Base de Dados

Carpooling App

14 de Abril de 2019

2MIEIC01 – Grupo 104

Francisco Batista	up201604320@fe.up.pt
João Rocha	up201708566@fe.up.pt
Tiago Alves	up201603820@fe.up.pt

Index

Context

In this project we pretend to create a database for an application that manages rides so different users can share them. The database should be capable of storing and manage all the necessary information for the complete operation.

Our idea is to create a platform where a user with a car creates a trip with some specifications, like starting/ending time and initial/final address, and other users can join the trip, with the intention of splitting the final costs. The path chosen will be calculated by a map provider (google maps etc.) and all the users with cars will have to register its model so the trip price can be calculated using its average consumption and the trip distance.

Specifications

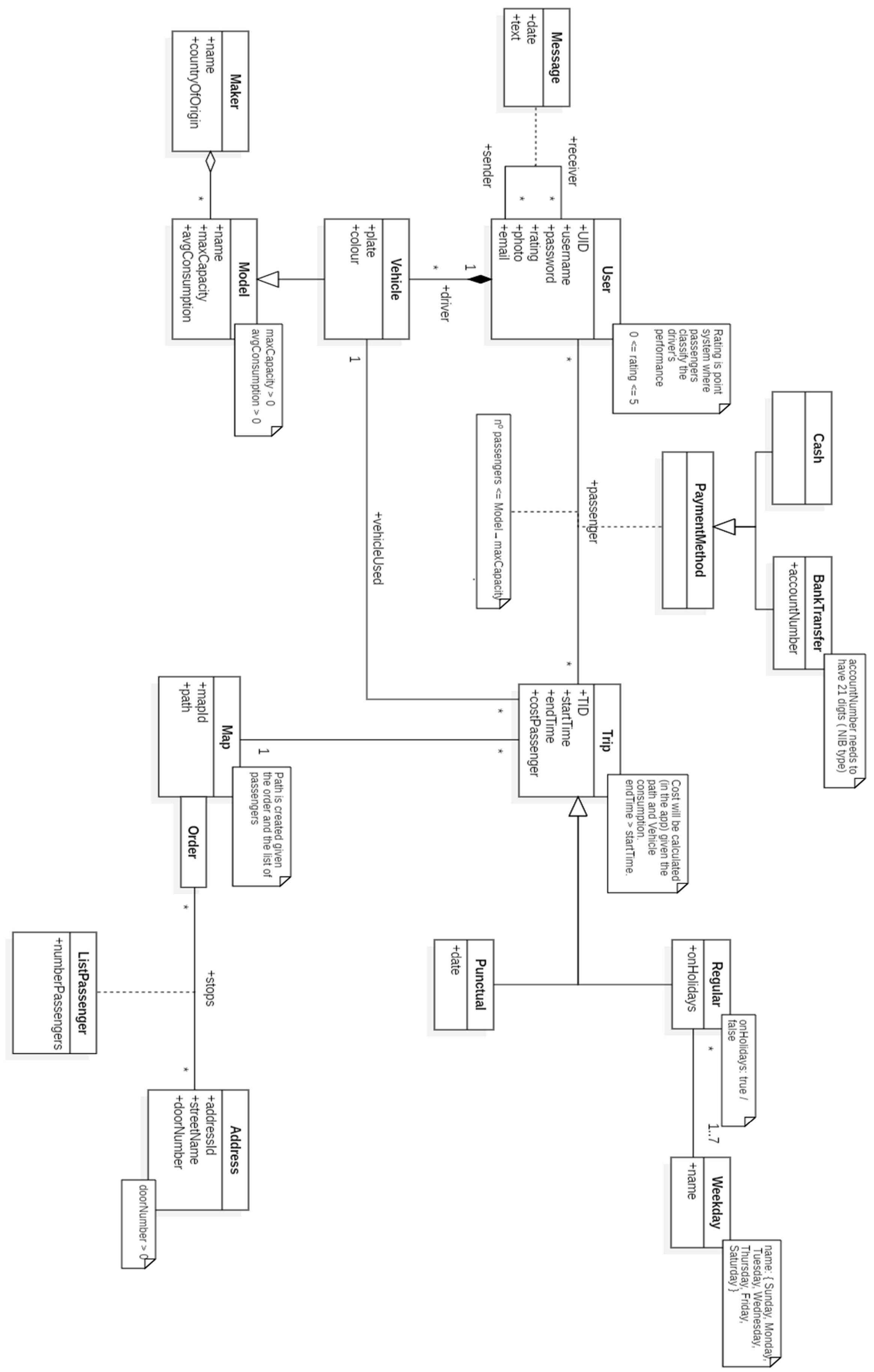
Any person using the *app* will be a **User**. The users will have a unique UID, username, password, rating, photo and email address. Users will be able to exchange many messages between themselves. They will be stored as **Message** with a date and text.

One of the Users will have a **Vehicle** which will be identified by their plate and color. Each Vehicle belongs to a **Model** that is composed by name, maximum capacity, average consumption and that belongs to a **Maker** with its name and country of origin. One user can own and drive as many vehicles as they wish.

The app also has a map provider that creates **Maps,** which creates a Path automatically using the **Order** of the **Addresses** associated (Like google maps). These Addresses, with streetName and doorNumber can be rearranged in different orders, creating different maps and paths.

The vehicle, users and the map are all connected to the **Trip** which will have an ID (TID), startTime, endTime, and cost per passenger (costPassenger) that will be calculated based on the path and number of passengers in each part of the trip. Each trip can only have a Map, a Vehicle, where the driver associated will be its driver, and it can have passengers as long as they don't surpass the Vehicle's maximum capacity. There will also be a **Payment Method** that will consist of either **Cash** or **Bank Transfer**. An account Number is needed if the User decides to opt for the Bank Transfer.

Finally, the Trip can be subdivided in two: The **Punctual** and **Regular** trip. The Regular trips are associated with a **Weekday**. The difference between them is that the Punctual must be associated with just one date and the regular will be associated with at least one day of the week. The regular trip also has an attribute called is onHolidays, which informs if there will be a "break" of the regular trip.



Message
+date
+text
Maker
+name
+countryOfOrigin
User
+UID
+username
+password
+rating
+photo
+email
+receiver
+sender
Model
+name
+maxCapacity
+avgConsumption
maxCapacity > 0
avgConsumption > 0
Vehicle
+plate
+colour
+driver
Rating is point system where passengers classify the driver's performance
0 <= rating <= 5
Cash
PaymentMethod
BankTransfer
+accountNumber
accountNumber needs to have 21 digts (NIB type)
nº passengers <= Model – maxCapacity
+passenger
+vehicleUsed
Trip
+TID
+startTime
+endTime
+costPassenger
Cost will be calculated (in the app) given the path and Vehicle consumption.
endTime > startTime.
Map
+mapId
+path
Path is created given the order and the list of passengers
Order
Punctual
+date
Regular
+onHolidays
onHolidays: true / false
Weekday
+name
name: { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday }
ListPassenger
+numberPassengers
+stops
Address
+addressId
+streetName
+doorNumber
doorNumber > 0
1
*
1..7

Relational Model

User (<u>UID</u>, username, password, rating, photo, email)

- UID → username, password, rating, photo, email
- UID is the primary key
- rating is one *derived attribute // de certeza? Porque?*

Atributos que são calculados com base noutros são considerados devirados.. não é?

Message (<u>idMessage</u>, date, text, sender →User, receiver →User)

- idMessage → date, text, sender, receiver
- idMessage is the primary key
- sender and receiver are *foreign keys*

Vehicle (<u>plate</u>, name →Model, color)

- plate is the primary key.
- plate → name, color.
- name is a *foreign key*

Driver (<u>UID</u> →User, <u>plate</u> → Vehicle)

- UID and plate are the *composite primary key*
- UID and plate are *foreign keys*

Model (<u>name</u>, maxCapacity, avgConsumption, idMaker →Maker)

- name is the *primary key*
- name → maxCapacity, avgConsumption, idMaker.
- idMaker is a *foreign keys*

Maker (<u>idMaker</u>, name, countryOfOrigin)

- idMaker is the *primary key*.
- idMaker → name, countryOfOrigin.

Map (mapId, path, order)

- mapId is the *primary key*
- path is a *derived attribute*
- order is a *foreign key*
- mapId → path, order

Address (addressId, streetName, doorNumber)

- addresId is the *primary key*
- addressId → streetName, doorNumber

Trip (TID, startTime, endTime, costPassenger, map → Map, plate → Vehicle)

- TID is the *primary key*
- TID→ startTime, endTime, costPassenger, idMapProvider, plate
- map and plate are *foreign keys*
- costPassenger is a *derived attribute*.

ListPassenger(address → Address , order → Order, numberPassengers)

- address and order are the *composite primary key*
- address, order → numberPassengers
- address and order are *foreign keys*

Order(orderId)

- order is the primary key.

Cash (pid→ PaymentMethod)

- pid is the *primary key*
- pid is a *foreign key*

BankTransfer (pid →PaymentMethod, accountNumber)

- pid → accountNumber
- pid is the *primary key*
- pid is a *foreign key*

PaymentMethod (user → User, trip → Trip, pid)

- user, trip → pid
- user and trip are the *composite primary key*
- user and trip a *foreign key*

Regular (TID →Trip, onHolidays)

- TID → onHolidays
- TID is the *primary key*
- TID is a *foreign key*

Punctual (TID →Trip, date)

- TID → date
- TID is the *primary key*
- TID is a *foreign key*

Weekday (name)

- Name is the *primary key*

Functional Dependencies and Normal Form Analysis

In all the previous examples, it is possible to see that in each relation, the left side of the functional dependencies is a key for that relation. Therefore, the relational model is in the **Boyce-Codd Normal Form** and consequently, in the **3rd Normal Form**.

In the following paragraphs it is shown that the closure of the attributes in the left side is all the attributes in that relation:

User: $\{UID\}^+$ = {UID, username, password, rating, photo, e-mail}

Message: $\{idMessage\}^+$ = {idMessage, date, text, sender, receiver}

Vehicle: $\{plate\}^+$ = {plate, idModel, color}

Driver: $\{UID, plate\}^+$ = { UID, plate}

Maker: $\{idMaker\}^+$ = {idMaker, name, countryOfOrigin}

MapProvider: $\{idMapProvider\}^+$ = {idMapProvider, path}

Address: $\{idAddress\}^+$ = {idAddress, streetName, doorNumber}

Stops: $\{idAddress, idMapProvider\}^+$ = { idAddress, idMapProvider, order, numberPassengers}

Trip: $\{TID\}^+$ = {TID, startTime, endTime, costPassenger, idMapProvider, plate}

Passenger: $\{UID, TID\}^+$ = {UID, TID, paymentMethod}

Cash: $\{idPaymentMethod\}^+$ = {idPaymentMethod}

BankTransfer: $\{idPaymentMethod\}^+$ = {idPaymentMethod}

Regular: $\{TID\}^+$ = {TID, isOnHolidays}

Punctual: $\{TID\}^+$ = {TID, date}

Weekday: $\{dayNumber\}^+$ = {dayNumber}

RegularWeekDay: $\{TID, dayNumber\}^+$ = {TID, dayNumber}

Restrictions

	Restriction	Implementation
User	There cannot exist two users with the same UID	UID PRIMARY KEY
	There cannot exist two users with the same username and all users must have a username	username UNIQUE NOT NULL
	The rating must be between 1 and 5. Default value is 0	rating CHECK (rating >= 1 AND rating <= 5) DEFAULT (0.00)
	Users must have a password associated	password NOT NULL
	Users must have an email associated, the users cannot share emails.	Email UNIQUE NOT NULL
Message	There cannot exist two messages with the same messageId	messageID PRIMARY KEY
	The sender and receiver are foreign keys.	sender REFERENCES User(uID), receiver REFERENCES User(uID)
	Date and text can’t be NULL.	date NOT NULL, text NOT NULL
Vehicle	There cannot exist two vehicles with the same plate, vehicles have a unique plate.	plate PRIMARY KEY
	driver is a foreign key	driver REFERENCES Model(idModel)

	Restriction	**Implementation**
Model	There cannot exist two models with the same name	name PRIMARY KEY
	maxCapacity must be greater than one	maxCapacity CHECK (maxCapacity >= 1) NOT NULL
	avgConsumption default value is 6.0. It cannot be a negative value	avgConsumption CHECK (avgConsumption >= 0) DEFAULT (6.00)
	maker is a foreign key. It cannot be NULL	maker REFERENCES Maker(name) NOT NULL
Maker	There cannot exist two makers with the same name	Name PRIMARY KEY
	countryOfOrigin cannot be null	countryOfOrigin NOT NULL
Map	There cannot exist two maps with the same mapId	mapId PRIMARY KEY
	All map must have a path	path NOT NULL
	All maps must have an order	order NOT NULL
Address	There cannot exist two addresses with the same addressID	addressId PRIMARY KEY
	All addresses must have a streetName	streetName NOT NULL
	Restriction	**Implementation**

Address	All addresses must have a doorNumber and must be greater than zero	doorNumber NOT NULL CHECK (doorNumber > 0)
Trip	There cannot exist two trips with the same TID	TID PRIMARY KEY
	mapId and plate are foreign keys	mapId REFERENCES Map(mapId), vehicle REFERENCES Vehicle(plate)
	endTime must be bigger than startTime and neither of them can be null	endTime CHECK (endTime > startTime) startTime NOT NULL endTime NOT NULL
	All trips must have a costPassenger associated and it must be greater than zero	costPassenger CHECK (costPassenger >= 0) NOT NULL
Order	There cannot be two orders with the same orderId	orderId PRIMARY KEY
ListPassengers	addressId and orderId are the composite primary keys	orderId REFERENCES Order(orderId), addressId REFERENCES Address(addressId), PRIMARY KEY (orderId , addressId)
	numberPassengers cannot be NULL	numberPassenger NOT NULL
PaymentMethod	User and trip are the composite Primary Key	user REFERENCES User(UID), trip REFERENCES Trip(TID), PRIMARY KEY (user, trip)
Cash	Pid is a foreign key, it is the Primary Key	Pid REFERENCES PaymentMethod(PID) PRIMARY KEY NOT NULL
	Restriction	**Implementation**

BankTransfer	Pid is a foreign key, it is the Primary Key	Pid REFERENCES PaymentMethod(PID) PRIMARY KEY NOT NULL
	account number must be 21 digits long. All BankTransfers must have an accountNumber associated	accountNumber CHECK (accountNumber >= 100000000000000000000 and accountNumber < 999999999999999999999) accountNumber NOT NULL
Regular	There cannot exist two regular trips with the same TID	TID PRIMARY KEY
	TID is a foreign key	TID REFERENCES Trip(TID)
	oHolidays has the default value of true	isOnHolidays DEFAULT(1)
Punctual	There cannot exist two punctual trips with the same TID	TID PRIMARY KEY
	TID is a foreign key	TID REFERENCES Trip(TID)
	date cannot be null	date NOT NULL
Weekday	There cannot exist two weekdays with the same day	day PRIMARY KEY
	day can only be one valid day of the week	(CHECK name=='Monday' or name=='Tuesday' or name=='Wednesday' or name=='Thursday' or name=='Friday' or name=='Saturday' or name=='Sunday')