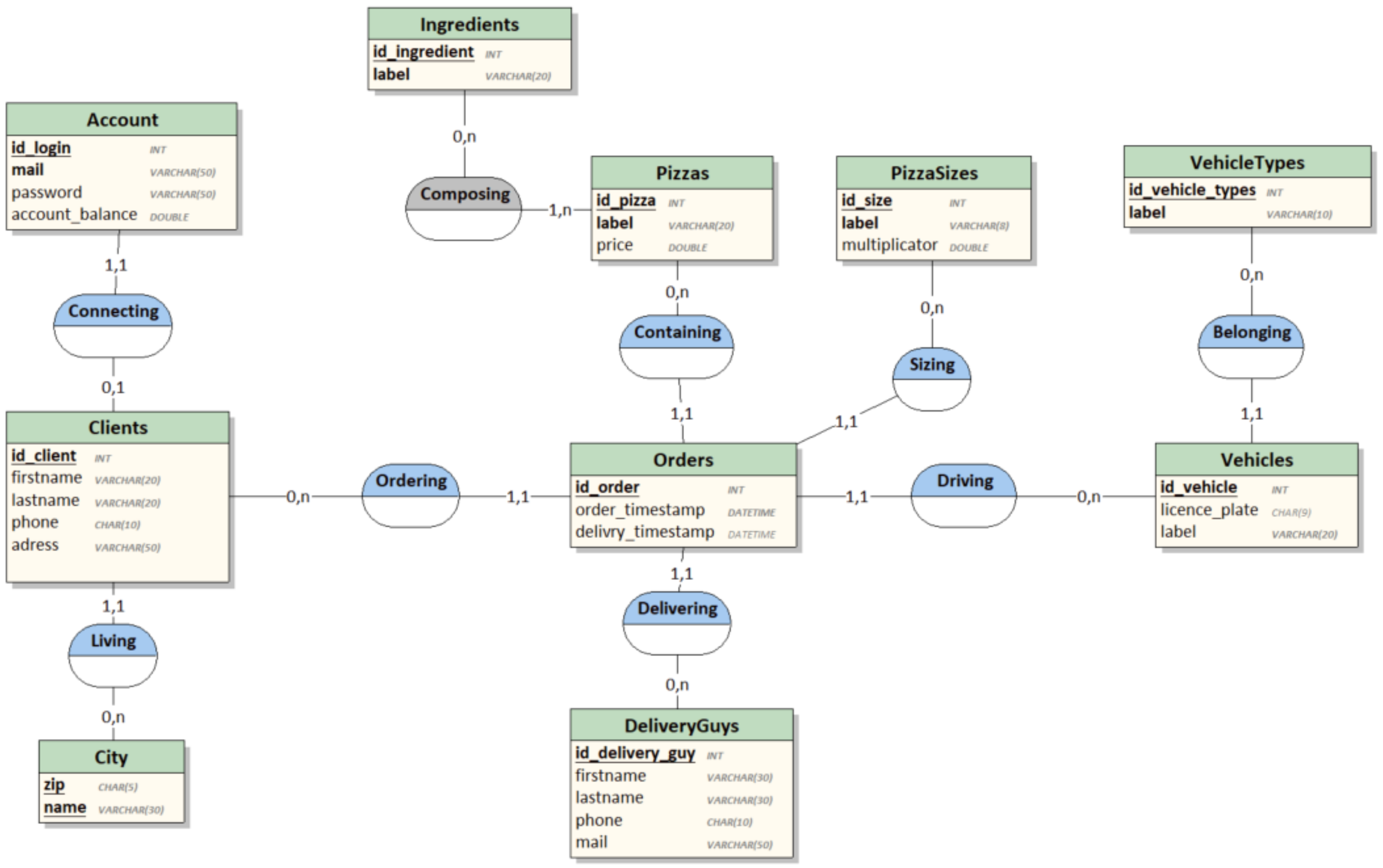
Base de données

Modèle Conceptuel de Données

Dans un premier temps, nous avons analysé le sujet et en avons déduit un **MCD**. En l'occurence, le schéma produit contient 10 entités et 9 associations.

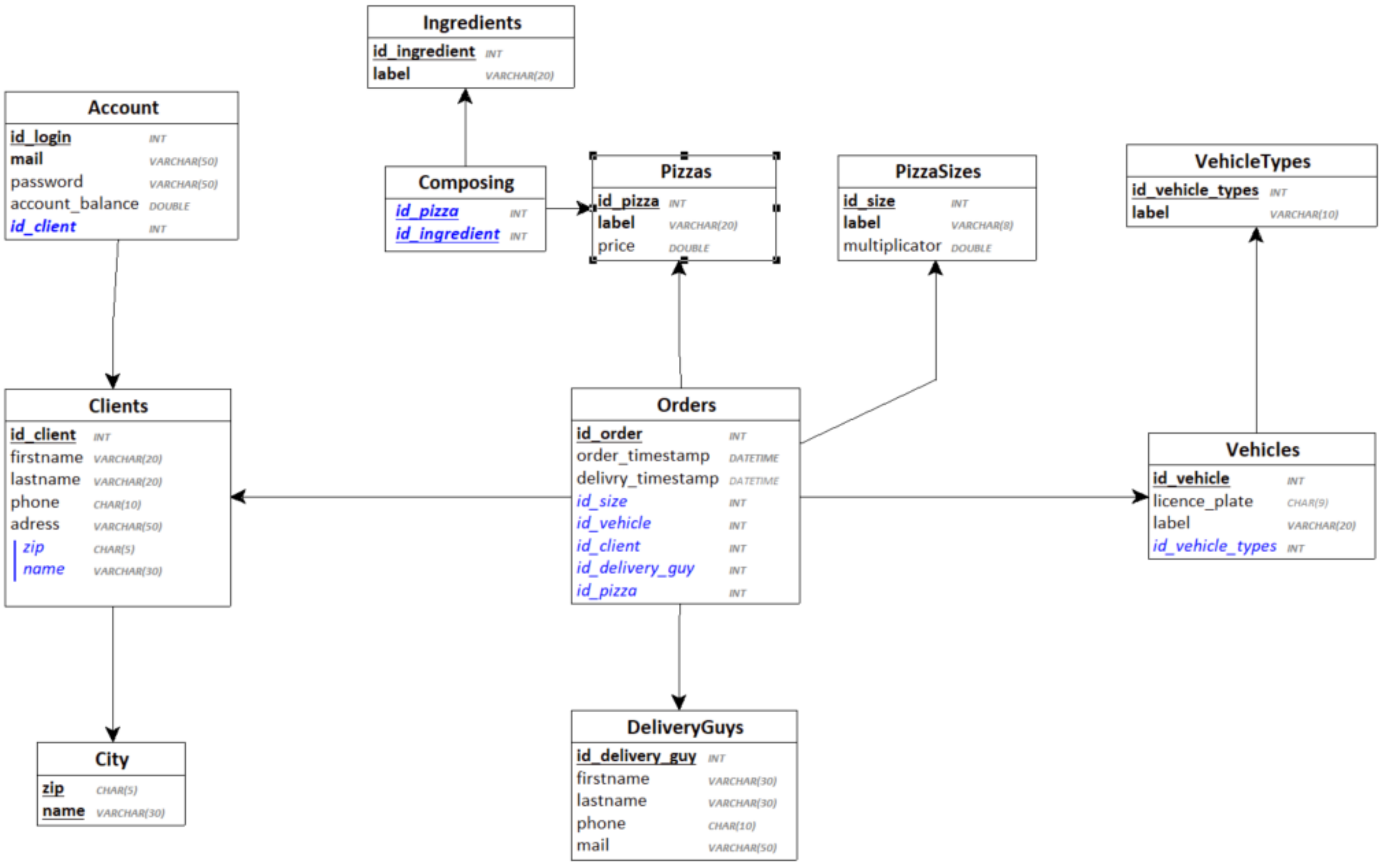
> Entités : `Account`, `Ingredients`, `Pizzas`, `PizzaSize`, `VehicleTypes`, `Clients`, `Orders`, `Vehicles`, `City`, et `DeliveryGuys` ;

> Associations : `Connecting`, `Composing`, `Containing`, `Sizing`, `Belonging`, `Ordering`, `Driving`, `Living`, et `DeliveryGuys` ;



Modèle Logique de Données

A partir du logiciel `Looping`, nous avons généré le MLD de la base de données.



Modèle Physique de Données

De même, nous avons déduit le MPD avec Looping.

```
CREATE TABLE Pizzas(
  id_pizza INT,
  label VARCHAR(20) NOT NULL,
  price DOUBLE NOT NULL,
  PRIMARY KEY(id_pizza),
  UNIQUE(label)
);

CREATE TABLE PizzaSizes(
  id_size INT,
  label VARCHAR(8) NOT NULL,
  multiplicator DOUBLE NOT NULL,
  PRIMARY KEY(id_size),
  UNIQUE(label)
);

CREATE TABLE Ingredients(
  id_ingredient INT,
  label VARCHAR(20) NOT NULL,
  PRIMARY KEY(id_ingredient),
  UNIQUE(label)
);
```

```sql
CREATE TABLE DeliveryGuys(
  id_delivery_guy INT,
  firstname VARCHAR(30) NOT NULL,
  lastname VARCHAR(30) NOT NULL,
  phone CHAR(10) NOT NULL,
  mail VARCHAR(50) NOT NULL,
  PRIMARY KEY(id_delivery_guy)
);

CREATE TABLE VehicleTypes(
  id_vehicle_types INT,
  label VARCHAR(10) NOT NULL,
  PRIMARY KEY(id_vehicle_types),
  UNIQUE(label)
);

CREATE TABLE City(
  zip CHAR(5),
  name VARCHAR(30),
  PRIMARY KEY(zip, name)
);

CREATE TABLE Vehicles(
  id_vehicle INT,
  licence_plate CHAR(9),
  label VARCHAR(20) NOT NULL,
  id_vehicle_types INT NOT NULL,
  PRIMARY KEY(id_vehicle),
  FOREIGN KEY(id_vehicle_types) REFERENCES VehicleTypes(id_vehicle_types)
);

CREATE TABLE Clients(
  id_client INT,
  firstname VARCHAR(20) NOT NULL,
  lastname VARCHAR(20) NOT NULL,
  phone CHAR(10) NOT NULL,
  adress VARCHAR(50) NOT NULL,
  zip CHAR(5) NOT NULL,
  name VARCHAR(30) NOT NULL,
  PRIMARY KEY(id_client),
  FOREIGN KEY(zip, name) REFERENCES City(zip, name)
);

CREATE TABLE Account(
  id_login INT,
  mail VARCHAR(50) NOT NULL,
  password VARCHAR(50) NOT NULL,
  account_balance DOUBLE NOT NULL,
  id_client INT NOT NULL,
  PRIMARY KEY(id_login),
  UNIQUE(id_client),
  UNIQUE(mail),
  FOREIGN KEY(id_client) REFERENCES Clients(id_client)
);
```

```sql
CREATE TABLE Orders(
  id_order INT,
  order_timestamp DATETIME NOT NULL,
  delivry_timestamp DATETIME,
  id_size INT NOT NULL,
  id_vehicle INT NOT NULL,
  id_client INT NOT NULL,
  id_delivery_guy INT NOT NULL,
  id_pizza INT NOT NULL,
  PRIMARY KEY(id_order),
  FOREIGN KEY(id_size) REFERENCES PizzaSizes(id_size),
  FOREIGN KEY(id_vehicle) REFERENCES Vehicles(id_vehicle),
  FOREIGN KEY(id_client) REFERENCES Clients(id_client),
  FOREIGN KEY(id_delivery_guy) REFERENCES DeliveryGuys(id_delivery_guy),
  FOREIGN KEY(id_pizza) REFERENCES Pizzas(id_pizza)
);

CREATE TABLE Composing(
  id_pizza INT,
  id_ingredient INT,
  PRIMARY KEY(id_pizza, id_ingredient),
  FOREIGN KEY(id_pizza) REFERENCES Pizzas(id_pizza),
  FOREIGN KEY(id_ingredient) REFERENCES Ingredients(id_ingredient)
);
```

Création de données

Après avoir importé la structure de la base de données dans MySQL, nous avons réalisé des données de tests.

```sql
INSERT  INTO  `account`
(`id_login`, `mail`, `password`, `account_balance`, `id_client`) VALUES
(1, 'ewen.bouquet@free.fr', 'c2fd602dba0e90c5418ce9fcbf0af9052e0e14b8', 1500.5, 1),
(2, 'fabien.courtois@gmail.com', 'af6cd4cbb67d2d73471dadd7005a23921c9bc829', 700.3, 2),
(3, 'loic.fournier@gmail.com', '57baf0938a84d48beabcd899514a14d0901b5927', 15, 3),
(4, 'serge.razionaff@gmail.com', '23e0f529d2146b2c9c5f50b5ad4ceafa4fa1e83b', 0, 4),
(5, 'lucas.billard@yahoo.fr', 'd89332e7bdab30358ae09713e494259d133541c7', 2, 5);

INSERT  INTO  `city`
(`zip`, `name`) VALUES
('77207', 'Torcy'),
('77400', 'Lagny-sur-Marne'),
('77427', 'Champs-sur-Marne'),
('77600', 'Bussy-Saint-Georges'),
('93161', 'Noisy-le-Grand'),
('93330', 'Neuilly-sur-Marne');

INSERT  INTO  `clients`
(`id_client`, `firstname`, `lastname`, `phone`, `adress`, `zip`, `name`) VALUES
(1, 'Ewen', 'Bouquet', '0781282363', '9 promenade Henri Xavier', '77600', 'Bussy-Saint-Georges
(2, 'Fabien', 'Courtois', '0781529456', '8 rue de la Fougère', '77427', 'Champs-sur-Marne'),
(3, 'Loïc', 'Fournier', '0782594678', '6 rue du Pamplemousse Sacré', '93330', 'Neuilly-sur-Mar
(4, 'Serge', 'Razianoff', '0794568271', '66 impasse Visotéra', '77427', 'Champs-sur-Marne'),
(5, 'Lucas', 'Billard', '0698452519', '36 rue du Kiwixi', '77600', 'Bussy-Saint-Georges');

INSERT  INTO  `composing`
(`id_pizza`, `id_ingredient`) VALUES
(1, 2),
(1, 3),
(1, 4),
(1, 5),
(1, 15),
(1, 34),
(2, 14),
(2, 15),
(2, 31),
(2, 32),
(3, 3),
(3, 4),
(3, 5);

INSERT  INTO  `deliveryguys`
(`id_delivery_guy`, `firstname`, `lastname`, `phone`, `mail`) VALUES
(1, 'Jean-Marie', 'Bigard', '0794531629', 'jean.heude@free.fr'),
(2, 'Jérémy', 'Ferrary', '0784956348', 'jeremy.ferrary@gmail.com'),
(3, 'Jean-Claude', 'Van Damme', '0794536148', 'jc.van-damme@gmail.com'),
(4, 'Kev', 'Adams', '0698462718', 'kev.adams@yahoo.fr'),
(5, 'Arnaud', 'Tsamere', '0798694827', 'arnaud.tsamere@free.fr');

INSERT  INTO  `ingredients`
(`id_ingredient`, `label`) VALUES
(23, 'Ail'),
(15, 'Basilic'),
(3, 'Champignon'),
(9, 'Chorizo'),
(14, 'Crème fraiche'),
(31, 'Fromage de chèvre'),
(20, 'Fromage râpée'),
(11, 'Gorgonzola'),
(34, 'Huile piquante'),
(8, 'Jambon'),
(26, 'Jambon de pays'),
(28, 'Maïs'),
(17, 'Merguez'),
(32, 'Miel'),
(5, 'Mozarella'),
(19, 'Oeuf'),
(2, 'Olives Noire'),
(1, 'Olives Verte'),
(22, 'Parmesan'),
(33, 'Persil'),
(21, 'Pomme de terre'),
(7, 'Poulet'),
(6, 'Ricotta'),
(27, 'Roquette'),
(12, 'Sauce barbecue'),
(13, 'Sauce chedar'),
(18, 'Sauce piquante'),
(4, 'Sauce tomate'),
(10, 'Saucisson'),
(25, 'Saumon fumée'),
(24, 'Sel'),
(30, 'Tomate cerise'),
(29, 'Tomate séchée'),
(16, 'Truffe');

INSERT  INTO  `orders`
(`id_order`, `order_timestamp`, `delivry_timestamp`, `id_size`, `id_vehicle`, `id_client`, `id
(1, '2021-06-10 18:40:13', '2021-06-10 19:00:13', 1, 1, 2, 1, 1),
(2, '2021-06-12 18:10:23', '2021-06-12 18:20:13', 2, 2, 2, 1, 1),
(3, '2021-06-10 18:22:45', '2021-06-10 18:35:13', 3, 5, 3, 5, 3),
(4, '2021-06-10 18:35:23', '2021-06-10 18:01:13', 1, 4, 1, 2, 3),
(5, '2021-06-10 18:40:53', '2021-06-10 19:30:13', 2, 5, 2, 3, 3),
(6, '2021-06-10 18:33:35', '2021-06-10 19:03:13', 3, 1, 5, 4, 3),
(7, '2021-06-10 18:32:48', '2021-06-10 19:12:13', 1, 2, 4, 4, 2),
(8, '2021-06-10 18:47:46', '2021-06-10 19:46:13', 2, 5, 1, 1, 2),
(9, '2021-06-10 18:46:18', '2021-06-10 19:32:13', 3, 4, 3, 2, 2),
(10, '2021-06-10 18:12:19', '2021-06-10 19:46:13', 1, 5, 1, 3, 2),
(11, '2021-06-10 18:43:56', '2021-06-10 19:06:13', 2, 1, 4, 5, 2),
(12, '2021-06-10 18:26:47', '2021-06-10 19:04:13', 3, 2, 2, 1, 3),
(13, '2021-06-10 18:46:36', '2021-06-10 19:12:13', 1, 3, 2, 5, 2),
(14, '2021-06-10 18:35:42', '2021-06-10 19:45:13', 2, 4, 5, 2, 3),
(15, '2021-06-10 18:22:43', '2021-06-10 19:58:13', 3, 5, 1, 1, 1);

INSERT  INTO  `pizzas`
(`id_pizza`, `label`, `price`) VALUES
(1, 'Regina', 10.5),
(2, 'Chèvre Miel', 12.5),
(3, 'Margherita', 8);

INSERT  INTO  `pizzasizes`
(`id_size`, `label`, `multiplicator`) VALUES
(1, 'naine', 0.6),
(2, 'humaine', 1),
(3, 'ogresse', 1.3);



INSERT  INTO  `vehicles`
(`id_vehicle`, `licence_plate`, `label`, `id_vehicle_types`) VALUES
(1, 'A1-A1A-1A', 'Renault 4L', 1),
(2, 'B1-A1A-1N', 'Peugeot 208', 1),
(3, 'B1-314-1N', 'Bugatti Chiron', 1),
(4, 'B1-4Z4-1N', 'Honda Forza 125', 2),
(5, '1Y-4D4-34', 'BMW R1200 ST', 2),
(6, '2Y-RD4-14', 'Volkswagen Arteon Sh', 1);

INSERT  INTO  `vehicletypes`
(`id_vehicle_types`, `label`) VALUES
(2, 'moto'),
(1, 'voiture');
```

Création de Fonctions et de Triggers

Afin de simplifier notre gestion du solde des clients dans notre Backend, nous avons implémenté dans la base de données 1 Trigger et 2 Fonctions. Le rôle du trigger est de décrémenter le solde du client en fonction du prix de sa commande et des critères de gratuité de l'entreprise.

```
DROP  FUNCTION  IF  EXISTS IsFreePizza;
DROP  FUNCTION  IF  EXISTS GetPizzaPrice;
DROP TRIGGER IF   EXISTS after_edit_order_timestamp;

DELIMITER //

-- IsFreePizza
CREATE  FUNCTION IsFreePizza(pOrderId INT, pClientId INT)
    RETURNS  INT
    BEGIN
        DECLARE isFreePizza INT;
        SET isFreePizza = (
            SELECT
            (
                (
                    -- Free 10 pizza
                    SELECT
                    (COUNT(r2_orde.id_order) % 10) = 0  AS free_10_pizza
                    FROM orders AS r2_orde
                    WHERE
                        r2_orde.id_client = pClientId
                )
                    OR
                (
                    -- Free late pizza
                    SELECT
                    (
                        DATEDIFF(r2_orde.delivry_timestamp, r2_orde.order_timestamp) >= 1
                            OR
                        (
                            DATEDIFF(r2_orde.delivry_timestamp, r2_orde.order_timestamp) = 0
                                AND
                            TIMEDIFF(r2_orde.delivry_timestamp, r2_orde.order_timestamp) >= "0
                        )
                    ) AS late_order
                    FROM orders AS r2_orde
                    WHERE
                        r2_orde.id_order = pOrderId
                )
            ) AS is_pizza_free
        );
        RETURN isFreePizza;
    END //
```

```
-- GetPizzaPrice
CREATE  FUNCTION GetPizzaPrice(pOrderId INT)
    RETURNS  INT
    BEGIN
        DECLARE pizzaPrice DOUBLE;
        SET pizzaPrice = (
          SELECT
              pizz.price * pisi.multiplicator AS price
          FROM orders AS orde
          JOIN pizzas AS pizz ON pizz.id_pizza = orde.id_pizza
          JOIN pizzasizes AS pisi ON pisi.id_size = orde.id_size
          WHERE
              orde.id_order = pOrderId
        );
        RETURN pizzaPrice;
    END //

-- After_edit_order_timestamp
CREATE TRIGGER after_edit_order_timestamp
AFTER  UPDATE  ON orders
FOR EACH ROW
    BEGIN
        DECLARE isFreePizza INT;
        DECLARE pizzaPrice DOUBLE;

        IF (NEW.delivry_timestamp IS  NOT  NULL) AND (OLD.delivry_timestamp IS  NULL) THEN

            SET isFreePizza = (SELECT IsFreePizza(NEW.id_order, NEW.id_client));

            IF (isFreePizza = 0) THEN

                    SET pizzaPrice = (SELECT GetPizzaPrice(NEW.id_order));

                    -- Pay the pizza
                    UPDATE account
                    SET account_balance = account_balance - pizzaPrice
                    WHERE
                        id_client = NEW.id_client;

            END  IF;
        END  IF;
    END //

DELIMITER ;
```

Création des requêtes SQL

Suite à la mise en place de notre base de données, nous avons développé des **requêtes SQL** afin de récolter des informations sur les données.

Connection à RaPizz

```
SELECT
    ac.id_client
FROM account AS ac
WHERE
    ac.mail = "ewen.bouquet@free.fr"
        AND
    ac.password = "c2fd602dba0e90c5418ce9fcbf0af9052e0e14b8";
```

Pizzas et leurs ingrédients

```
SELECT
    pizz.id_pizza, pizz.label, pizz.price, ingr.id_ingredi ingr.label
FROM pizzas         AS pizz
JOIN composing      AS comp     ON comp.id_pizza       = pizz.id_pizza
JOIN ingredients    AS ingr     ON comp.id_ingredient  = ingr.id_ingredient;
```

Les véhicules n'ayant jamais servi

```
SELECT
    DISTINCT(vehi.id_vehicle), vehi.label
FROM vehicles           AS vehi
LEFT JOIN orders        AS orde     ON orde.id_vehicle = vehi.id_vehicle
WHERE orde.id_vehicle IS NULL
GROUP BY vehi.id_vehicle;
```

Nombre de commandes par client

```
SELECT
    clien.id_client,
    clien.firstname,
    clien.lastname,
    COUNT(orde.id_order) AS `orders_nb`
FROM clients            AS clien
JOIN orders             AS orde     ON orde.id_client = clien.id_client
GROUP BY clien.id_client;
```

Moyenne de prix des commandes

```
SELECT
    AVG(pizz.price * size.multiplicator)
FROM orders             AS orde
JOIN pizzas             AS pizz     ON orde.id_pizza = pizz.id_pizza
JOIN pizzasizes              AS size     ON orde.id_size  = size.id_size;
```

Clients ayant commandé plus que la moyenne

```
SELECT
    r1_clien.id_client, r1_clien.firstname, r1_clien.lastname
FROM clients            AS r1_clien
WHERE
    EXISTS (
        SELECT r2_orde.id_order
            FROM orders         AS r2_orde
        JOIN pizzas             AS r2_pizz     ON r2_orde.id_pizza    = r2_pizz.id_pizza
        JOIN pizzasizes              AS r2_size     ON r2_orde.id_size     = r2_size.id_size
        WHERE
            r2_orde.id_client = r1_clien.id_client
                AND
            r2_pizz.price * r2_size.multiplicator > (
                SELECT
                    AVG(r3_pizz.price * r3_size.multiplicator)
                FROM orders             AS r3_orde
                JOIN pizzas             AS r3_pizz     ON r3_orde.id_pizza = r3_pizz.id_pizza
                JOIN pizzasizes         AS r3_size     ON r3_orde.id_size  = r3_size.id_size
            )
    );
```

Suivi du chiffre d’affaires

```
SELECT
    DATE_FORMAT(orde.delivry_timestamp, "%Y/%d"),
    SUM(pizz.price * pisi.multiplicator) AS turnover
FROM orders          AS orde
JOIN pizzas          AS pizz     ON pizz.id_pizza = orde.id_pizza
JOIN pizzasizes      AS pisi     ON pisi.id_size  = orde.id_size
GROUP BY DATE_FORMAT(orde.delivry_timestamp, "%Y/%d");
```

Refus d’honorer les commandes pour lesquelles le solde du compte client est insuffisant

```
SELECT
    0 <= (
        (
            -- Client account
            SELECT
                acco.account_balance
            FROM account AS acco
            WHERE
                acco.id_client = 1
        )
```

```
            -
        (
            -- Order price
            SELECT
                pizz.price * pisi.multiplicator AS price
            FROM orders         AS orde
            JOIN pizzas         AS pizz     ON pizz.id_pizza = orde.id_pizza
            JOIN pizzasizes     AS pisi     ON pisi.id_size  = orde.id_size
            WHERE
                orde.id_order = 1
        )
    ) AS can_buy;
```

Non-facturation des pizzas gratuites (retard ou fidélité)

```
SELECT
    (
        (
            -- Free 10 pizza
            SELECT
                (
                    COUNT(r2_orde.id_order) % 10
                ) = 0 AS free_10_pizza
            FROM orders                 AS r2_orde
            WHERE
                r2_orde.id_client = 1
        )
            OR
        (
            -- Free late pizza
            SELECT
                (
                    DATEDIFF(r2_orde.delivry_timestamp, r2_orde.order_timestamp) >= 1
                        OR
                    (
                        DATEDIFF(r2_orde.delivry_timestamp, r2_orde.order_timestamp) = 0
                            AND
                        TIMEDIFF(r2_orde.delivry_timestamp, r2_orde.order_timestamp) >= "00:30:
                    )
                ) AS late_order
            FROM orders                 AS r2_orde
            WHERE
                r2_orde.id_order = 1
        )
    ) AS is_pizza_free;
```

Identification du plus mauvais livreur et de la voiture utilisée

```sql
SELECT
    r2_degu.id_delivery_guy,
    r2_degu.firstname,
    r2_degu.lastname,
    r2_vehi.id_vehicle,
    r2_vehi.label,
    r2_vehi.licence_plate,
    COUNT(*) AS late_orders_nb
FROM deliveryguys   AS r2_degu
JOIN orders         AS r2_orde      ON r2_orde.id_delivery_guy = r2_degu.id_delivery_guy
JOIN vehicles       AS r2_vehi      ON r2_orde.id_vehicle      = r2_vehi.id_vehicle
WHERE
    DATEDIFF(r2_orde.delivry_timestamp, r2_orde.order_timestamp) >= 1
        OR
    (
        DATEDIFF(r2_orde.delivry_timestamp, r2_orde.order_timestamp) = 0
            AND
        TIMEDIFF(r2_orde.delivry_timestamp, r2_orde.order_timestamp) >= "00:30:00"
    )
GROUP BY r2_degu.id_delivery_guy
ORDER BY COUNT(*) DESC
LIMIT 1;
```

Identification de l’ingrédient favori

```sql
SELECT
    *,
    COUNT(*) AS order_nb
FROM ingredients    AS ingr
JOIN composing      AS comp     ON comp.id_ingredient = ingr.id_ingredient
JOIN pizzas         AS pizz     ON pizz.id_pizza      = comp.id_pizza
JOIN orders         AS orde     ON orde.id_pizza      = pizz.id_pizza
GROUP BY ingr.id_ingredient
ORDER BY COUNT(*) DESC
LIMIT 1;
```

Informations de commande

```sql
SELECT
    ord.id_order, ord.order_timestamp, ord.delivry_timestamp, pizza.id_pizza,
    pizza.label, pizza.price, size.id_size, size.label, deli.id_delivery_guy,
    deli.firstname, deli.lastname, vehi.id_vehicle, vehi.licence_plate,
    vehi.label, client.id_client, client.firstname, client.lastname
FROM
    orders AS ord
JOIN pizzas       AS pizza  ON pizza.id_pizza = ord.id_pizza
JOIN pizzasizes   AS size   ON size.id_size = ord.id_size
JOIN deliveryguys AS deli   ON deli.id_delivery_guy = ord.id_delivery_guy
JOIN vehicles     AS vehi   ON vehi.id_vehicle = ord.id_vehicle
JOIN clients      AS client ON client.id_client = ord.id_client
```

Tous les clients ayant acheté toutes les pizzas (division)

```
SELECT * FROM clients AS clien_0
WHERE NOT EXISTS (
    -- Les id de pizzas non commandées par le client
    (
        -- Toutes id de pizzas
        SELECT pizza_1.id_pizza
        FROM pizzas AS pizza_1
    )
        EXCEPT
    (
        -- Tous id de pizzas commandées par 1 client
        SELECT DISTINCT(order_1.id_pizza)
        FROM orders AS order_1
        WHERE order_1.id_client = clien_0.id_client
    )
);
```

Meilleur Client

```
-- Meilleur Client
SELECT
    clie.firstname, clie.lastname, orde.id_client, COUNT(orde.id_client) AS nbOrders
FROM
    orders orde
INNER JOIN clients clie ON
    orde.id_client = clie.id_client
GROUP BY
    orde.id_client
ORDER BY
    nbOrders
DESC
LIMIT 1;
```

Meilleure pizza

```
SELECT
    pizz.label,
     orde.id_pizza,
    COUNT( orde.id_pizza) AS nbOrders
FROM
    orders  orde
INNER JOIN pizzas pizz ON
    orde.id_pizza = pizz.id_pizza
```

```
GROUP BY
    orde.id_pizza
ORDER BY
    nbOrders
DESC
LIMIT 1;
```

Pire pizza

```
SELECT
    pizz.label,
     orde.id_pizza,
    COUNT( orde.id_pizza) AS nbOrders
FROM
    orders  orde
INNER JOIN pizzas pizz ON
    orde.id_pizza = pizz.id_pizza
GROUP BY
    orde.id_pizza
ORDER BY
    nbOrders
ASC
LIMIT 1;
```

Vérification du solde

```
SELECT
    acco.account_balance
FROM
    account acco
WHERE
    acco.id_client = 1;
```

Facturation de la commande

```
UPDATE account acco
SET
    acco.account_balance = acco.account_balance -
    (
        -- Order price
        SELECT
            pizz.price * pisi.multiplicator AS price
        FROM
            orders AS orde
        JOIN pizzas AS pizz
        ON
            pizz.id_pizza = orde.id_pizza
        JOIN pizzasizes AS pisi
```

```sql
            ON
                pisi.id_size = orde.id_size
            WHERE
                orde.id_order = 1 -- => $orderId
        )
    WHERE
        acco.id_client = 1; -- => $clientId
```

Vehicules libres

```sql
SELECT
    vehi.licence_plate, vehi.label
FROM
    orders orde
INNER JOIN vehicles vehi ON
    orde.id_vehicle = vehi.id_vehicle
WHERE
    orde.id_vehicle NOT IN(
    SELECT
        vehi.id_vehicle
    FROM
        orders orde
    INNER JOIN vehicles vehi ON
        orde.id_vehicle = vehi.id_vehicle
    WHERE
        orde.delivry_timestamp IS NULL
    GROUP BY
        vehi.id_vehicle
)
GROUP BY
    vehi.id_vehicle;
```

Livreurs libres

```sql
SELECT
    deli.*
FROM
    orders orde
INNER JOIN deliveryguys deli ON
    orde.id_delivery_guy = deli.id_delivery_guy
WHERE
    orde.id_delivery_guy NOT IN(
    SELECT
        deli.id_delivery_guy
    FROM
        orders orde
    INNER JOIN deliveryguys deli ON
        orde.id_delivery_guy = deli.id_delivery_guy
    WHERE
```

```
            orde.delivry_timestamp IS NULL
        GROUP BY
            deli.id_delivery_guy
    )
    GROUP BY
        deli.id_delivery_guy;
```

Insertion d'une commande

```
INSERT INTO `orders`(
    `id_order`,
    `order_timestamp`,
    `delivry_timestamp`,
    `id_size`,
    `id_vehicle`,
    `id_client`,
    `id_delivery_guy`,
    `id_pizza`
)
VALUES(
    NULL,
    NOW(),
    NULL,
    id_size,
    id_vehicle,
    id_client,
    id_delivery_guy,
    id_pizza
);
```

Interface Homme Machine

Suite à l'implémentation des requêtes SQL, nous avons réalisé une **IHM** permettant de gérer les commandes de pizzas de RaPizz.

> Passer une commande ;

> Visualiser les commandes ;

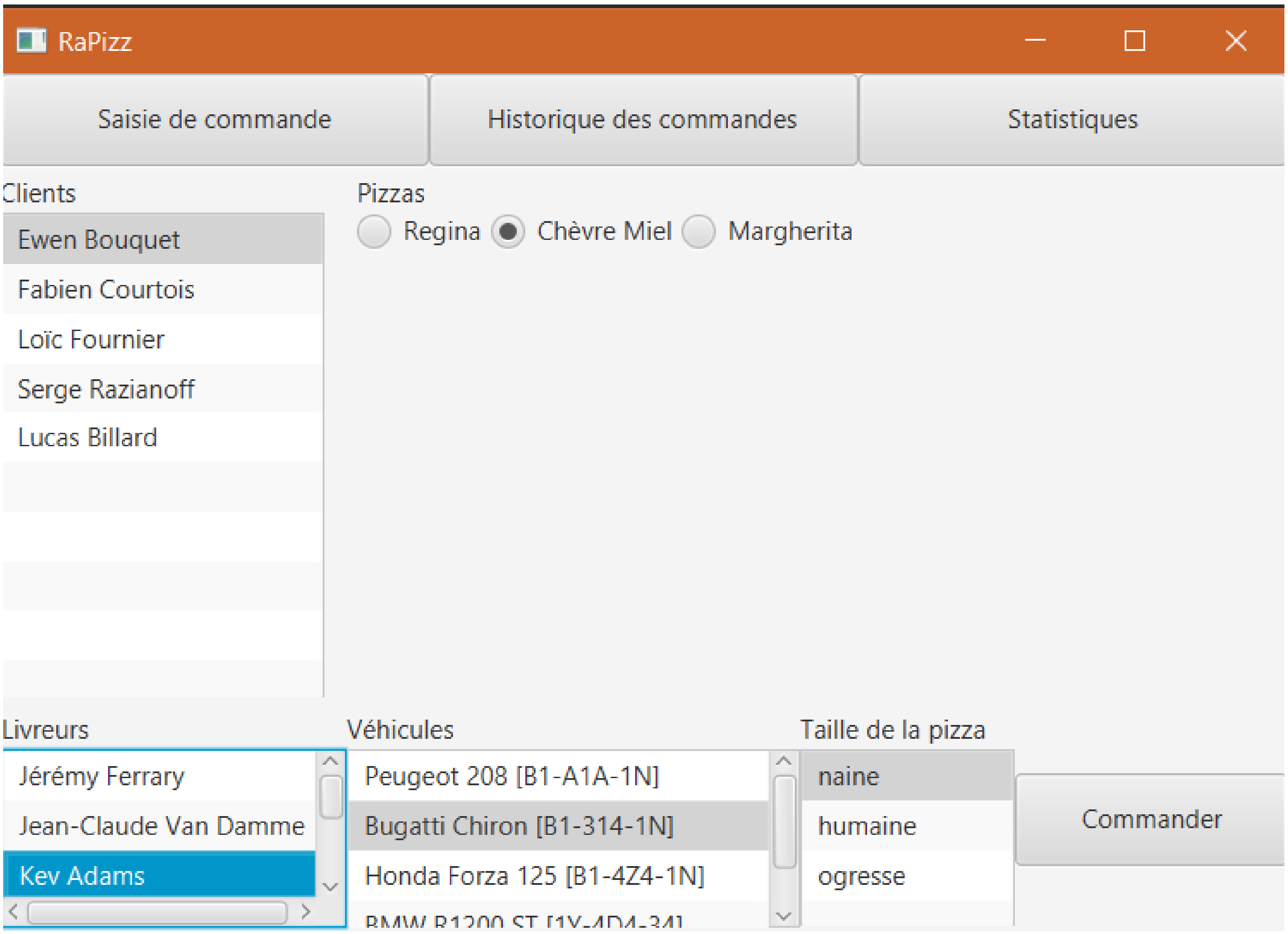
> Afficher les statistiques de la pizzeria ;

Passage de commande

Cette page permet d'effectuer une commande :

- Saisie du client ;
- Saisie de la pizza commandée et de sa taille;

- Saisie du livreur et du véhicule de livraison ;



Clients
- Ewen Bouquet
- Fabien Courtois
- Loïc Fournier
- Serge Razianoff
- Lucas Billard

Pizzas: Regina [✓ Chèvre Miel] Margherita

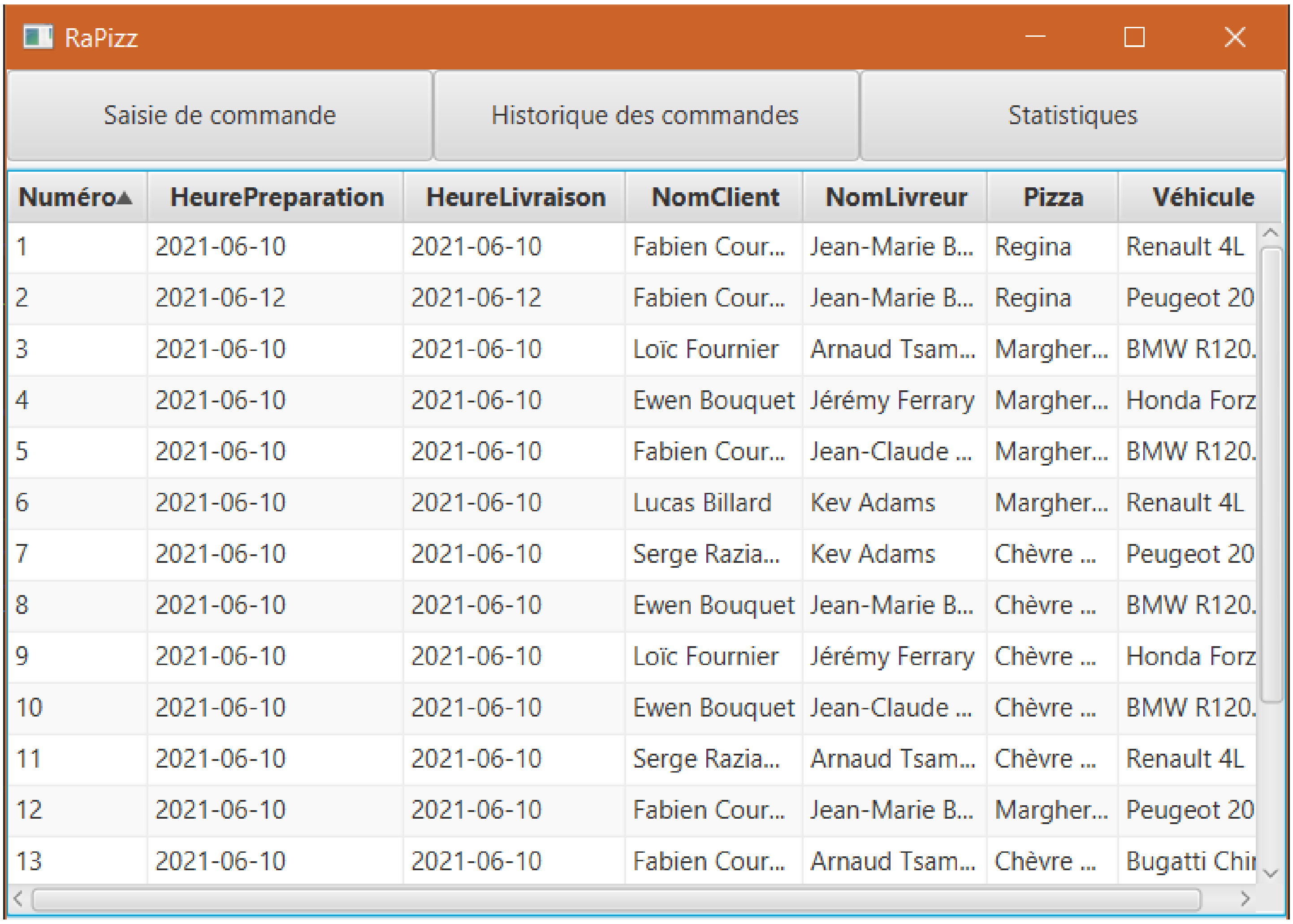
Livreurs
- Jérémy Ferrary
- Jean-Claude Van Damme
- Kev Adams

Véhicules
- Peugeot 208 [B1-A1A-1N]
- Bugatti Chiron [B1-314-1N]
- Honda Forza 125 [B1-4Z4-1N]
- BMW R1200 ST [1Y-4D4-34]

Taille de la pizza
- naine
- humaine
- ogresse

Visualisation des commandes

Cette page permet de visualiser les différentes commandes.

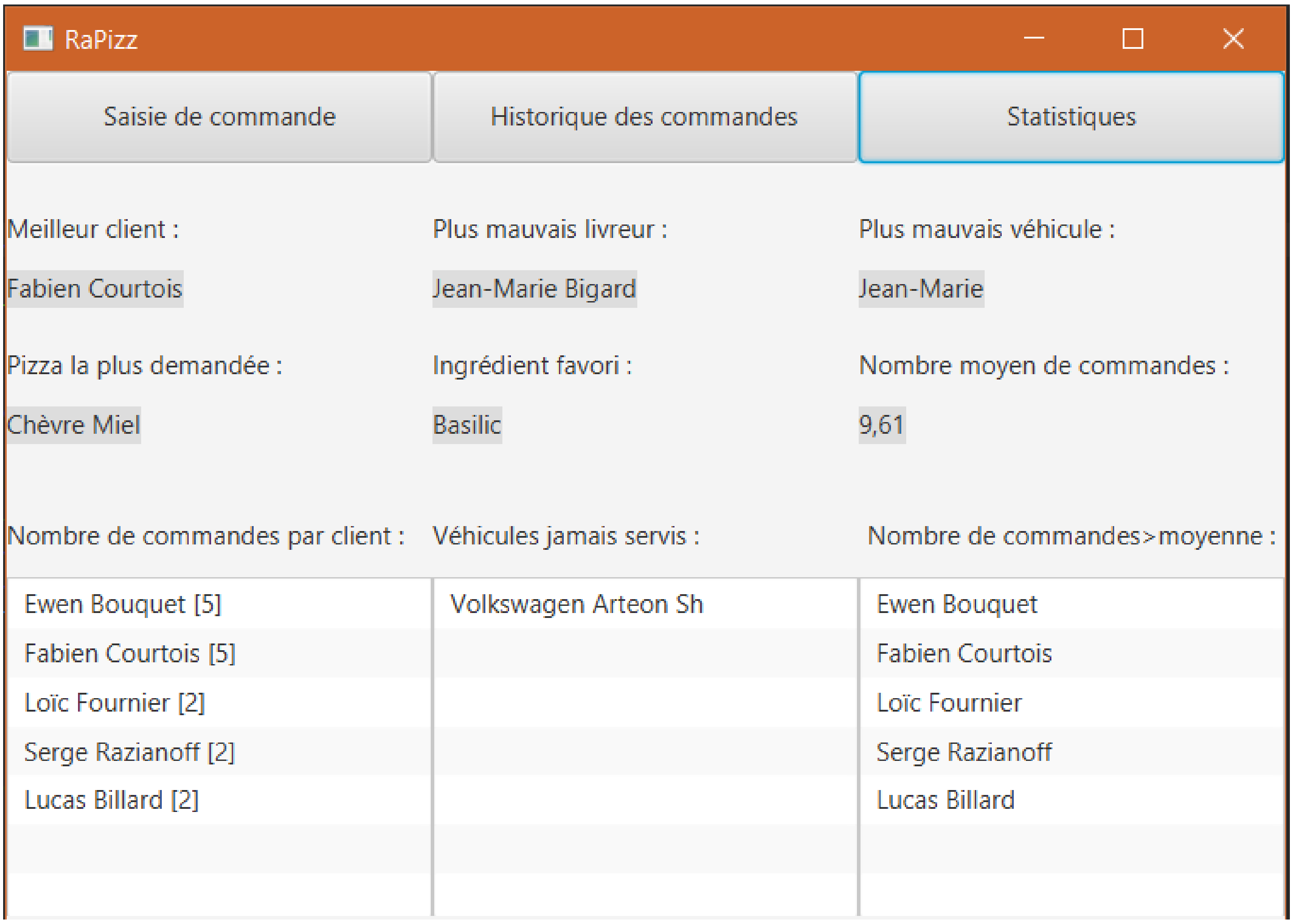


Numéro▲	HeurePreparation	HeureLivraison	NomClient	NomLivreur	Pizza	Véhicule
1	2021-06-10	2021-06-10	Fabien Cour...	Jean-Marie B...	Regina	Renault 4L
2	2021-06-12	2021-06-12	Fabien Cour...	Jean-Marie B...	Regina	Peugeot 20
3	2021-06-10	2021-06-10	Loïc Fournier	Arnaud Tsam...	Margher...	BMW R120.
4	2021-06-10	2021-06-10	Ewen Bouquet	Jérémy Ferrary	Margher...	Honda Forz
5	2021-06-10	2021-06-10	Fabien Cour...	Jean-Claude ...	Margher...	BMW R120.
6	2021-06-10	2021-06-10	Lucas Billard	Kev Adams	Margher...	Renault 4L
7	2021-06-10	2021-06-10	Serge Razia...	Kev Adams	Chèvre ...	Peugeot 20
8	2021-06-10	2021-06-10	Ewen Bouquet	Jean-Marie B...	Chèvre ...	BMW R120.
9	2021-06-10	2021-06-10	Loïc Fournier	Jérémy Ferrary	Chèvre ...	Honda Forz
10	2021-06-10	2021-06-10	Ewen Bouquet	Jean-Claude ...	Chèvre ...	BMW R120.
11	2021-06-10	2021-06-10	Serge Razia...	Arnaud Tsam...	Chèvre ...	Renault 4L
12	2021-06-10	2021-06-10	Fabien Cour...	Jean-Marie B...	Margher...	Peugeot 20
13	2021-06-10	2021-06-10	Fabien Cour...	Arnaud Tsam...	Chèvre ...	Bugatti Chir

Statistiques de l'application

Cette page permet d'afficher les statistiques de l'application, à savoir :

- Le meilleur client ;
- Le plus mauvais livreur ;
- Le plus mauvais véhicule ;
- La pizza la plus demandée ;
- L'ingrédient favori ;
- Le nombre moyen de commandes par personnes ;
- Le nombre de commandes par client ;
- Les véhicules n'ayant jamais servi ;
- Le nombre de commandes supérieures à la moyenne ;



Meilleur client :
Fabien Courtois

Plus mauvais livreur :
Jean-Marie Bigard

Plus mauvais véhicule :
Jean-Marie

Pizza la plus demandée :
Chèvre Miel

Ingrédient favori :
Basilic

Nombre moyen de commandes :
9,61

Nombre de commandes par client :	Véhicules jamais servis :	Nombre de commandes>moyenne :
Ewen Bouquet [5]	Volkswagen Arteon Sh	Ewen Bouquet
Fabien Courtois [5]		Fabien Courtois
Loïc Fournier [2]		Loïc Fournier
Serge Razianoff [2]		Serge Razianoff
Lucas Billard [2]		Lucas Billard

Connexion entre Java et la Base de données avec JDBC

Nous avons mis en place un singleton appelé `SqlManager` pour centraliser la gestion des accès à la base de données.

Connexion à JDBC

```
private static final String username = "root";
private static final String password = "";
private static final String serverName = "localhost";
private static final String database = "rapizz";
private static final int port = 3306;

Context context;
MysqlDataSource dataSource;
Connection con;

// singleton pattern
private SQLManager() {
    try {
        dataSource = new MysqlDataSource();
```

```
            dataSource.setUser(username);
            dataSource.setPassword(password);
            dataSource.setServerName(serverName);
            dataSource.setDatabaseName(database);
            dataSource.setPort(port);
            dataSource.setServerTimezone("UTC");
            System.out.println("Tentative de connexion...");
            con = dataSource.getConnection();
            System.out.println("Connection = " + con);

        } catch (SQLException e) {
            System.err.println("[ERROR]");
            System.err.println("MySQL Connexion exception: " + e);
        }
    }

    private Connection getCon() {
        try {
            return dataSource.getConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }

    private static SQLManager INSTANCE;

    public static SQLManager getInstance() throws SQLException {
        if (INSTANCE == null) {
            INSTANCE = new SQLManager();
        }
        return INSTANCE;
    }
```

Utilisation de JDBC pour récupérer des données

Dans cette fonction, nous réupérons le **meilleur client** via une requête SQL appliqué à la **base de données**.

```
    public Client bestClient() throws SQLException {

        String requestString = "SELECT\r\n"
                + "    clie.firstname,\r\n"
                + "    clie.lastname,\r\n"
                + "    orde.id_client,\r\n"
                + "    COUNT(orde.id_client) AS nbOrders\r\n"
                + "FROM\r\n"
                + "    orders orde\r\n"
                + "INNER JOIN clients clie ON\r\n"
                + "    orde.id_client = clie.id_client\r\n"
                + "GROUP BY\r\n"
                + "    orde.id_client\r\n"
                + "ORDER BY\r\n"
                + "    nbOrders\r\n"
                + "DESC\r\n"
                + "LIMIT 1;";

        PreparedStatement pStatement = getCon().prepareStatement(requestString);

        ResultSet rSet = pStatement.executeQuery();

        if(rSet.next()) {

            String firstname = rSet.getNString(1);
            String lastname = rSet.getNString(2);
            int id = rSet.getInt(3);
            return new Client(id,firstname,lastname);
        }

        return null;

    }
```

Utilisation de JDBC pour insérer des valeurs dans la base de donnée

```
    public boolean insertOrder(Pizza pizza, Client client, Vehicle vehicle, DeliveryGuy deliveryGu

        final String request =  "INSERT INTO `orders`(\r\n"
            + "    `id_order`,\r\n"
            + "    `order_timestamp`,\r\n"
            + "    `delivry_timestamp`,\r\n"
            + "    `id_size`,\r\n"
            + "    `id_vehicle`,\r\n"
            + "    `id_client`,\r\n"
            + "    `id_delivery_guy`,\r\n"
            + "    `id_pizza`\r\n"
            + ")\r\n"
            + "VALUES(\r\n"
            + "    NULL,\r\n"
            + "    NOW(), \r\n"
            + "    NULL, \r\n"
            + "    ?, \r\n"
            + "    ?, \r\n"
            + "    ?, \r\n"
            + "    ?,\r\n"
            + "    ?\r\n"
            + ");";

        PreparedStatement s = getCon().prepareStatement(request);
        s.setInt(1, size.getId());
        s.setInt(2, vehicle.getId());
        s.setInt(3, client.getId());
        s.setInt(4, deliveryGuy.getId());
        s.setInt(5, pizza.getId());
        return !s.execute();
    }
```

Utilisation de modèles pour renvoyer les données

Nous avons utilisé des **classes modèles** afin de **formatter** de manière uniforme les données récupérées suite à une requête SQL. Par exemple, la classe `Client` représente une ligne de la table `clients`.

```
package model;

public class Client {

    // Champs privés
    private final int id;
    private final String firstName;
    private final String lastName;
    private final String phone;
    private final String adress;

    // Constructeurs
    public Client(int id, String firstName, String lastName) {
        this(id,firstName,lastName,null,null);
    }

    public Client(int id, String firstName, String lastName, String phone,String adress) {
        super();
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.phone = phone;
        this.adress = adress;
    }

    // Getters
    public int getId() {
        return id;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFullName() {
        return getFirstName()+" "+getLastName();
    }
    // Hash code
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((adress == null) ? 0 : adress.hashCode());
        result = prime * result + ((firstName == null) ? 0 : firstName.hashCode());
        result = prime * result + id;
        result = prime * result + ((lastName == null) ? 0 : lastName.hashCode());
        result = prime * result + ((phone == null) ? 0 : phone.hashCode());
        return result;
    }

    // Equals
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Client other = (Client) obj;
        if (adress == null) {
            if (other.adress != null)
                return false;
        } else if (!adress.equals(other.adress))
            return false;
        if (firstName == null) {
            if (other.firstName != null)
                return false;
        } else if (!firstName.equals(other.firstName))
            return false;
        if (id != other.id)
            return false;
        if (lastName == null) {
            if (other.lastName != null)
                return false;
        } else if (!lastName.equals(other.lastName))
            return false;
        if (phone == null) {
            if (other.phone != null)
                return false;
        } else if (!phone.equals(other.phone))
            return false;
        return true;
    }

    // To string
    @Override
    public String toString() {
        return firstName + " " + lastName;
    }
    }
```

Accès au projet complet

Vous trouverez associé à ce pdf un zip contenant l'ensemble du **code du projet** développé sous Eclipse ainsi que les **requêtes SQL**.

Le projet est également diponible sur Github.

Membres du projet

- Lucas BILLARD
- Ewen BOUQUET
- Fabien COURTOIS
- Loic FOURNIER