Uma imagem com texto, símbolo

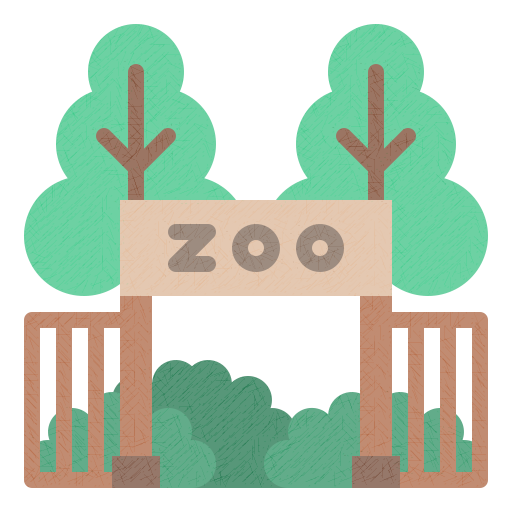
Descrição gerada automaticamente

**Base de Dados**

**Jardim Zoológico**

***L.EIC · BDAD · Grupo 207 · 2021/2022***

***Docente Lázaro Costa***



Daniela Tomás, up202004946

Nuno Penafort, up202008405

Sofia Sousa, up202005932

**Índice**

Contexto 3

Diagrama UML 4

Diagrama UML(revisto) 5

Esquema Relacional 6

Análise Dependências Funcionais e Formas Normais 7

Restrições 9

Autoavaliação 12

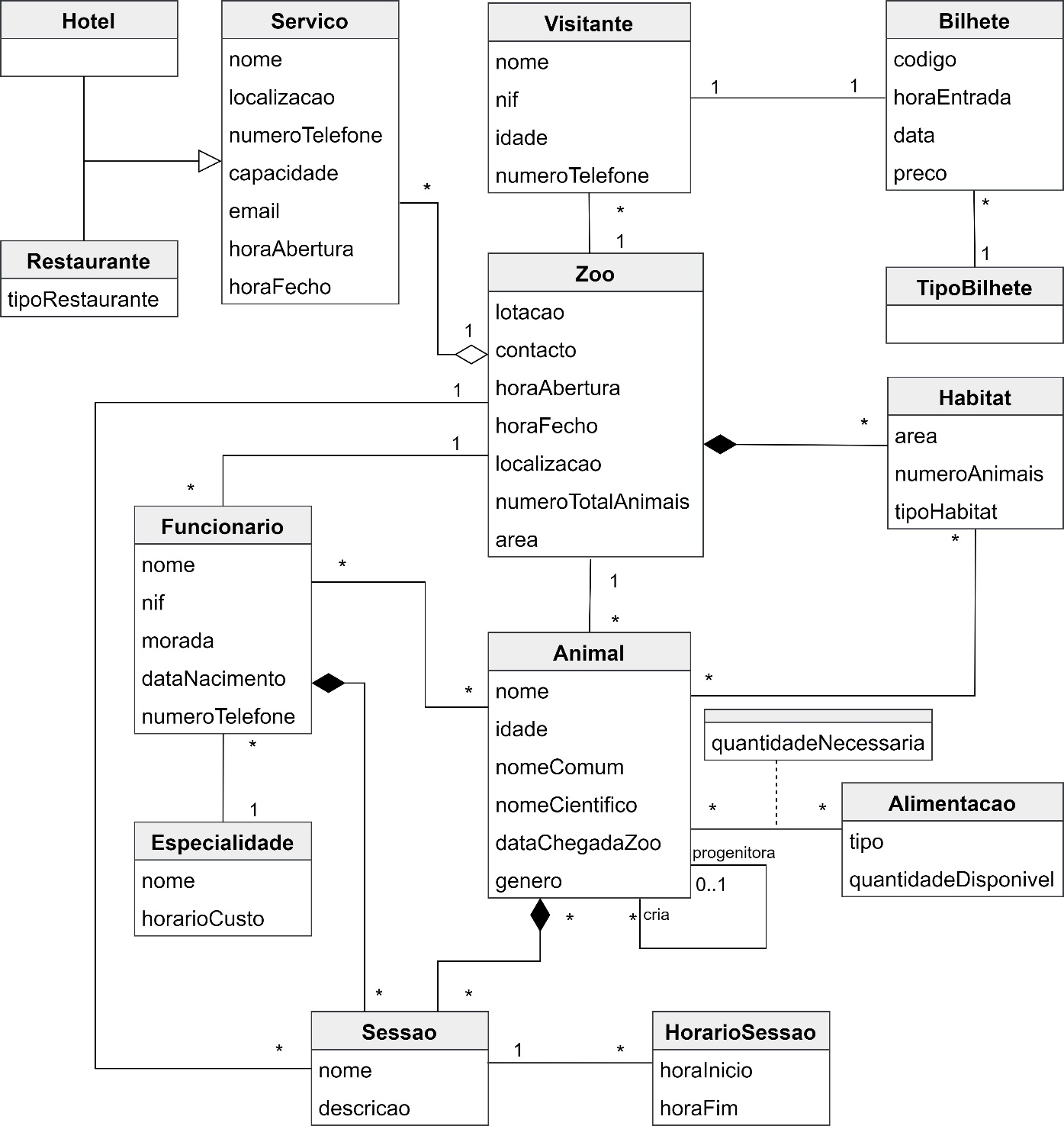
**Contexto**

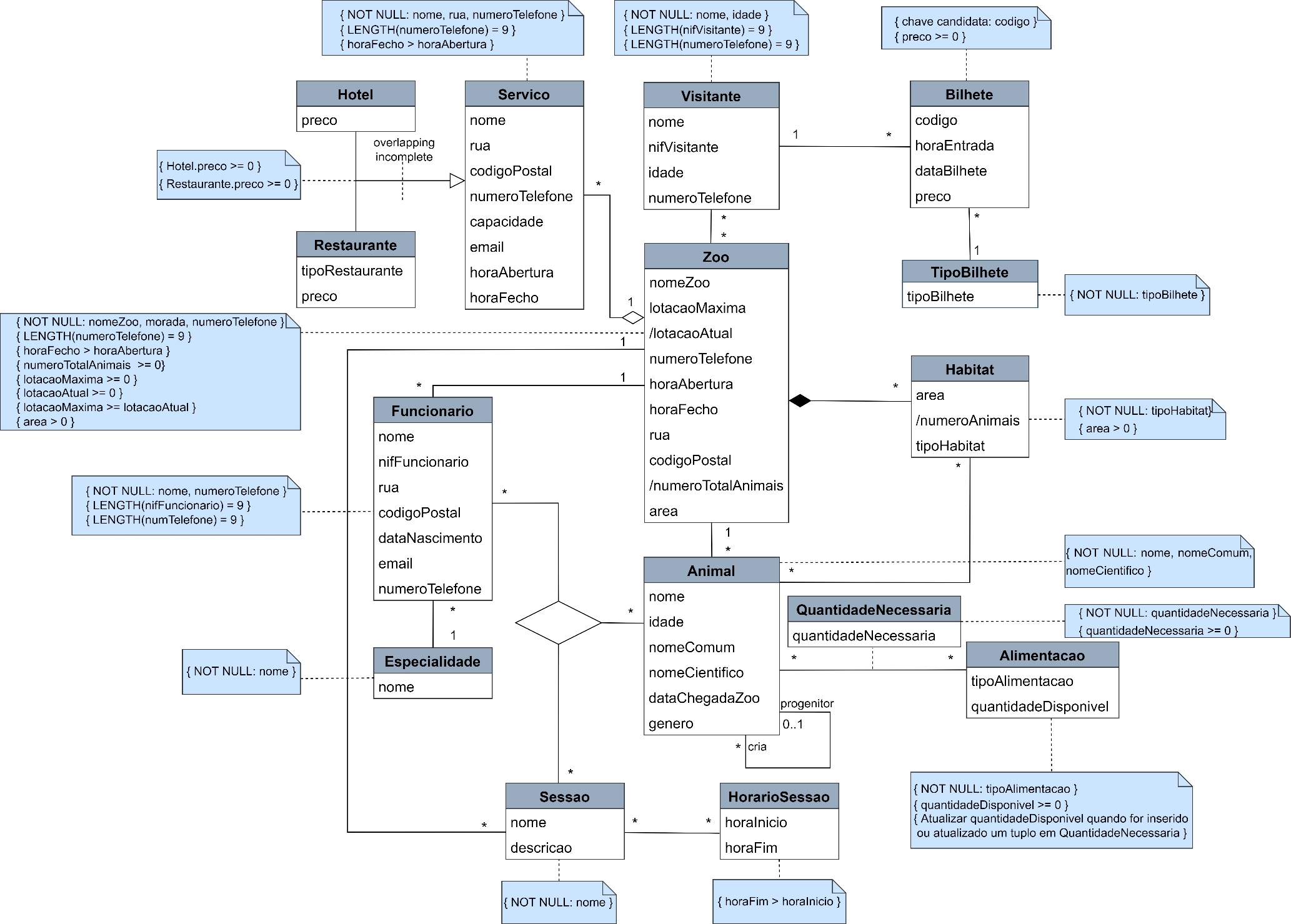
Um Jardim Zoológico pretende armazenar informação relativa ao seu funcionamento. O Zoo tem um nome, uma rua, um código postal, o número de telefone, uma área, hora de abertura e de fecho, o número total de animais e uma lotação atual e máxima que, de acordo com esta, o Zoo pode receber inúmeros visitantes. Sobre os visitantes é necessário conhecer o nome, o NIF, o número de telefone e a idade. Cada visitante tem apenas um bilhete, cujo preço varia consoante a idade e tem uma hora de entrada, a data e um código único. Além disso, o Zoo tem disponíveis sessões que são protagonizadas por animais e apresentadas por um tratador e têm um nome, uma descrição e um horário com hora de início e de fim. Os tratadores também são responsáveis por tratar dos animais fora das sessões.

Existem vários tipos de funcionários para além dos tratadores (manutenção, receção, etc…) e é necessário conhecer o nome, NIF, rua, código postal, data de nascimento, número de telefone e email.

Quanto aos animais é importante saber o nome, o nome comum, o nome científico, o género, a idade, data de chegada ao Zoo e a alimentação, onde é preciso ter em conta a quantidade de alimentos necessários, a quantidade de alimentos disponíveis e o tipo de alimentação. Os animais vivem em diferentes tipos de habitats com uma determinada área e com um determinado número de animais e podem ou não ter uma progenitora que pode ter várias crias.

O Zoo tem alguns serviços associados como, por exemplo, hotéis e restaurantes. Ambos possuem um nome, hora de abertura e de fecho, rua, código postal, número de telefone, capacidade e email. Os hotéistêm um preço base e os restaurantes servem um tipo característico de comida e têm só um preço com tudo incluído.

**Diagrama UML**

**Diagrama UML (revisto)**

**Esquema Relacional**

* **Zoo** ( idZoo, lotaçãoMaxima, lotacaoAtual, numeroTelefone, horaAbertura, horaFecho, rua, codigoPostal, numeroTotalAnimais, area )
* **Visitante** ( nifVisitante, nome, idade, numeroTelefone )
* **Bilhete** ( codigo, horaEntrada, dataBilhete, preco, nifVisitante -> Visitante, idTipoBilhete -> TipoBilhete )
* **TipoBilhete** ( idTipoBilhete, tipoBilhete )
* **Servico** ( idServico, nome, numeroTelefone, capacidade, email, horaAbertura, horaFecho, rua, codigoPostal, idZoo -> Zoo )
* **Hotel** ( idServico -> Servico, preco )
* **Restaurante** ( idServico -> Servico, tipoRestaurante, preco )
* **Animal** ( idAnimal, nome, idade, nomeComum, nomeCientifico, dataChegadaZoo, genero, idZoo -> Zoo, idProgenitor -> Animal )
* **Funcionario** ( nifFuncionario, nome, rua, codigoPostal, dataNascimento, email, numeroTelefone, idEspecialidade -> Especialidade, idZoo -> Zoo )
* **Especialidade** ( idEspecialidade, nome )
* **Habitat** ( idHabitat, area, numeroAnimais, tipoHabitat, idZoo -> Zoo )
* **Sessao** ( idSessao, nome, descricao, idZoo -> Zoo )
* **HoraioSessao** ( idHorarioSessao, horaInicio, horaFim )
* **SessaoHoraioSessao** ( idSessao -> Sessao, idHorarioSessao -> HorarioSessao )
* **Alimentacao** ( idAlimentacao, tipoAlimentacao, quantidadeDisponivel )
* **VisitanteZoo** ( nifVisitante -> Visitante, idZoo -> Zoo)
* **FuncionarioAnimalSessao** ( idAnimal -> Animal, nifFuncionario -> Funcionario, idSessao -> Sessao )
* **AnimalHabitat** ( idAnimal -> Animal, idHabitat -> Habitat )
* **QuantidadeNecessaria** ( idAnimal -> Animal, idAlimentacao -> Alimentacao, quantidadeNecessaria )

**Análise Dependências Funcionais e Formas Normais**

* **Zoo:**

FDs:

* idZoo -> nomeZoo, lotacaoMaxima, lotacaoAtual, numeroTelefone, rua, codigoPostal, horaAbertura, horaFecho, numeroTotalAnimais, area
* **Visitante:**

FDs:

* nifVisitante -> nome, idade, numeroTelefone
* **Bilhete:**

FDs:

* codigo -> horaEntrada, dataBilhete, preco, idTipoBilhete
* **TipoBilhete:**

FDs:

* idTipoBilhete -> tipoBilhete
* tipoBilhete -> idTipoBilhete
* **Servico:**

FDs:

* idServico -> nome, numeroTelefone, capacidade, email, horaAbertura, horaFecho, rua, codigoPostal, idZoo
* **Hotel:**

FDs:

* idServico -> preco
* **Restaurante:**

FDs:

* idServico -> tipoRestaurante, preco
* **Animal:**

FDs:

* idAnimal -> nome, idade, nomeComum, nomeCientifico, dataChegadaZoo, genero, idZoo, idProgenitor
* **Funcionario:**

FDs:

* nifFuncionario -> nome, rua, codigoPostal, dataNascimento, email, numeroTelefone, idEspecialidade, idZoo
* **Especialidade:**

FDs:

* idEspecialidade -> nome
* **Habitat:**

FDs:

* idHabitat -> area, numeroAnimais, tipoHabitat, idZoo
* **Sessao:**

FDs:

* idSessao -> nome, descricao, idZoo
* **HoraioSessao:**

FDs:

* idHorarioSessao -> horaInicio, horaFim
* **Alimentacao:**

FDs:

* idAlimentacao -> tipoAlimentacao, quantidadeDisponivel
* **QuantidadeNecessaria:**

FDs:

* idAnimal, idAlimentacao -> quantidadeNecessaria

As relações estão todas na 3ª Forma Normal e na Forma Normal de Boyce-Codd, porque a partir lado esquerdo de cada dependência funcional podemos obter todos os atributos da relação, ou seja, existe sempre uma (super)chave do lado esquerdo.

**Restrições**

* **Zoo:**
* Não podem existir zoos com o mesmo id:

idZoo PRIMARY KEY

* Não podem existir nomes e números de telefone iguais:

nomeZoo UNIQUE

numeroTelefone UNIQUE

* Os zoos têm sempre nome e número de telefone atribuídos:

nomeZoo NOT NULL

numeroTelefone NOT NULL

* A hora de fecho é maior do que a hora de abertura:

CHECK (horaFecho > horaAbertura)

* O número de telefone tem de ter 9 caracteres:

CHECK (LENGTH (numeroTelefone) = 9)

* O número total de animais, a lotação máxima e a lotação atual são maiores ou iguais a zero e a área é maior que zero:

CHECK (numeroTotalAnimais >= 0)

CHECK (lotacaoMaxima >= 0)

CHECK (lotacaoAtual >= 0)

CHECK (area > 0)

* A lotação máxima é maior que a lotação atual:

CHECK (lotacaoMaxima >= lotacaoAtual)

* **Visitante:**
* Não podem existir visitantes com o mesmo id:

nifVisitante PRIMARY KEY

* Não podem existir números de telefone e NIFs iguais:

numeroTelefone UNIQUE

* Os visitantes têm sempre nome e idade atribuídos:

nome NOT NULL

idade NOT NULL

* O número de telefone tem de ter 9 caracteres e o NIF 9 caracteres:

CHECK (LENGTH (numeroTelefone) = 9)

CHECK (LENGTH (nifVisitante) = 9)

* **Bilhete:**
* Não podem existir visitantes com o mesmo id:

codigo PRIMARY KEY

* O preço é maior ou igual a zero:

CHECK (preco >= 0)

* O id do visitante corresponde a um id da classe Visitante e o id do zoo corresponde a um id da classe Zoo:

nifVisitante REFERENCES Visitante (nifVisitante)

idTipoBilhete REFERENCES TipoBilhete (idTipoBilhete)

* **TipoBilhete:**
* Não podem existir tipos de bilhete com o mesmo id:

idTipoBilhete PRIMARY KEY

* Os tipos de bilhete têm sempre um tipo de bilhete atribuído:

tipoBilhete NOT NULL

* **Servico:**
* Não podem existir serviços com o mesmo id:

idServico PRIMARY KEY

* Não podem existir rua e números de telefone iguais:

numeroTelefone UNIQUE

* Os serviços têm sempre um nome, rua e número de telefone atribuídos:

nome NOT NULL

rua NOT NULL

numeroTelefone NOT NULL

* A hora de fecho é maior do que a hora de abertura:

CHECK (horaFecho > horaAbertura)

* O número de telefone tem de ter 9 caracteres:

CHECK (LENGTH (numeroTelefone) = 9)

* O id do zoo corresponde a um id da classe Zoo:

idZoo REFERENCES Zoo (idZoo)

* **Hotel / Restaurante:**
* Não podem existir hotéis/restaurantes com o mesmo id e estes ids correspondem a um id da classe Servico:

idServico PRIMARY KEY REFERENCES Servico (idServico)

* O preço tanto dos hotéis como dos restaurantes é maior ou igual a zero:

CHECK (preco >= 0)

* **Animal:**
* Não podem existir animais com o mesmo id:

idAnimal PRIMARY KEY

* Os animais têm sempre nome, nome comum e nome científico atribuídos:

nome NOT NULL

nomeComum NOT NULL

nomeCientifico NOT NULL

* O id do zoo corresponde a um id da classe Zoo e o id do progenitor corresponde a um id da classe Animal:

idZoo REFERENCES Zoo (idZoo)

idProgenitor REFERENCES Progenitor (idAnimal)

* **Funcionario:**
* Não podem existir funcionários com o mesmo id:

nifFuncionario PRIMARY KEY

* Não podem existir números de telefone, NIFs e emails iguais:

numeroTelefone UNIQUE

email UNIQUE

* Os funcionários têm sempre nome e número de telefone atribuídos:

nome NOT NULL

numeroTelefone NOT NULL

* O número de telefone tem de ter 9 caracteres e o NIF 9 caracteres:

CHECK (LENGTH (numeroTelefone) = 9)

CHECK (LENGTH (nifVisitante) = 9)

* O id do zoo corresponde a um id da classe Zoo e o id da especialidade corresponde a um id da classe Especialidade:

idZoo REFERENCES Zoo (idZoo)

idEspecialidade REFERENCES Especialidade (idEspecialidade)

* **Especialidade:**
* Não podem existir especialidades com o mesmo id:

idEspecialidade PRIMARY KEY

* A especialidade tem sempre um nome atribuído:

nome NOT NULL

* **Habitat:**
* Não podem existir habitats com o mesmo id:

idHabitat PRIMARY KEY

* Os habitats têm sempre um tipo:

tipoHabitat NOT NULL

* O número de animais é maior ou igual a zero e a área é maior que zero:

CHECK (numeroAnimais >= 0)

CHECK (area > 0)

* O id do zoo corresponde a um id da classe Zoo:

idZoo REFERENCES Zoo (idZoo)

* **Sessao:**
* Não podem existir sessões com o mesmo id:

idSessao PRIMARY KEY

* As sessões têm sempre nome:

nome NOT NULL

* O id do zoo corresponde a um id da classe Zoo:

idZoo REFERENCES Zoo (idZoo)

* **HorarioSessao:**
* Não podem existir horários de sessões com o mesmo id:

idHorarioSessao PRIMARY KEY

* Os horários das sessões têm sempre nome:

nome NOT NULL

* A hora de fim é maior do que a hora de início:

CHECK (horaFim > horaInicio)

* **SessaoHorarioSessao:**
* Não podem existir instâncias com o mesmo conjunto {idSessao, idHorarioSessao}:

PRIMARY KEY (idSessao, idHorarioSessao)

* O id da sessão corresponde a um id da classe Sessao, o id do horário da sessão corresponde a um id da classe HorarioSessao:

idSessao REFERENCES Sessao (idSessao)

idHorarioSessao REFERENCES HorarioSessao (idHorarioSessao)

* **Alimentacao:**
* Não podem existir alimentações com o mesmo id:

idAlimentacao PRIMARY KEY

* A alimentação tem sempre um tipo e quantidade disponível:

tipoAlimentacao NOT NULL

quantidadeDisponivel NOT NULL

* A quantidade disponível é maior ou igual a zero:

CHECK (quantidadeDisponivel >= 0)

* **VisitanteZoo:**
* Não podem existir instâncias com o mesmo conjunto {nifVisitante, idZoo}:

PRIMARY KEY (nifVisitante, idZoo)

* O id do visitante corresponde a um id da classe Visitante e o id do zoo corresponde a um id da classe Zoo:

nifVisitante REFERENCES Visitante (nifVisitante)

idZoo REFERENCES Zoo (idZoo)

* **FuncionarioAnimalSessao:**
* Não podem existir instâncias com o mesmo conjunto {nifFuncionario, idAnimal, idSessao}:

PRIMARY KEY (nifFuncionario, idAnimal, idSessao)

* O id do funcionário corresponde a um id da classe Funcionário, o id do animal corresponde a um id da classe Animal e o id da sessão corresponde a um id da classe Sessao:

nifFuncionario REFERENCES Funcionario (nifFuncionario)

idAnimal REFERENCES Animal (idAnimal)

idSessao REFERENCES Zoo (idSessao)

* **AnimalHabitat:**
* Não podem existir instâncias com o mesmo conjunto {idAnimal, idHabitat}:

PRIMARY KEY (idAnimal, idHabitat)

* O id do animal corresponde a um id da classe Animal e o id do habitat corresponde a um id da classe Habitat:

idAnimal REFERENCES Animal (idAnimal)

idHabitat REFERENCES Habitat (idHabitat)

* **QuantidadeNecessaria:**
* Não podem existir instâncias com o mesmo conjunto {idAnimal, idAlimentacao}:

PRIMARY KEY (idAnimal, idAlimentacao)

* O id do animal corresponde a um id da classe Animal e o id da alimentação corresponde a um id da classe Alimentacao:

idAnimal REFERENCES Animal (idAnimal)

idAlimentacao REFERENCES Alimentacao (idAlimentacao)

* A quantidade necessária não pode ser nula:

quantidadeNecessaria NOT NULL

* A quantidade necessária tem de ser maior ou igual a zero:

CHECK (quantidadeNecessaria >= 0)

**Autoavaliação**

Daniela Tomás: 33,3%

Nuno Penafort: 33,3%

Sofia Sousa: 33,3%