**Faculdade de Engenharia da Universidade do Porto**



**Parque de Diversões**

**Base de Dados 2018/2019**

**Mestrado Integrado em Engenharia Informática e Computação**

[**Carla Alexandra Teixeira Lopes**](https://sigarra.up.pt/feup/pt/func_geral.formview?p_codigo=467117)

**João Miguel Rocha da Silva**

Estudantes & Autores:

Alexandre Miguel de Araújo Carqueja up201705049 up201705049[@fe.up.pt](mailto:up201705580@fe.up.pt)

David Freitas Dinis up201706766 [up201706766@fe.up.pt](mailto:up201705580@fe.up.pt)

Margarida Alves Pinho up201704599 [up201704599@fe.up.pt](mailto:up201704599@fe.up.pt)

Índice

1. Introdução

1.1 Tema do trabalho

1. Diagrama UML
   1. Primeira Entrega
   2. Segunda Entrega
2. Associações UML
   1. Associação Época/Horário/Serviço
   2. Associação Cliente/Serviço
   3. Associação Funcionário/Serviço
3. Classes UML
   1. Classe Serviço
4. Esquema Relacional
5. Dependências Funcionais
6. Análise de Dependências Funcionais e Formas Normais
7. Lista e forma de implementação das restrições

**Introdução**

**Tema do Projeto**

O nosso projeto desenvolve-se à volta das necessidades que a gestão de um parque de diversões tem para o seu armazenamento de dados. No nosso modelo guardamos informações relativas aos Clientes e Funcionários, criando as associações necessárias com os restantes elementos do parque.

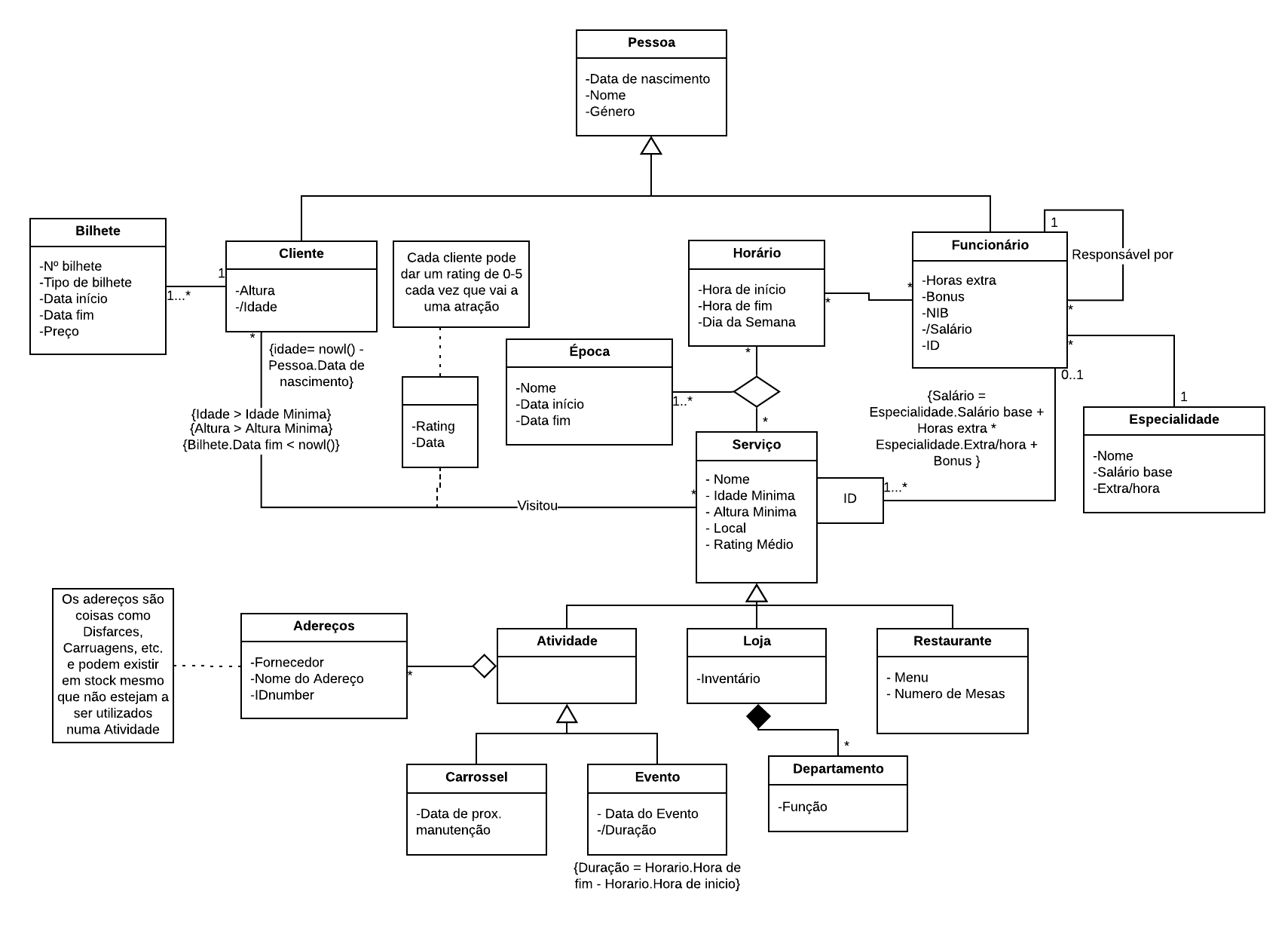
Para os clientes guardamos informações relativas aos seus bilhetes, bem como um registo de todos os serviços de que usufruíram, e ainda o grau de satisfação em cada um deles. Cada vez que os clientes visitam o parque só podem adquirir um bilhete no qual estará especificado o seu tipo e preço, bem como os dias da visita.

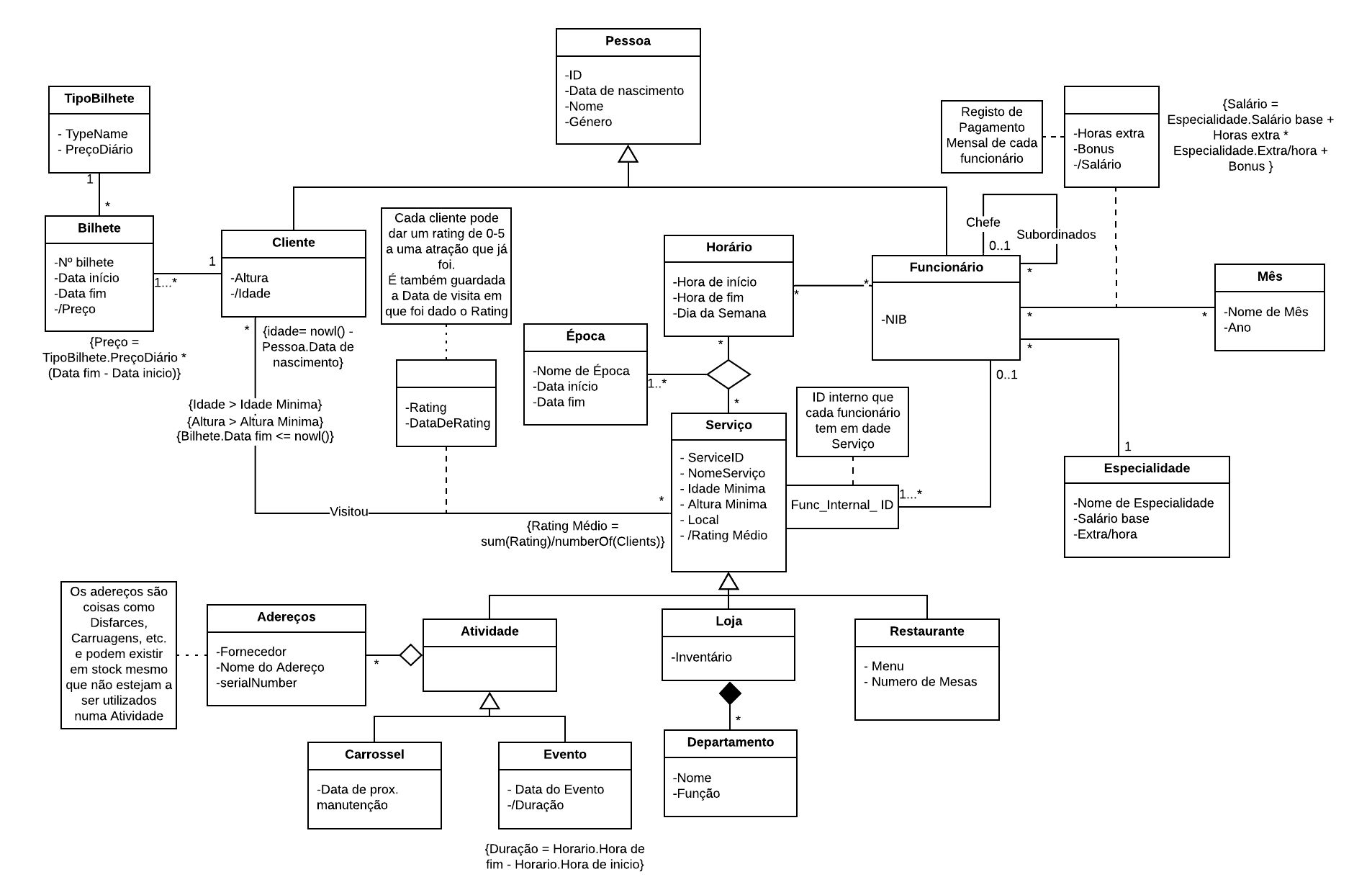
Para os funcionários temos acesso aos seus horários e especialidade, o que nos permite também calcular e aceder ao seu salário. Para além disso, também guardamos os serviços em que trabalham dentro do parque, bem como a pessoa responsável por esse funcionário.

Finalmente, para todos os serviços do Parque (Lojas, Atividades, Restaurantes), conseguimos obter qual o seu horário de funcionamento, dependendo da época do ano, bem como impor condições sobre os clientes que os podem frequentar, segundo a sua idade, altura e data de validade do bilhete. Tanto os Carrosséis como os Eventos fazem parte de Atividades às quais pertencem adereços e sobre os quais guardamos o seu fornecedor, nome e número de identificação. Todas as lojas têm um inventário e são formadas por diversos departamentos. Todos os restaurantes têm um menu e número de mesas.

**Diagrama UML**

**2.1 Primeira entrega**



**2.2 Segunda entrega**

**Associações UML**

No texto que se segue irão ser esclarecidas algumas das associações que se estabeleceram.

**Associação Época/Horário/Serviço:**

Nesta associação ternária é pretendido estabelecer uma relação de dependência entre as 3 classes.

Para cada serviço, dependendo da época em que nos encontramos, existirá um horário de funcionamento (com a possibilidade de existência de vários horários, dependendo do dia da semana). Ao mesmo tempo, para cada horário, numa determinada época, podem existir vários serviços correspondentes. E finalmente, para cada serviço, cada horário que ele tem, estará em vigor numa ou várias épocas.

Garantimos assim uma flexibilidade de horários dependendo da época, bem como a possibilidade de pesquisar qualquer um destes atributos em função dos outros.

**Associação Cliente/Serviço:**

Com esta associação pretende-se estabelecer uma relação restrita entre as duas classes, em que se guardam todas as visitas que um cliente fez aos serviços do parque.

Assim, cada cliente poderá usufruir de um serviço, no entanto, nalguns só o poderá fazer se a sua idade e altura forem de acordo com os requisitos de cada serviço. A partir desta relação também se pretende saber a classificação (entre 0 e 5) dada a cada serviço, pelo cliente, bem como a data em que essa classificação foi dada, sendo assim necessário uma classe de associação.

**Associação Funcionário/Serviço:**

Nesta associação estabelece-se a relação de trabalho que cada funcionário pode ter nos vários serviços do Parque.

Adicionalmente, com o ID de um funcionário conseguimos determinar se ele trabalha ou não numa determinado serviço.

**Classes UML**

No texto que se segue será esclarecida uma das classes criadas.

**Classe Serviço**

A classe Serviço é uma generalização completa e exclusiva dos serviços que o parque de diversões pode oferecer.

Os serviços podem então ser Lojas, Restaurantes ou Atividades (sendo que as atividades podem ser carroceis ou eventos, como por exemplo desfiles). Assim sendo, todos os serviços têm restrições à entrada de clientes (altura mínima e idade mínima), bem como um nome e indicação do local dentro do parque.

Finalmente, todos os casos especiais de um Serviço vão ter funcionários atribuídos e horários específicos consoante a época.

**Esquema Relacional**

**Pessoas** (ID, DataNascimento, Nome, Género)

**Clientes** (ID->Pessoa, Altura, Idade)

**TipoDeBilhete** (TypeName, PreçoDiario)

**Bilhetes** (Nbilhete, typeName->TipoBilhete, DataInicio, DataFim, Preço, IDcliente->Cliente)

**Especialidades** (NomeEspecialidade, SalárioBase, Extra/hora)

**Funcionários** (ID->Pessoa, NIB, NomeEspecialidade->Especialidades)

**Mês** (IDmes, NomeDeMes, Ano)

**RegistoPagamentos** (IDmes->Mes, ID->Funcionário, HorasExtra, Bonus, Salário)

**Hierarquias** (Subordinado->Funcionário, Chefe->Funcionário)

**Horários** (IDhorario, HoraInicio, HoraFim, DiaSemana)

**HoráriosTrabalho** (IDhorario->Horário, ID->Funcionário)

**Épocas** (NomeDeEpoca, DataInicio, DataFim)

**Serviços** (ServiceID, NomeDeServiço, IdadeMinima, AlturaMinima, Local, RatingMedio)

**ServiçosÉpocasHorários** (ServiceID->Serviço, NomeDeEpoca->Época, IDhorario->Horário)

**FuncionárioDeServiço** (ServiceID->Serviço, IDfuncionario->Funcionários, Func\_Internal\_ID)

{Func\_Internal\_ID, ServiceID} **UK**

**Visitas** (IDcliente->Cliente, ServiceID->Serviço, DataDeRating, Rating)

**Lojas** (StoreServiceID->Serviço, Inventário)

**Restaurantes** (RestaurantServiceID->Serviço, Menu, Nmesas)

**Atividades** (ActivityServiceID->Serviço)

**Carrosseis** (CarrousselServiceID->Atividade, DataProxManutenção)

**Eventos** (EventServiceID->Atividade, DataDeEvento, Duração)

**Adereços** (SerialNumber, Fornecedor, NomeDeAdereço, ActivityServiceID->Atividade)

**Departamentos** (Nome, StoreServiceID->Loja, Funcao)

**Dependências Funcionais**

**Pessoas**

{ID} -> {DataNascimento, Nome, Género}

**Clientes**

{ID} -> {Altura, Idade}

**Funcionários**

{ID} -> {NIB, NomeEspecialidade}

**Bilhetes**

{Nbilhete} -> {typeName, DataInicio, DataFim, Preço, clientID}

{typeName, DataInicio, DataFim} -> Preço

**TipoDeBilhete**

{TypeName} -> {PreçoDiario}

**Especialidades**

{NomeEspecialidade} -> {SalárioBase, Extra/hora}

**Mês**

{IDmes} -> {NomeDeMes, Ano}

{NomeDeMes, Ano} UK

**Hierarquias**

{Subordinado} -> {Chefe}

**Horários**

{IDhorario} -> {HoraInicio, HoraFim, DiaSemana}

{HoraInicio, HoraFim, DiaSemana} UK

**HoráriosTrabalho**

{Horário, Funcionário}

**Épocas**

{NomeDeEpoca} -> {DataInicio, DataFim}

**Serviços**

{ServiceID} ->{NomeServiço, IdadeMinima, AlturaMinima, Local, RatingMedio}

**ServiçosÉpocasHorários**

{Serviço, Época, Horário}

**FuncionárioDeServiço**

{ServiceID, ID} -> {Func\_Internal\_ID}

{ServiceID, Func\_Internal\_ID} -> {ID}

{Func\_Internal\_ID, ServiceID} UK

**Visitas**

{Cliente, ServiceID} -> {DataDeRating, Rating}

**Lojas**

{StoreServiceID} -> {inventário}

**Restaurantes**

{RestaurantServiceID} -> {menu, Nmesas}

**Atividades**

{ActivityServiceID}

**Carrosseis**

{CarrousselServiceID} -> {DataProxManutenção}

**Eventos**

{EventServiceID} -> {DataDeEvento, Duração}

**Adereços**

{serialNumber} -> {fornecedor, NomeDeAdereço, Atividade

**Análise de Dependências Funcionais e Formas Normais**

Na maior parte das dependências funcionais do nosso esquema relacional existe uma única chave e uma única dependência (por exemplo, Pessoa: {ID} -> {DataNascimento, Nome, Género}). Estes casos encontram-se na forma normal BNCF e como a 3º forma normal é um superset de BNCF podemos afirmar que também se encontram na 3º forma normal.

No entanto existem algumas exceções a esta regra:

* Bilhetes: {Nbilhete} -> {TypeName, DataInicio, DataFim, Preço, clientID}

{TypeName, DataInicio, DataFim} -> Preço

Esta relação não se encontra na forma BNCF, pois {TipoBilhete, DataInicio, DataFim} não é uma superkey, adicionalmente, também não se encontra na 3º forma normal, pois o Preço não faz parte de nenhuma chave da relação.

Isto é justificado, pois o Preço é um elemento derivado de TypeName.getPrice() \* (DataFim - DataInicio).

* FuncionárioDeServiço: {ServiceID, ID} -> {Func\_Internal\_ID}

{ServiceID, Func\_Internal\_ID} -> {ID}

{Func\_Internal\_ID, ServiceID} UK

Esta relação não se encontra na BNCF, pois a segunda FD não tem uma superkey. No entanto, esta relação encontra-se na 3º Forma normal, pois {ID} faz parte da chave {ServiceID, ID}.

**Lista e forma de implementação das restrições**

No texto que se segue serão listadas as restrições relativas a cada relação seguidas da sua forma de implementação.

**Pessoas**(ID, DataNascimento, Nome, Género)

Não pode haver duas pessoas com o mesmo ID (restrição UNIQUE) e para além disso o ID é chave primária (restrição PRIMARY KEY e NOT NULL).

**Clientes**(ID->Pessoa, Altura, Idade)

ID é chave primária (restrição PRIMARY KEY e NOT NULL) e também é chave estrangeira para Pessoa (restrição chave estrangeira).

**TipoDeBilhete**(TypeName, PreçoDiário)

TypeName é chave primária (restrição PRIMARY KEY e NOT NULL).

**Bilhetes** (ClienteID ->Cliente, Nbilhete, TypeName->TipoBilhete, DataInicio, DataFim, Preço)

Nbilhete é chave primária (restrição PRIMARY KEY e NOT NULL);

ClienteID é chave estrangeira para Cliente (restrição chave estrangeira);

TypeName é chave estrangeira para TipoDeBilhete (restrição chave estrangeira).

**Especialidades** (NomeEspecialidade, SalárioBase, Extra/hora)

NomeEspecialidade é chave primária (restrição PRIMARY KEY e NOT NULL).

**Funcionários** (ID->Pessoa, NIB, NomeEspecialidade->Especialidades)

ID é chave primária (restrição PRIMARY KEY e NOT NULL) e é também chave estrangeira para Pessoa (restrição chave estrangeira);

NomeEspecialidade é chave estrangeira (restrição chave estrangeira).

**Mês** (IDmes, NomeDeMes, Ano)

IDmes é chave primária (restrição PRIMARY KEY e NOT NULL).

**RegistoPagamentos** (IDmes->Mes, ID->Funcionário, HorasExtra, Bonus, Salário)

IDmes e ID são o composto da chave primária (restrição PRIMARY KEY e NOT NULL), e ambas são chaves estrangeiras para Mês e Funcionários respetivamente (restrição chave estrangeira).

**Hierarquias** (Subordinado->Funcionário, Chefe->Funcionário)

Subordinado é chave primária (restrição PRIMARY KEY e NOT NULL) e estrangeira para Funcionários (restrição chave estrangeira);

Chefe é chave estrangeira para Funcionários (restrição chave estrangeira).

**Horários** (IDhorario, HoraInicio, HoraFim, DiaSemana)

IDhorario é chave primária (restrição PRIMARY KEY e NOT NULL).

**HoráriosTrabalho**(IDhorario->Horário, ID->Funcionário)

IDhorario e ID são o composto da chave primária (restrição PRIMARY KEY e NOT NULL), e ambas são chaves estrangeiras para Horário e Funcionários respetivamente (restrição chave estrangeira).

**Épocas** (NomeDeEpoca, DataInicio, DataFim)

NomeDeEpoca é chave primária (restrição PRIMARY KEY e NOT NULL).

**Serviços** (ServiceID, NomeDeServiço, IdadeMinima, AlturaMinima, Local, RatingMedio)

ServiceID é chave primária (restrição PRIMARY KEY e NOT NULL).

**ServiçosÉpocasHorários**(ServiceID->Serviço, NomeDeEpoca->Época, IDhorario->Horário)

ServiceID, NomeDeEpoca e IDhorario são o composto da chave primária (restrição PRIMARY KEY e NOT NULL) e são também chaves estrangeiras, para Serviços Épocas e Horários respetivamente (restrição chave estrangeira).

**FuncionárioDeServiço**(ServiceID->Serviço, IDfuncionario->Funcionários, Func\_Internal\_ID)

{Func\_Internal\_ID, ServiceID} **UK**

IDfuncionario e ServiceID são o composto da chave primária e são também chaves estrangeiras para Serviço e Funcionários respetivamente (restrição chave estrangeira).

**Visitas** (IDcliente->Cliente, ServiceID->Serviço, DataDeRating, Rating)

ID e ServiceID são o composto da chave primária (restrição PRIMARY KEY e NOT NULL) e são também chaves estrangeiras para Clientes e Serviços respetivamente (restrição chave estrangeira).

**Lojas** (StoreServiceID->Serviço, Inventário)

StoreServiceID é chave primária (restrição PRIMARY KEY e NOT NULL) e estrangeira para Serviços (restrição chave estrangeira).

**Restaurantes** (RestaurantServiceID->Serviço, Menu, Nmesas)

RestaurantServiceID é chave primária (restrição PRIMARY KEY e NOT NULL) e estrangeira para Serviços (restrição chave estrangeira).

**Atividades** (ActivityServiceID->Serviço)

ActivityServiceID é chave primária (restrição PRIMARY KEY e NOT NULL) e estrangeira para Serviços (restrição chave estrangeira).

**Carrosseis** (CarrousselServiceID->Atividade, DataProxManutenção)

CarrousselServiceID é chave primária (restrição PRIMARY KEY e NOT NULL) e estrangeira para Atividades (restrição chave estrangeira).

**Eventos** (EventServiceID->Atividade, DataDeEvento, Duração)

EventServiceID é chave primária (restrição PRIMARY KEY e NOT NULL) e estrangeira para Atividades (restrição chave estrangeira).

**Adereços** (SerialNumber, Fornecedor, NomeDeAdereço, ActivityServiceID->Atividade)

SerialNumber é chave primária (restrição PRIMARY KEY e NOT NULL);

ACtivityServiceID é chave estrangeira para Atividade (restrição chave estrangeira).

**Departamentos** (Nome, StoreServiceID->Loja, Função)

Nome é chave primária (restrição PRIMARY KEY e NOT NULL);

StoreServiceID é chave estrangeira para Loja (restrição chave estrangeira).