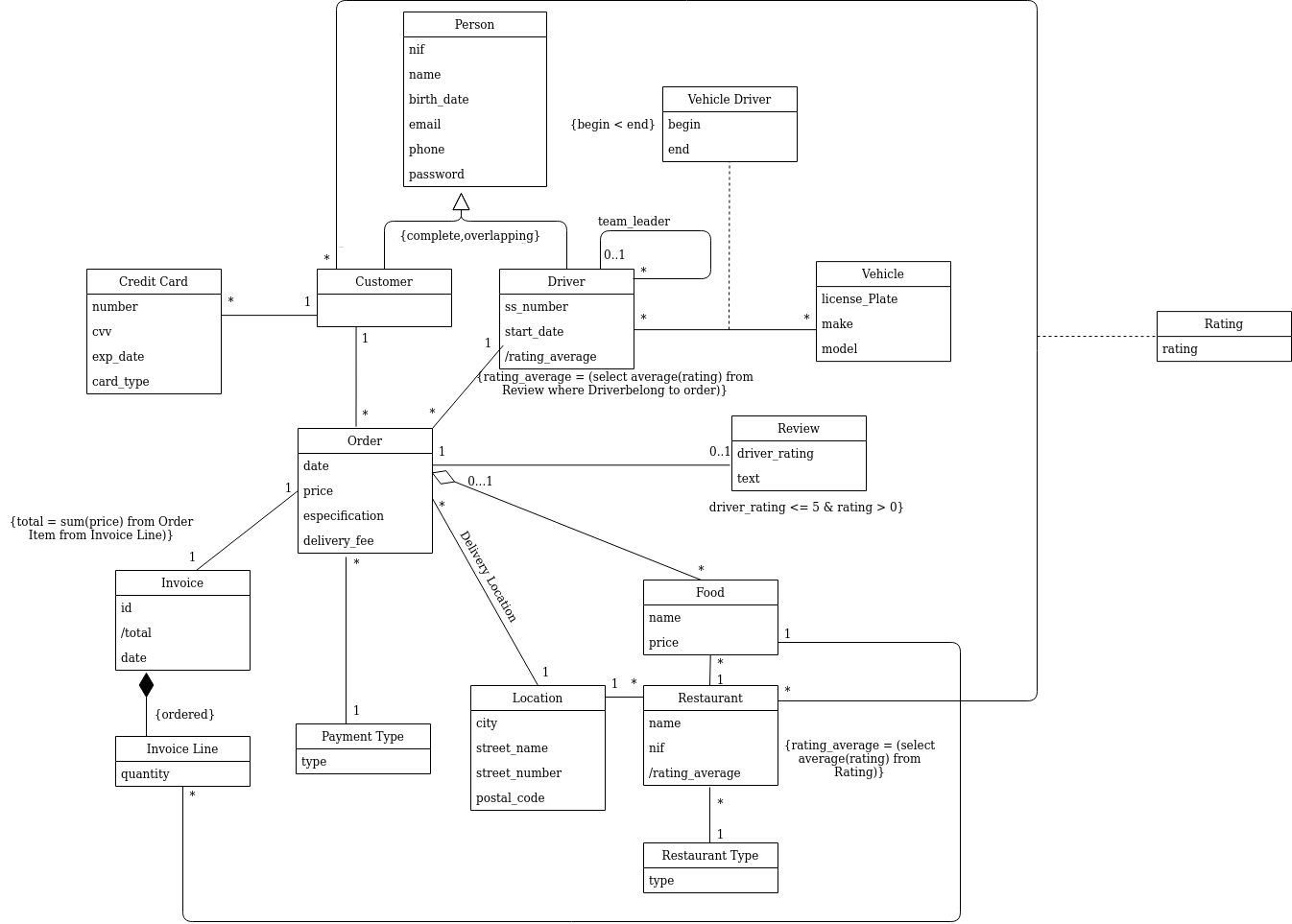
BDAD – Relatório: Empresa de Entregas

João Martins, José Miguel Maçães e Miguel Charchalis

**Esquema UML:**



**Contexto:**

De cada cliente interessa saber o nome, o NIF, o email, a data de nascimento, o número de telemóvel e a palavra passe. Isto tudo também é informação necessária relativa a cada condutor, para além do número de segurança social e a data em que começou a trabalhar na plataforma. Para além disso interessa saber as horas de início e de fim da sessão de trabalho do condutor.

Cada cliente necessita de ter um cartão de crédito, do qual deve ser conhecido o número, o CVV, a data de validade e a rede (VISA, MasterCard, etc.)

Cada condutor deve ter a si associado um veículo, identificado pela matrícula, do qual interessa saber a marca e o modelo. Um condutor pode ser chefe de equipa, sendo responsável por outros condutores.

O cliente pode efetuar o pedido, que será entregue por um condutor e é constituído por uma data, um modo de pagamento e um preço, calculado a partir do preço da comida e da taxa de entrega. Para além disso terá uma avaliação que terá uma classificação, entre 1 e 5, e poderá ou não ter um texto. As classificações são usadas para calcular a média de classificações do condutor. O pedido estará também associado a um local de entrega, caracterizado pela cidade, nome da rua, número da rua, código postal.

O Restaurante tem como elementos identificativos o nome, o tipo, a localização, da qual interessa saber o mesmo que o local de entrega, e a classificação, é a média de classificações de clientes. Disponibiliza pratos, cada um com um nome e preço, que constituem os itens pedidos, juntamente com a quantidade de cada prato. Estes itens fazem parte do pedido e estão presentes na fatura, também associada ao pedido composta por um número identificativo, os itens pedidos e a sua quantidade, o preço total e a data.

Modelo Relacional:

Person (NIF, name, birth\_date, email, phone, password)

Person\_FD = {NIF -> name, birth\_date, email, phone, password}

Customer (customerNIF -> Person) - Não existe violações, cada Customer está associado a uma pessoa.

Driver (driverNIF -> Person, ss\_number, start\_date, team\_leader, /rating\_average) - Não existe violações, cada Driver está associado a uma pessoa.

Vehicle (license\_plate, make, model) - Não existe violações, cada Vehicle está associado a um Driver.

VehicleDriver (driverNIF -> Driver, vehicle\_license\_plate -> Vehicle, begin, end) - Class associação.

Team (driverNIF -> Driver, leaderNIF -> Driver)

CreditCard (number, cvv, exp\_date, card\_type, customerNIF -> Customer) - Não existe violações, cada CreditCard está associado a um Customer.

Demand (demandID, date, price, especification, delivery\_fee, customerNIF -> Customer, driverNIF -> Driver, locationID -> Location, paymentTypeID -> PaymentType, creditCardID -> CreditCard) - Não existe violações, cada Demand está associada a um Customer, Location, PaymentType e uma Food.

PaymentType (paymentTypeID, type) - Não existe dependências.

Review (reviewID, rating, text, demandID -> Demand) - Não existe violações, cada Review está associada a um Demand.

Rating (ratingID, /rating) - Class associação.

Food (foodID, name, price, restaurantID -> Restaurant) - Não existe violações, cada Food está associada a um Restaurant.

Demanded (foodID -> Food, demandID -> Demand)

Restaurant (restaurantID, name, NIF, locationID ->Location, /rating\_average, restaurantTypeID -> RestaurantType) - Não existe violações, cada Restaurant está associado a um RestaurantType e uma Location.

RestaurantType (restaurantTypeID, type) - Não existe dependências.

Location (locationID, city, street\_name, street\_number, postal\_code) - Não existe dependências.

Invoice (id, /total, date, demandID→Demand) - Não existe violações, cada Invoice está associada a um Demand.

InvoiceLine (invoice\_lineID, quantity, foodID -> Food, invoiceID -> Invoice) - Não existe violações, cada InvoiceLine está associada a uma Food e a uma Invoice.