Universidade do Porto
FEUP Faculdade de Engenharia

Festival de Música

Turma 4 Grupo 4

Afonso Sá – up201604605

André Serralheiro – up201604566

Luís Marques – up201104354

TEMA

Num festival de música atuam várias bandas. As bandas atuam em diferentes palcos, que podem estar associados a diferentes géneros de música. Um palco pode ter mais do que um género de música associado, mas, por norma, certos géneros só atuam em palcos associados aos mesmos. Para além disso, em cada palco só pode tocar uma banda de cada vez, como tal, há um horário que deve ser cumprido.

Para o bom funcionamento do festival, existem várias infraestruturas (WC, zona de restauração, etc). Em cada infraestrutura há staff disponível para gerir e realizar a manutenção da mesma. Há staff interno que é gerido pelos organizadores, mas, de modo a facilitar a organização, há empresas subcontratadas, bem como, empresas patrocinadoras do festival, que também disponibilizam staff. Os diferentes palcos fazem parte das infraestruturas, necessitando também de manutenção. A cada elemento do staff é disponibilizado um número identificativo (Id). Para além disso, cada um possui também um atributo de disponibilidade que permite identificar, em casos de indisponibilidade desse elemento para a realização do seu turno, alguém que o possa substituir.

Os bilhetes para o festival são disponibilizados para cada dia ou para o evento completo (bilhete geral), sendo que cada bilhete tem o seu ID único. O cliente pode comprar vários bilhetes, tanto diários, como gerais.

Com o objetivo de facilitar a criação de uma nova base de dados para anos seguintes, caso se queira realizar o festival, a própria informação do festival é guardada (Nome, Localidade, Website, Contacto).

Diagrama UML

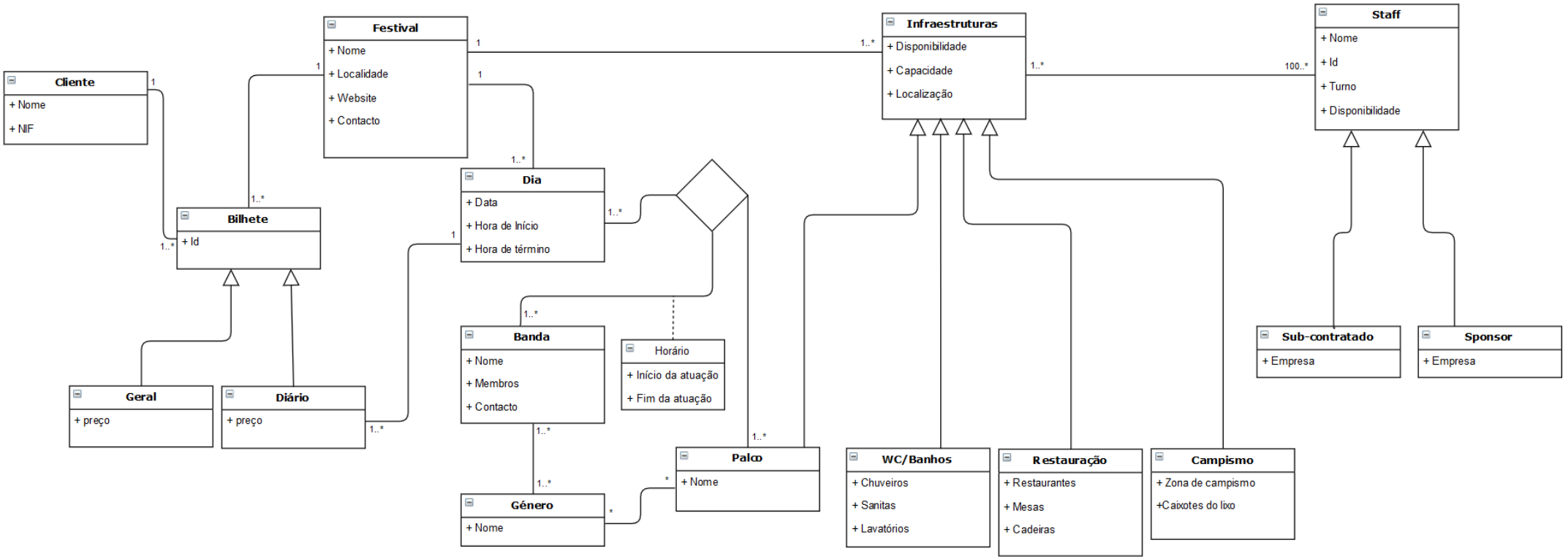
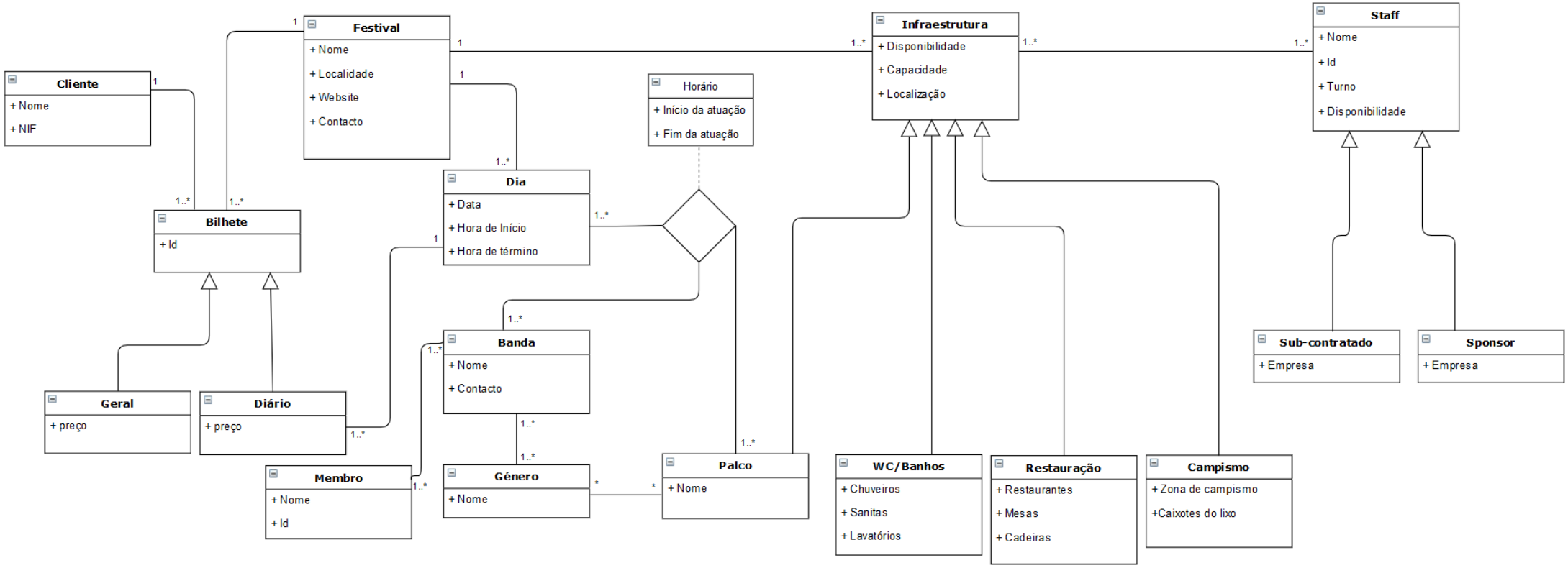


Diagrama UML (Atualizado)



Mudanças ao UML

As mudanças realizadas no diagrama UML foram realizadas de acordo com as sugestões do professor, excetuando uma sugestão que discordamos e vamos proceder a defender o nosso ponto de vista.

No diagrama foi nos sugerido que removêssemos a classe Festival, no entanto, na nossa opinião esta classe deverá ser mantida uma vez que a mesma organização pode querer realizar um novo festival num ano diferente ou numa localidade diferente, como tal, pode manter a sua organização da base de dados.

Para além disso, adicionamos uma nova classe chamada Membro que está ligada a banda com uma ligação complete many-to-many. Esta classe serve de substituição do atributo da banda chamado membros e foi criada com o intuito de facilitar o modo como os dados relativos aos membros da banda são guardados em SQL.

Contexto

Classes:

- Festival:
  Atributos: Nome, Localidade, Website e Contacto.
  Explicação: Um festival é identificado pelo seu nome e localidade onde este é realizado. É também disponibilizado ao público um website e contacto de modo a facilitar o contacto do público com os organizadores.

- Bilhete:
  Atributos: Id (identificação)
  Explicação: A classe bilhete é o que permite ao cliente a entrada no festival.

- Geral:
  Atributos: Preço
  Explicação: Um bilhete geral permite ao cliente acesso completo ao festival, tendo assim acesso às atuações e campismo durante todos os dias.

- Diário:
  Atributos: Preço
  Explicação: Um bilhete diário permite ao cliente acesso ao festival, tendo assim acesso às atuações e campismo desse dia. Um bilhete diário está associado a um só dia.

- Cliente:
  Atributos: Nome, NIF.
  Explicação: O cliente é identificado pelo seu nome e NIF, e pode comprar vários bilhetes, tanto gerais como diários.

- Infraestruturas:

  Atributos: Disponibilidade, Capacidade e Localização.

  Explicação: O atributo disponibilidade permite identificar a infraestrutura que está com o serviço interrompido por uma de várias razões (manutenção, limpezas, etc.), localização (onde essa infraestrutura de localiza no próprio festival) e capacidade (número de pessoas que podem estar ao mesmo tempo numa dada infraestrutura.

- WC/Banhos:

  Atributos: Chuveiros, sanitas e lavatórios.

- Restauração:

  Atributos: Restaurantes, mesas e cadeiras.

- Campismo:

  Atributos: Zona de campismo e caixotes do lixo.

- Staff:

  Atributos: Nome, id, turno e disponibilidade.

  Explicação: O staff tem um id único que os distingue assim como nome turno e disponibilidade. O atributo disponibilidade permite identificar algum membro que esteja disponível para o caso de um outro membro falhe o seu turno e este o possa substituir.

- Subcontratado:

  Atributo: Empresa.

  Explicação: Membro do staff que é disponibilizado por uma empresa que disponibiliza o staff. Esta empresa é contratada pela organização do festival.

- Sponsor:

  Atributo: Empresa.

  Explicação: Membro do staff que é disponibilizado por uma empresa que que patrocina o festival.

- Dia:

  Atributos: Data, hora de início, hora de término.

  Explicação: Um dia no qual acontece o festival, contém dois atributos de horas, do início do festival nesta data e do fim.

- Banda:

  Atributos: Nome, Contacto
  Explicação: Uma banda que atua no festival, contém informação sobre o seu nome, membros e contacto. Está associada a palcos e dias através de uma relação tripla, utilizando um horário. Está também associada a um ou mais géneros musicais.

- Membro:

  Atributos: Id, Nome
  Explicação: Cada banda é composta por um ou mais membros, esta classe substitui o atributo Membros na classe Banda. É guardado o nome do membro e o seu Id para o relacionar à banda.

- Género:

  Atributos: Nome
  Explicação: Género musical, está associado às bandas e aos palcos.

- Palco:

  Atributos: Nome
  Explicação: Infraestrutura onde ocorrem os concertos, associado a géneros musicais e associado a dias e bandas em relação tripla através de um horário.

Esquema Relacional

Cliente (NIF, Nome);

{NIF} → {Nome}

Bilhete (id, NIF→Cliente);

{id} → {NIF→Cliente}

BilheteGeral (id→Bilhete, Preço);

{id→Bilhete} → {Preço}

BilheteDiário (id→Bilhete, Preço, Data→Dia);

{id→Bilhete} → {Preço, Data→Dia}

Dia (id, Data, HoraInício, HoraTérmino);

{id} → {Data, HoraInício, HoraTérmino}

Banda (Nome, Contacto, id→Membro);

{Nome} → {Contacto, id→Membro}

{Contacto} → {Nome}

Membro (id, Nome, Nome→Banda);

{id} → {Nome, Nome→Banda}

Horário(Dia→Data, Início, Fim, id→Palco, Nome→Banda);

{Dia→Data, id→Palco, Nome→Banda} → {Início, Fim}

Género (id, Nome, id→Palco, Nome→Banda);

{id} → {Nome, id→Palco, Nome→Banda}

{Nome→Banda, id→Palco} → {Nome}

GéneroPalco (id→Género, id→Palco);

GéneroBanda (id→Género, Nome→Banda);

Palco (id→Infraestrutura, Nome);

{id→Infraestrutura} → {Nome}

Infraestrutura (id, Disponibilidade, Capacidade, Localização);

{id} → {Disponibilidade, Capacidade, Localização}

WC (id→Infraestrutura, Chuveiros, Sanitas, Lavatórios);

{id→Infraestrutura} → {Chuveiros, Sanitas, Lavatórios}

Restauração (id→Infraestrutura, Restaurantes, Mesas, Cadeiras);

{id→Infraestrutura} → {Restaurantes, Mesas, Cadeiras}

Campismo (id → Infraestrutura, Zona, CaixoteLixo);

{id→Infraestrutura} → {Zona, CaixoteLixo}

Staff (id, Nome, Turno, Disponibilidade);

id → Nome

Nome → Turno, Disponibilidade

Sub-Contratado (id→Staff, Empresa);

{id→Staff} → {Empresa}

Sponsor (id → Staff, Empresa);

{id→Staff} → {Empresa}

Dependências Funcionais e Formas Normais

Para haver violações à forma normal Boyce-Codd é necessário identificar relações onde o(s) atributo(s) da esquerda não é(são) (super)key. Logo, no nosso modelo conseguimos encontrar duas classes onde tal acontecia. Essas classes são a classe Banda e Staff, onde o atributo Contacto e o atributo Nome, respetivamente, não são key, e como tal violam então a forma normal Boyce-Codd, bem como, consequentemente, violam a terceira forma normal.

As restantes relações não violam nenhuma destas formas, visto que seguem as várias regras das mesmas, ou seja, os atributos da esquerda são (super)key e segue a regra da não-transitividade.

Restrições à base de dados

Para garantir uma base de dados consistente e acessível, foram usadas várias restrições como a restrição chave, de integridade referencial, NOT NULL e UNIQUE.

O uso de PRIMARY KEY é essencial para qualquer base de dados, uma vez que garante a identificação de cada instância da mesma classe.

- Em cada classe há um atributo id que é único e é chave dessa classe. Com o id é possível saber todos os restantes atributos.

A restrição NOT NULL é útil pois há classes que requerem o preenchimento de atributos para essa classe justificar a sua existência.

- Em subclasses como *GeneroPalco*, *GeneroBanda* e *StaffInfrastrutura* em que se os atributos forem nulos a existência destas subclasses deixa de fazer sentido.

Já a restrição UNIQUE é usada para atributos que não sendo chaves primárias são únicos para várias instâncias.

- Na classe *Cliente* o atributo NIF é diferente de qualquer outro cliente.

Decisões/Dificuldades (Entrega I)

A maior dificuldade que tivemos foi a decisão do modo como se fariam as ligações entre palco e género. No início tínhamos pensado de modo a que cada palco teria de ter um género associado, no entanto pode haver palcos sem géneros musicais associados, como por exemplo um palco de comédia. Como tal mudamos as ligações de 1..* para * (many-to-many).

Decisões/Dificuldades (Entrega II)

Primeiramente, uma das decisões desta entrega foi o facto de termos mantido a classe Festival no nosso diagrama UML, uma vez que para nós faz sentido em termos de organização manter a classe, como já explicamos previamente.

Uma das dificuldades sentidas a realizar este trabalho foi criar as relações das classes e analisar as dependências funcionais das mesmas. Esta dificuldade foi sentida uma vez que nesta área foram feitas decisões que iam moldar a nossa base de dados, e como tal, queríamos efetuar decisões que fizessem sentido, mas que fossem, também, práticas.