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

João Miguel Rocha da Silva

Estudantes & Autores:

Alexandre Miguel de Araújo Carqueja up201705049 up201705049@fe.up.pt

David Freitas Dinis up201706766 up201706766@fe.up.pt

Margarida Alves Pinho up201704599 up201704599@fe.up.pt

Índice

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.

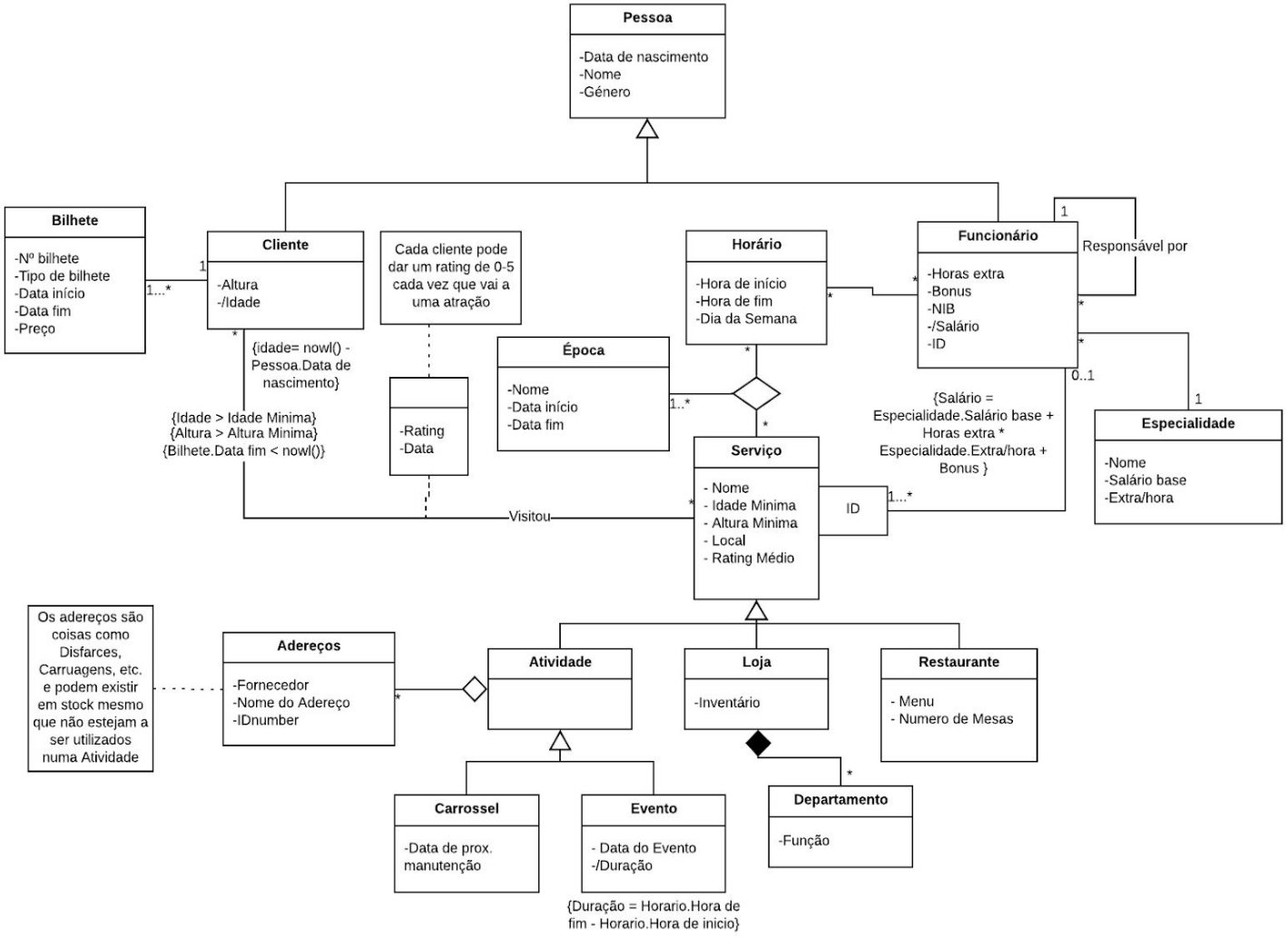
Para os clientes guardamos informações relativas aos seus bilhetes, bem como um registo de todos os serviços de que usufruíram, e ainda o grau de satisfação em cada um deles. Cada vez que os clientes visitam o parque só podem adquirir um bilhete no qual estará especificado o seu tipo e preço, bem como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

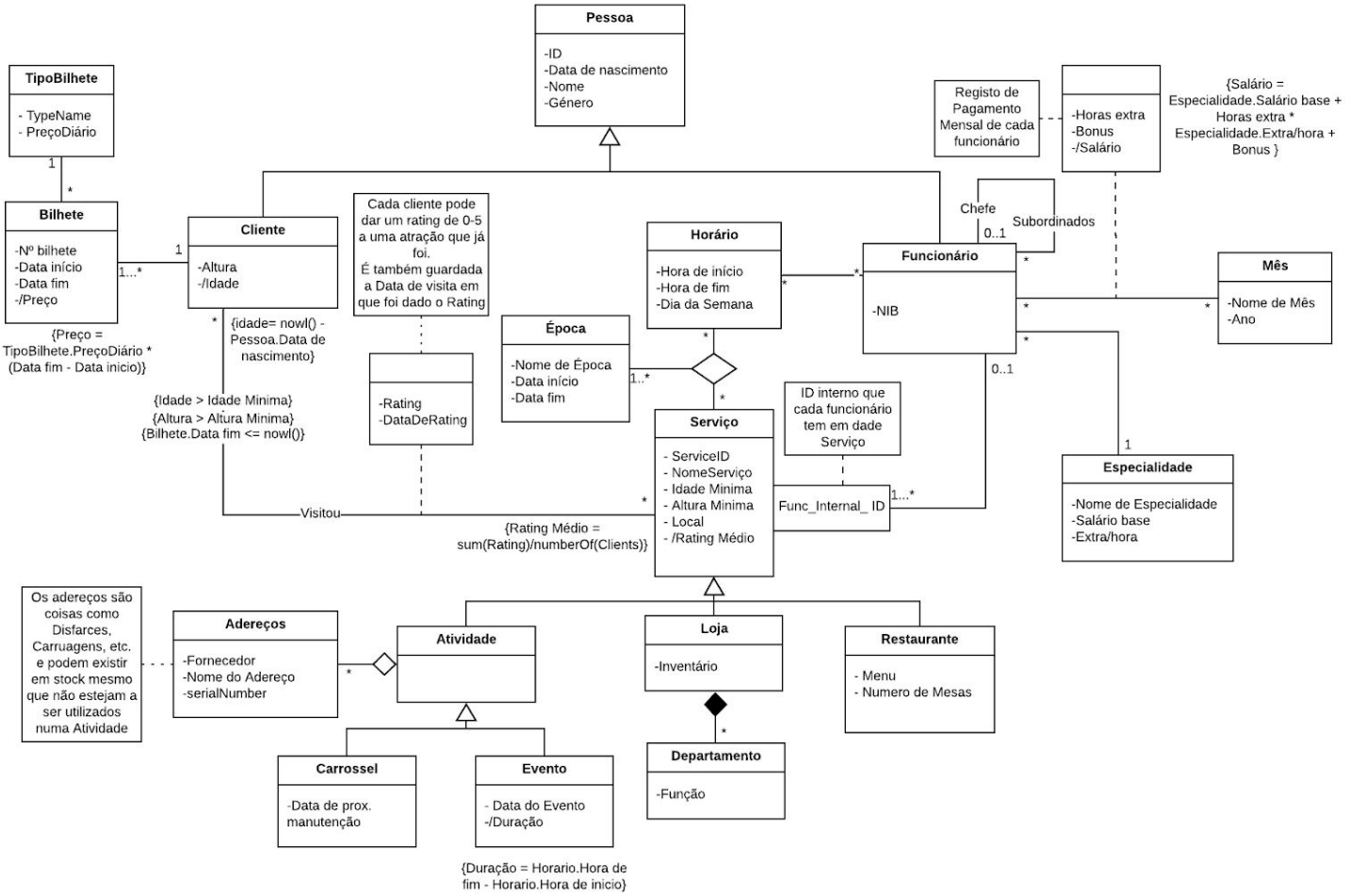
Finalmente, para todos os serviços do Parque (Lojas, Atividades, Restaurantes), conseguimos obter qual o seu horário de funcionamento, dependendo da época do ano, bem como impor condições sobre os clientes que os podem frequentar, segundo a sua idade, altura e data de validade do bilhete. Tanto os Carrosséis 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)
ID é chave primária.

Clientes(ID->Pessoa, Altura, Idade)
ID é chave primária e estrangeira para Pessoas.

TipoDeBilhete(TypeName, PreçoDiario)
TypeName é chave primária.

Bilhetes(Nbilhete, typeName->TipoBilhete, DataInicio, DataFim, Preço, IDcliente->Cliente)
Nbilhete é chave primária.
clienteID é chave estrangeira para Cliente.
typeName é chave estrangeira para TipoDeBilhete.

Especialidades(NomeEspecialidade, SalárioBase, Extra/hora)
NomeEspecialidade é chave primária.

Funcionários(ID->Pessoa, NIB, NomeEspecialidade->Especialidades)
ID é chave primária e estrangeira para Pessoa.
NomeEspecialidade é chave estrangeira.

Mês(IDmes, NomeDeMes, Ano)
IDmes é chave primária.

RegistoPagamentos(IDmes->Mes, ID->Funcionário, HorasExtra, Bonus, Salário)
IDmes e ID são o composto da chave primária, e ambas são chaves estrangeiras(Para Mês e Funcionários).

Hierarquias(Subordinado->Funcionário, Chefe->Funcionário)
Subordinado é chave primária e estrangeira para Funcionários.
Chefe é chave estrangeira para Funcionários.

Horários(IDhorario, HoraInicio, HoraFim, DiaSemana)
IDhorario é chave primária.

HoráriosTrabalho(IDhorario->Horário, ID->Funcionário)
IDhorario e ID são o composto da chave primária, e ambas são chaves estrangeiras (para Horário e Funcionários).

Épocas(NomeDeEpoca, DataInicio, DataFim)
NomeDeEpoca é chave primária.

Serviços(ServiceID, NomeDeServiço, IdadeMinima, AlturaMinima, Local, RatingMedio)
ServiceID é chave primária.

ServiçosÉpocasHorários(ServiceID->Serviço, NomeDeEpoca->Época, IDhorario->Horário)
ServiceID, NomeDeEpoca e IDhorario são o composto da chave primária (E todos são chaves estrangeiras, para Serviços, Épocas e Horários).

FuncionárioDeServiço(ServiceID->Serviço, IDfuncionario->Funcionários, Func_Internal_ID)
{Func_Internal_ID, ServiceID} **UK**
IDfuncionario e ServiceID são o composto da chave primária (e ambas são chaves estrangeiras para Serviço e Funcionários).

Visitas(IDcliente->Cliente, ServiceID->Serviço, DataDeRating, Rating)
ID e ServiceID são o composto da chave primária (e ambas são chaves estrangeiras para Clientes e Serviços).

Lojas(StoreServiceID->Serviço, Inventário)
StoreServiceID é chave primária e estrangeira para Serviços.

Restaurantes(RestaurantServiceID->Serviço, Menu, Nmesas)
RestaurantServiceID é chave primária e estrangeira para Serviços.

Atividades(ActivityServiceID->Serviço)
ActivityServiceID é chave primária e estrangeira para Serviços.

Carrosseis(CarrousselServiceID->Atividade, DataProxManutenção)
CarrousselServiceID é chave primária e estrangeira para Atividades.

Eventos(EventServiceID->Atividade, DataDeEvento, Duração)
EventServiceID é chave primária e estrangeira para Atividades.

Adereços(SerialNumber, Fornecedor, NomeDeAdereço, ActivityServiceID->Atividade)
SerialNumber é chave primária.
ACtivityServiceID é chave estrangeira para Atividade.

Departamentos(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, IDfuncionário} -> {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 das Dependências Funcionais

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, IDfuncionário} -> {Func_Internal_ID}
  {ServiceID, Func_Internal_ID} -> {IDfuncionário}
  {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 {IDfuncionário} faz parte da chave {ServiceID, IDfuncionário}.

Interrogações

1- Para cada género, obter as faixas etárias mais prevalentes de clientes.

2- Obter as Atividades com melhor e pior rating médio.

3- Obter as épocas ordenadas por mais bilhetes comprados.

4- Para o último mês em que houve pagamentos e para cada especialidade saber quais os funcionários que receberam um ordenado superior a 2500€ (E dar a informação que decompõe esse salário, como as horas extra, salário base, etc.)

5- Obter todos os clientes que foram só aos carrosséis. (E o nome dos carrosséis)

6- Obter todos os carrosséis com data de manutenção no próximo ano (Nome de carrossel; data a fazer a manutenção).

7- Obter todos os funcionários que trabalham aos fins de semana e os seus horários todos.

8- Obter todos os chefes e seus subordinados empregados, bem como todos os outros sub-chefes abaixo de cada Chefe

9- Obter o nº de dias médio que os clientes visitam o nosso parque (Bem como o máximo que alguma vez alguém comprou).

10- Obter a lista de eventos ordenada por ordem decrescente de rating médio.

Gatilhos

1- Trigger de Preenchimentos de elementos derivados

Este ficheiro tem vários gatilhos que preenchem/atualizam elementos derivados da nossa base de dados, que apenas podem ser calculados e não inseridos.

1.1- Preencher o Rating médio dos Serviços

Na classe Serviços 'Rating médio' é um atributo derivado, por isso foi necessário implementar um gatilho para preencher a tabela Serviços. Assim, partir da informação guardada na tabela Visitas (a classificação dada por cada cliente) é possível obter o rating médio de cada serviço .

1.2- Preencher o Salário no registo de pagamentos

Na tabela 'RegistoPagamentos' é guardada a informação relativa ao pagamento de salários (o que um determinado funcionário recebeu num certo mês de um dado ano). Cada salário é calculado a partir de outros dados como o salário base da especialidade do funcionário; as horas extras feitas e quanto se ganha por cada hora extra na especialidade do funcionário; e outros eventuais bonus. Por isso criou-se um gatilho que preenche o Salário no registo de pagamentos.

1.3. Preencher a idade dos clientes

Na classe Cliente 'idade' é um atributo derivado, assim sendo para preencher a tabela Clientes foi necessário adicionar um gatilho que calcula a idade, a partir da data de nascimento e da data atual, e que a adiciona à base de dados.

1.4. Preencher o preço de compra de cada bilhete

Foi necessário adicionar outro gatilho que preencha o preço na tabela 'Bilhete'. Este gatilho calcula o valor pretendido através da multiplicação do preço diário (tendo em conta o tipo de bilhete) pelo número de dias da visita.

2. Verificar se as características dos Clientes e seus Bilhetes é válida para cada visita

Este gatilho verifica, para cada INSERT INTO na tabela de Visitas, se os clientes que visitaram um certo serviço o fizeram dentro das datas de início e fim do seu bilhete, e que têm altura e idade mínima para frequentar o serviço em questão.

3. Propagar cada Inserção numa Vista para as tabelas apropriadas

Este gatilho em vez de realizar um INSERT INTO para a vista ServicoComEmpregados, insere os dados corretos nas tabelas da nossa base de dados.