# Projeto de Base de Dados

– MyTransport

José Carlos Mello - 87456
Thiago Brasil - 96149

## Contexto

- Criação de uma base de dados para uma plataforma de aluguel de veículos para entregas ou transportes curtos; Facilidade de pesquisa e ampla disponibilidade de veículos.

## Problemas

- Uma das maiores realidades dos estudantes e trabalhadores é a dificuldade de se locomover rapidamente entre sua casa e faculdade, assim como na obtenção de veículos para entregas e curtas locomoções, muitas vezes recorrendo a preços abusivos de outras plataformas ou tendo que se deslocar longas distâncias por longos períodos. Isso poderia ser simplificado com a criação de uma plataforma organizada e unificada com ampla disposição de veículos.

## Objetivos

- Uma plataforma unificada.
- Ampla disponibilidade de conteúdo.
- Organização de todo conteúdo.
- Facilidade de pesquisa por veículos disponíveis.
- Facilidade no rápido conserto de veículos.

## Tarefas

- Criação de uma base de dados adequada à plataforma em caso.
- Implementação de um frontend que permita operações relevantes sobre a base de dados de acordo com as necessidades estabelecidas.

# Análise de Requisitos

## State of the art

- Como estabelecido nos objetivos da cadeira em questão, iremos utilizar o SGBD Microsoft SQL Server e a linguagem SQL para criação e manipulação da base de dados, bem como inserção de valores na mesma.
- Tendo em conta o requisito anterior, a linguagem de programação do frontend está limitada às connection libraries disponíveis para SQL Server (C#, C++, Java, Node.js, Python, entre outras).
- Independentemente da linguagem escolhida, várias tecnologias estão disponíveis para frontend, desde Windows Forms, a Qt, Tinkter, etc. sendo que a utilização de uma que não seja Windows Forms será sempre acompanhada de um relatório complementar que identifique as diferenças quanto à forma de interação.

## Requisitos Funcionais

### Atores

- Alunos: Estudantes de diversas Universidades.
- Pessoas que buscam uma maneira mais rápida de locomoção.
- Trabalhadores: Pessoas que também pretendam utilizar a plataforma para entrega de mercadorias e serviços.
- Admin: Os administradores da plataforma. Responsáveis por validar veículos, lidar com erros e inconsistências, e impedir a propagação de informação factualmente errada.
- Mecânicos: Pessoas que se disponibilizem para realizar consertos e a devida ajuda para manter os veículos funcionando.

# Requisitos Não-funcionais

### Performance

- Utilização paralela da aplicação por múltiplos utilizadores não deve afetar significativamente a performance da mesma,
- Múltiplos pedidos não devem demorar tempo excessivo.
- Pesquisas por veículos disponíveis também devem ser concluídas em tempo útil.
- A interface deve ser responsiva principalmente quando acopladas a operações com a base de dados.

### Portability and compatibility

- Devem ser suportados diferentes tipos de veículos.

### Reliability, availability and maintainability

- A aplicação deve permitir acessos de múltiplos utilizadores simultaneamente.
- Caso a aplicação seja desligada ou em caso de ocorrência de erro, os dados inseridos na base de dados devem ser preservados.

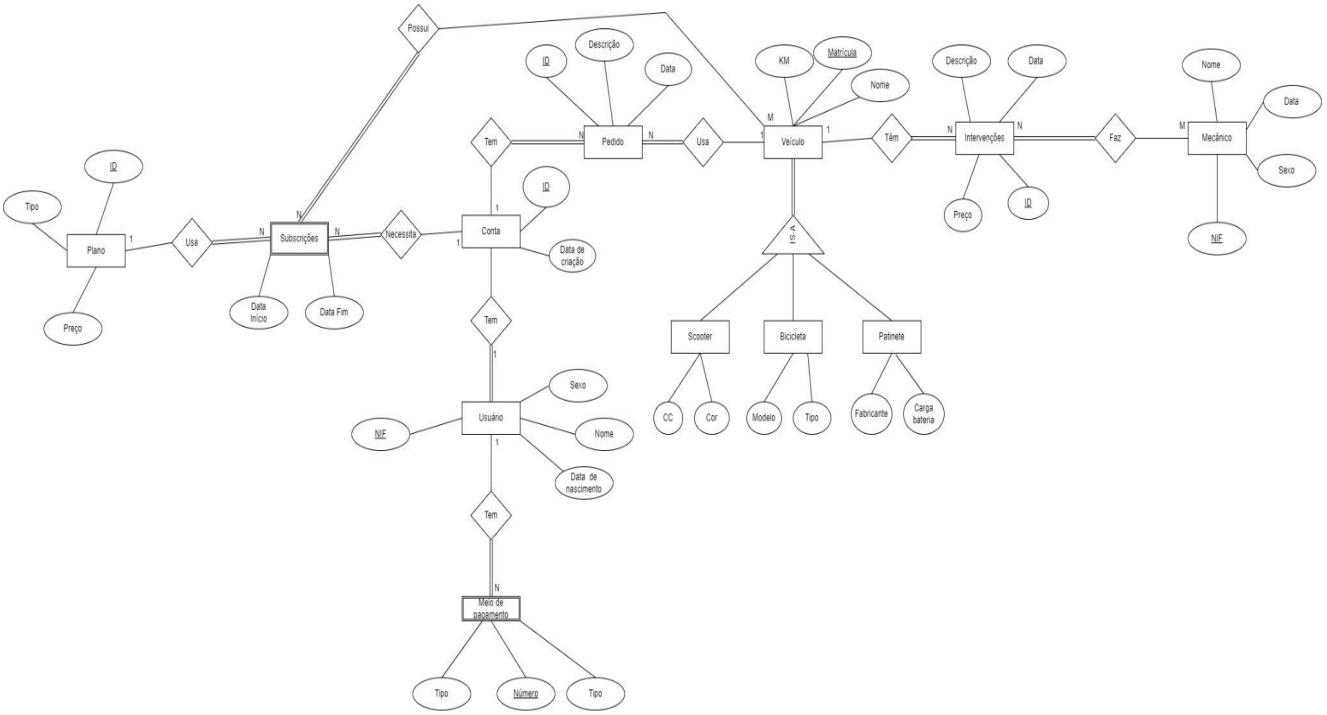
### Security

- Proteção de tipos de dados e integridade das tabelas da base de dados.
- Acessos à base de dados devem ser protegidos de acessos abusivos ou que não respeitem a integridade dos dados (ex, SQL injections).

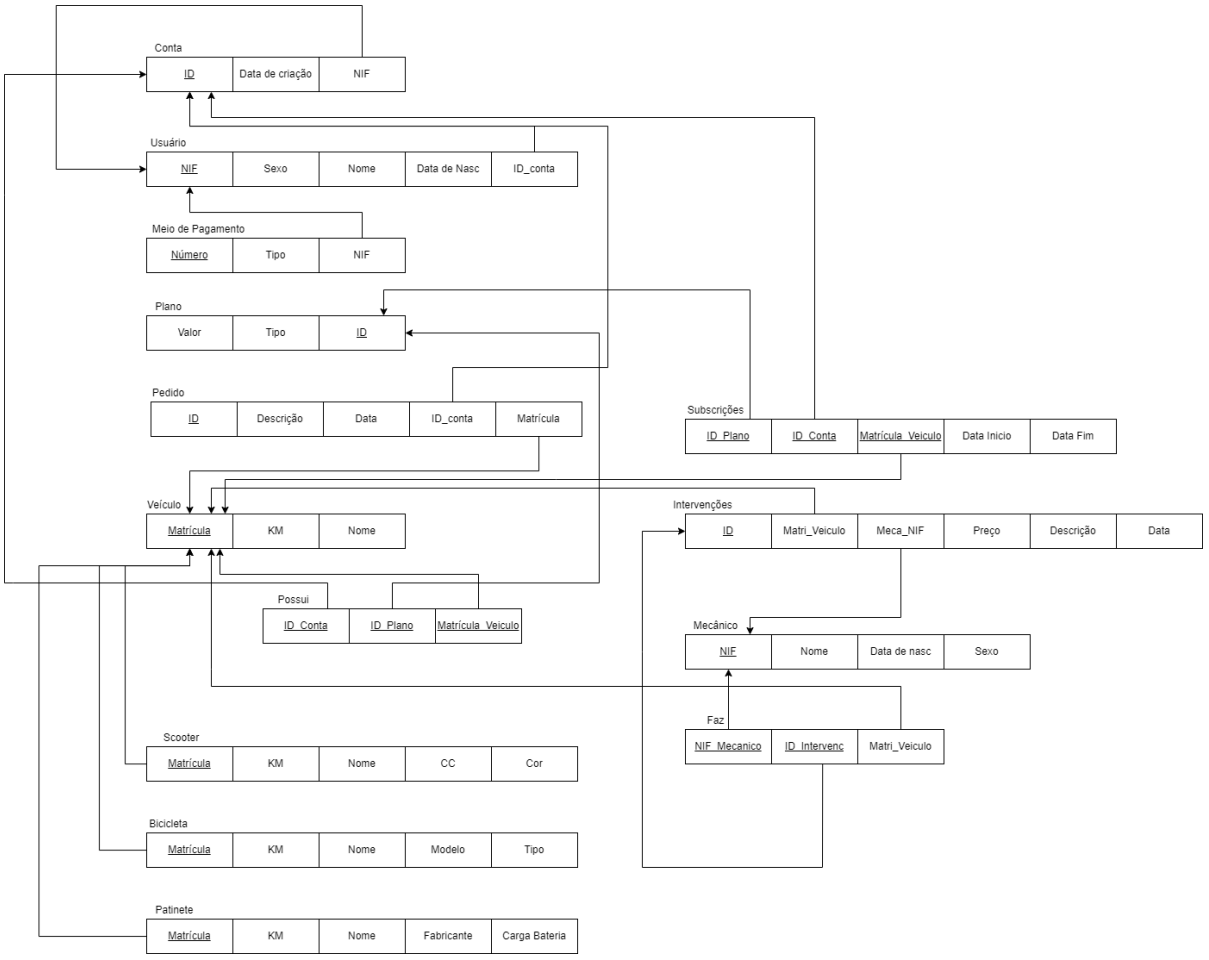
### Usability

- A interface da aplicação deve ser fácil de utilizar.
- Acessos feitos à base de dados devem ser tornados claros para quem estiver utilizando a aplicação.
- Opções de input devem fornecer feedback, inclusive em caso de erro.

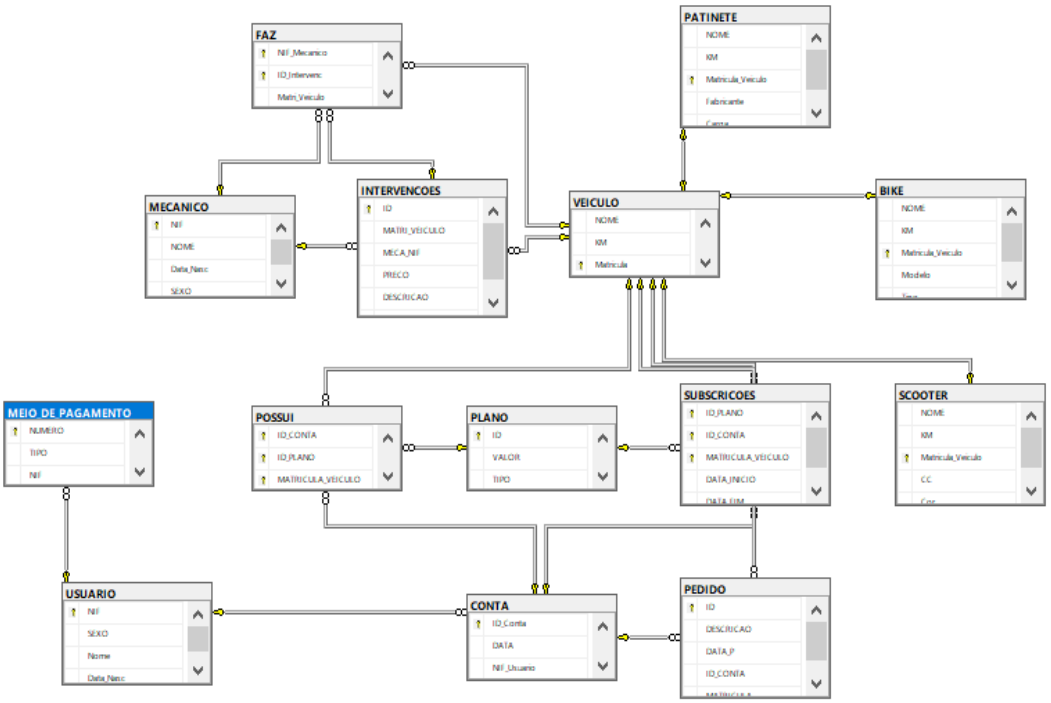
- **DER**


Possui
ID
Descrição
Data
KM
Matrícula
Nome
Descrição
Data
Nome
Data
Tem
Pedido
Usa
Veículo
Tém
Intervenções
Faz
Mecânico
Sexo
Tipo
ID
Plano
Usa
Subscrições
Necessita
Conta
ID
Data de criação
Preço
Data Início
Data Fim
Tem
Usuário
Sexo
NIF
Nome
Data de nascimento
Tem
Meio de pagamento
Tipo
Número
Tipo
Scooter
Bicicleta
Patinete
CC
Cor
Modelo
Tipo
Fabricante
Carga bateria
Preço
ID
NIF

- **Esquema Relacional**



- **Database Diagram**



## ● Normalização

Tendo em conta o objetivo da disciplina e do desenho de uma base de dados relacional e, tendo em vista, a minimizar a redundância dos dados, foram feitos diversos testes para verificar que formas normais estávamos a respeitar. Após estas validações, foi concluído que o modelo de dados se encontrava na Terceira Forma Normal (3FN), respeitando também as formas normais anteriores.Com isso, concluiu-se que as relações estavam todas normalizadas, não necessitando de futuras alterações.

## ● Indexes

Ao analisar os tipos de queries que seriam feitas na nossa aplicação, optamos por criar um índice tipo e valor, com isso seria mais rápida a procura por planos que tivessem um melhor custo benefício ao cliente.

## ● Triggers

Foram criados alguns triggers para melhorar o desempenho e agilidade da base de dados.
Exemplo TriggerCreateUser: Trigger criado para a criação de contas após a criação de um usuário.

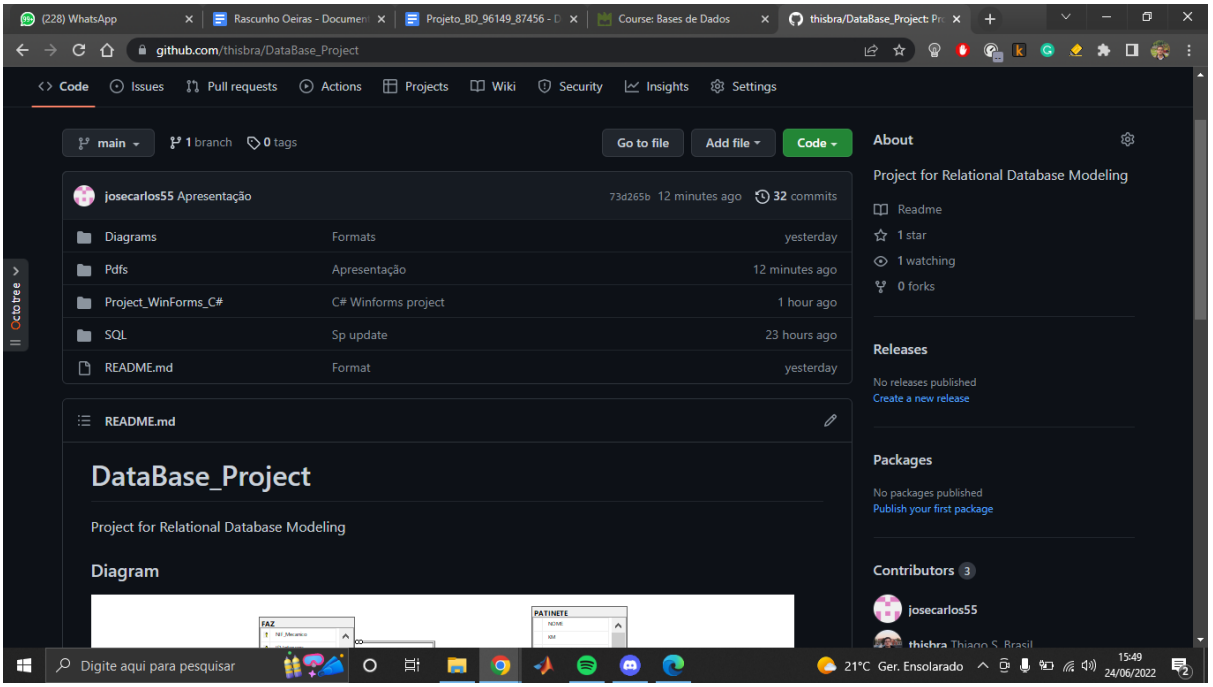
## ● Stored Procedures

Foram criados alguns stored procedures para melhor manutenção e desempenho da base de dados.
Exemplo sp_addInterven: Stored procedure utilizado para criar intervenções, atribuindo a ela um valor de ID e a data local de quando foi feita essa solicitação.

- **Git Management**

  O trabalho foi controlado, versionado e submetido pelo *github*:

(https://github.com/thisbra/DataBase_Project)



## ● Conclusão

Além das explicações  já referidas e não havendo nenhuma query que a álgebra relacional não resolvesse, optou-se também por não utilizar Cursores devido à sua falta de desempenho prático para o nosso trabalho.Com isso, podemos concluir que os objetivos do trabalho foram superados e que foi desenvolvido conhecimento na área de base de dados.