Projeto Assessoria de Eventos de Festa - Parte 3

Sistema de Gerenciamento de Imagem e Som para Festas

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)

Disciplina: SCC240 Base de Dados

Profa. Dra. Elaine Parros M. de Sousa

Turma 1

Integrantes do Grupo	**NoUSP**
Bruno Mendes da Costa	9779433
Felipe Alves Siqueira	9847706
Josué Grâce Kabongo Kalala	9770382
Matheus Henrique Junqueira Saldanha	9763234

Data de Entrega: 25/06/2018

---

1 Descrição do Problema

Somos a seção de TI de uma Empresa de Imagem e Som para Eventos (EISE) em Parques de Diversão e Cruzeiros. Foi solicitada a modelagem do sistema responsável por gerenciar todos os dados com que esta EISE trabalha, referentes a todos os elementos envolvidos para a produção de conteúdo multimídia, ou seja, fotos, vídeos, som e música.

Cada festa contratada possui nome, data de início, data de fim e número de convidados. Para prover os serviços de multimídia, a EISE conta com diversos funcionários dos quais dados pessoais são armazenados na base de dados, a saber: nome, CPF, RG, telefones, endereço, as equipes as quais eventualmente podem pertencer durante a execução de suas tarefas, e sua função dentro da empresa.

Vários aspectos do trabalho realizado pela EISE envolvem equipamentos, que podem ser divididos em câmeras, *drones*, sonoros e de estruturação. Todo equipamento possui nome, descrição, tipo, a marca e o modelo do equipamento. Não é necessária a distinção entre cada unidade física de cada tipo e modelo de equipamento na base de dados, logo deseja-se gerenciar cada item através de um cadastro de quantidades de itens disponíveis. Em particular, as câmeras possuem diversos atributos específicos que deseja-se armazenar, a saber: visão noturna, certificação IP (indica o grau de resistência a poeira e a água), fonte de alimentação, conectividade (*e.g. WiFi* ou cabo), grau de resistência a queda, zoom de longo alcance e estabilizador de imagem.

Com os dados armazenados na base, deverá ser possível gerar um relatório de todos os funcionários que foram escalados para uma determinada festa. Um caso particular são os técnicos, cujas tarefas serão especificadas posteriormente, que podem trabalhar mesmo não sendo escalados para uma festa específica e, portanto, não aparecem no relatório de festa alguma.

1.1 Festa no Parque de Diversões

Uma festa no parque ocorre em um parque de diversões, que possui CNPJ, nome, endereço e um mapa, que é uma imagem de satélite no qual se projeta um sistema de coordenadas. Essas festas são identificadas por sua data de início e pelo CNPJ do parque onde ocorrem; e seus endereços, para onde são enviados os equipamentos e os funcionários, são os endereços dos parques onde elas ocorrem. Atrações dentro de um parque mudam de festa para festa, e possuem nome, número (para quando houver atrações de mesmo nome) e área, que é uma poligonal. Dentro do parque há pontos de instalação, como postes e torres, onde serão instalados sistemas de som e câmeras; esses pontos possuem descrição, coordenadas e outros atributos a serem expostos posteriormente.

Para a realização da filmagem e fotografia de uma festa no parque de diversões foi estabe-

lecido que serão utilizados sistemas de câmera e *drones*. A captura de imagens poderá ser feita de forma aérea, por meio dos *drones*, tanto filmagem quanto fotografia, ou através de câmeras que serão fixadas nos pontos de instalação do parque de diversões. Em ambos os casos a operabilidade sobre os equipamentos será remota, com equipes separadas para a gravação de imagens com os *drones* e outra gerenciando os sistemas para filmagem.

Em relação as imagens aéreas do evento, haverá uma equipe de *drone* responsável por cada *drone*. Esta equipe será composta por um piloto auxiliado por um co-piloto, um operador de câmera auxiliado por um assistente, *drones* reservas, câmeras e técnicos. Os técnicos da equipe de *drones*, dependendo de sua especialidade, serão responsáveis pela instalação de uma ou mais câmeras nos *drones*, reparos, manutenção, resgate, trocas de baterias e qualquer tipo de ajuste que seja necessário antes ou durante a festa, mantendo na base de dados a data da manutenção. Vale mencionar também que todos os nossos *drones* são regulamentados pela ANAC, ou seja, possuem um número de registro único; e também possuem tempo máximo de voo, alcance de comunicação remota, e o tipo de fonte de alimentação suportado. Ademais, todos os pilotos possuem número de habilitação.

No que tange aos sistemas de filmagem em pontos fixos, haverá operadores de câmera, uma equipe de suporte, composta de vários assistentes onde cada assistente só auxilia um operador, e uma equipe de manutenção, composta de técnicos, que será responsável pela instalação e manutenção das câmeras nos pontos de instalação. É desejável saber o tempo de experiência dos operadores de câmera, armazenando o início da carreira de cada um, a fim de escalá-los para serviços que exijam mais experiência quando necessário. Dependendo das características dos pontos de instalação, câmeras com diferentes atributos podem ser necessárias, o que reforça a necessidade de se armazenar os diversos atributos já mencionados das câmeras. Além disso, a respeito dos pontos de instalação, deseja-se também armazenar no banco de dados informações sobre iluminação, conectividade, risco de contato com água e tipo de fonte de alimentação elétrica.

A partir do sistema de banco de dados, será possível consultar quais operadores de câmera trabalharam com quais modelos de câmeras e em quais festas, as características dessas câmeras, *drones* utilizados, pilotos e co-pilotos responsáveis pelos *drones* e também os assistentes que auxiliaram os operadores na festa.

Quanto ao sistema de som das festas em parques, a equipe responsável por seu planejamento almeja a imersão dos participantes da festa por meio do sistema de som. Para atingir isso, é preciso ter uma boa noção de todas as fontes de som instaladas no parque e de suas localizações.

Para uma dada festa no parque, equipamentos sonoros e de estruturação (cabeamento, por exemplo) são instalados nos pontos de instalação. O cabeamento de som instalado permite mul-

tiplexar diferentes sons para diferentes pontos de som, então deseja-se que seja possível agrupar os pontos de som em grupos numerados, como se fossem grafos. Além disso, é preciso saber a potência dos equipamentos sonoros instalados nos pontos de instalação; e, no contexto de festa no parque, todos tais equipamentos são posse da EISE.

Com o banco de dados, deverá ser possível consultar os equipamentos sonoros e de estruturação instalados em um parque, suas coordenadas, suas especificações e o grafo numerado ao qual cada um pertence.

1.2 Festa no Cruzeiro

Festas em cruzeiro são identificadas pela data de início e pelo número IMO (identificador de navios) do cruzeiro onde ocorreu; e também possuem os locais efetivamente utilizados na festa. O serviço de filmagem e fotografia oferecido pela EISE envolve uma equipe de fotógrafos, que produzem o álbum da festa, e uma de cinegrafistas, que produzem o *making of*.

Dentro da equipe de fotografia no cruzeiro, distinguimos três categorias, dependendo do tempo de experiência de cada integrante: fotógrafo especialista, fotógrafo técnico e fotógrafo júnior. Os fotógrafos especialistas são aqueles que de fato fotografam, e a eles são alocadas câmeras do estoque de equipamentos da EISE para utilizarem. Os fotógrafos júnior são os responsáveis por dar assistência aos especialistas no manuseamento dos equipamentos de fotografia. Os fotógrafos técnicos são aqueles que trabalham no laboratório de fotografia realizando tarefas de edição de foto e montagem de álbum. Os nomes de cada fotógrafo são registrados como participantes da cobertura além de aparecerem na lista dos fotógrafos de cada álbum final. O álbum produzido deve ser armazenado, assim como os fotógrafos que o produziram, suas categorias e o nome da festa.

Ainda no cruzeiro, a equipe de filmagem acompanha a equipe de fotografia e é responsável por realizar o *making of* do evento. O *making of* é o conjunto de todos os momentos destaques do evento, que inclui a filmagem do navio, das excursões e das diferentes atividades dentro do navio. A equipe de filmagem trabalhando no evento pode conter até cinco pessoas dependendo do número de convidados para a festa. O *making of* produzido deve ser armazenado, junto com os nomes dos cinegrafistas que o produziram e o nome da festa.

Dessa forma, é possível consultar quais fotógrafos e cinegrafistas trabalharam num determinado evento assim como acessar o álbum e o *making of* que lhe pertence.

Quanto aos elementos de som no cruzeiro, tem-se um show ao vivo realizado por bandas todos os dias. Cada show possui sua data armazenada no banco. As horas de início e fim previstas também devem ser armazenadas, pois podem variar entre os shows. No banco também são armazenados, para cada banda participante da festa, o nome, a data de criação e o estilo musical.

É desejável que as bandas variem todos os dias da viagem, logo cada uma delas só pode realizar um único show em cada festa no cruzeiro. Armazena-se também o nome e o CPF dos músicos que compõem cada uma das bandas.

A EISE possui bandas particulares, as quais podem ser usadas para oferecer suporte a músicos contratados. Caso ocorra algum imprevisto relacionado a banda contratada, uma banda particular realiza um *cover*. Não há obrigatoriedade de embarcação de um número exato de bandas particulares em cada cruzeiro. É estabelecido um contrato com uma identificação numérica única com cada banda não ligada a EISE que participará da festa.

Toda infraestrutura de som necessária que se encontrar ausente na embarcação, como sintetizadores, caixas de som, microfones e instrumentos musicais, devem ser providenciados pela EISE. Armazena-se todos os equipamentos usados em cada show, sendo identificados explicitamente quais pertencem a EISE.

Assim como em festas no parque de diversões, em todo equipamento de imagem ou som que forem necessárias manutenções, reparos ou testes, a EISE providenciará um ou mais técnicos de sua equipe para o serviço, seja antes ou durante a festa, mantendo a data da manutenção realizada.

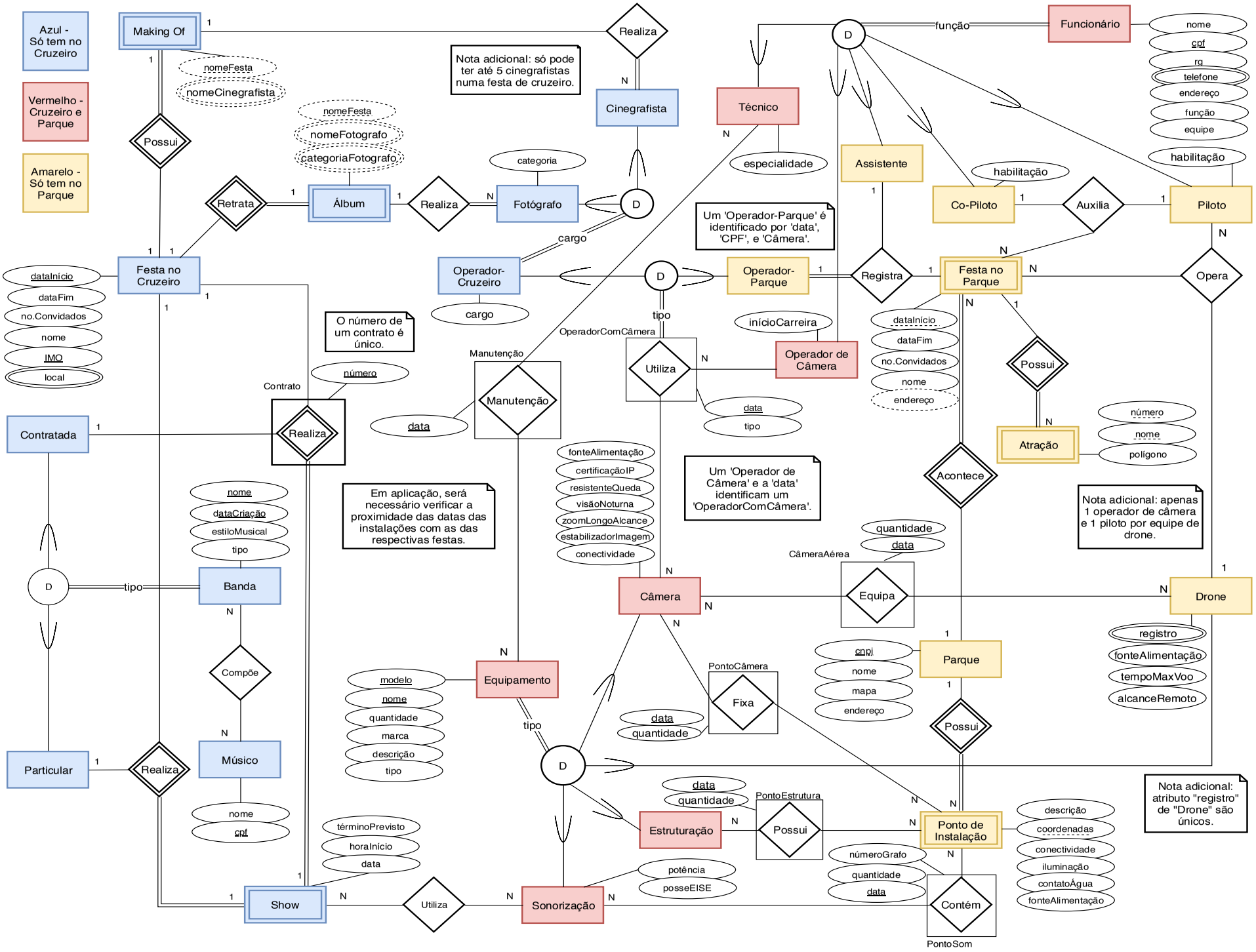
Com este histórico armazenado, será possível consultar quais bandas e quais artistas trabalharam com a EISE.

1.3 Notas adicionais

- Considera-se que cada operador de câmera só pode trabalhar em uma única festa por dia.
- O processo de mapear os pontos de instalação de um parque é demorado e custoso, e por isso deseja-se armazenar o histórico de instalações de câmeras, sonorização e estruturação em pontos de instalação, de forma que se possa reutilizar esses dados caso outra festa seja realizada no mesmo parque.
- Deseja-se também armazenar o histórico de técnicos que fizeram manutenção em cada modelo de *drone*, para fins de avaliação do trabalho deles; e pelos mesmos motivos, deseja-se armazenar histórico de quais operadores de câmera trabalharam em cada festa.
- Por fim, deseja-se armazenar quais câmeras estavam instaladas em um dado modelo de *drone* em cada festa, para fins de controle de orçamento e projeção de orçamentos futuros.

2 Diagrama Entidade-Relacionamento

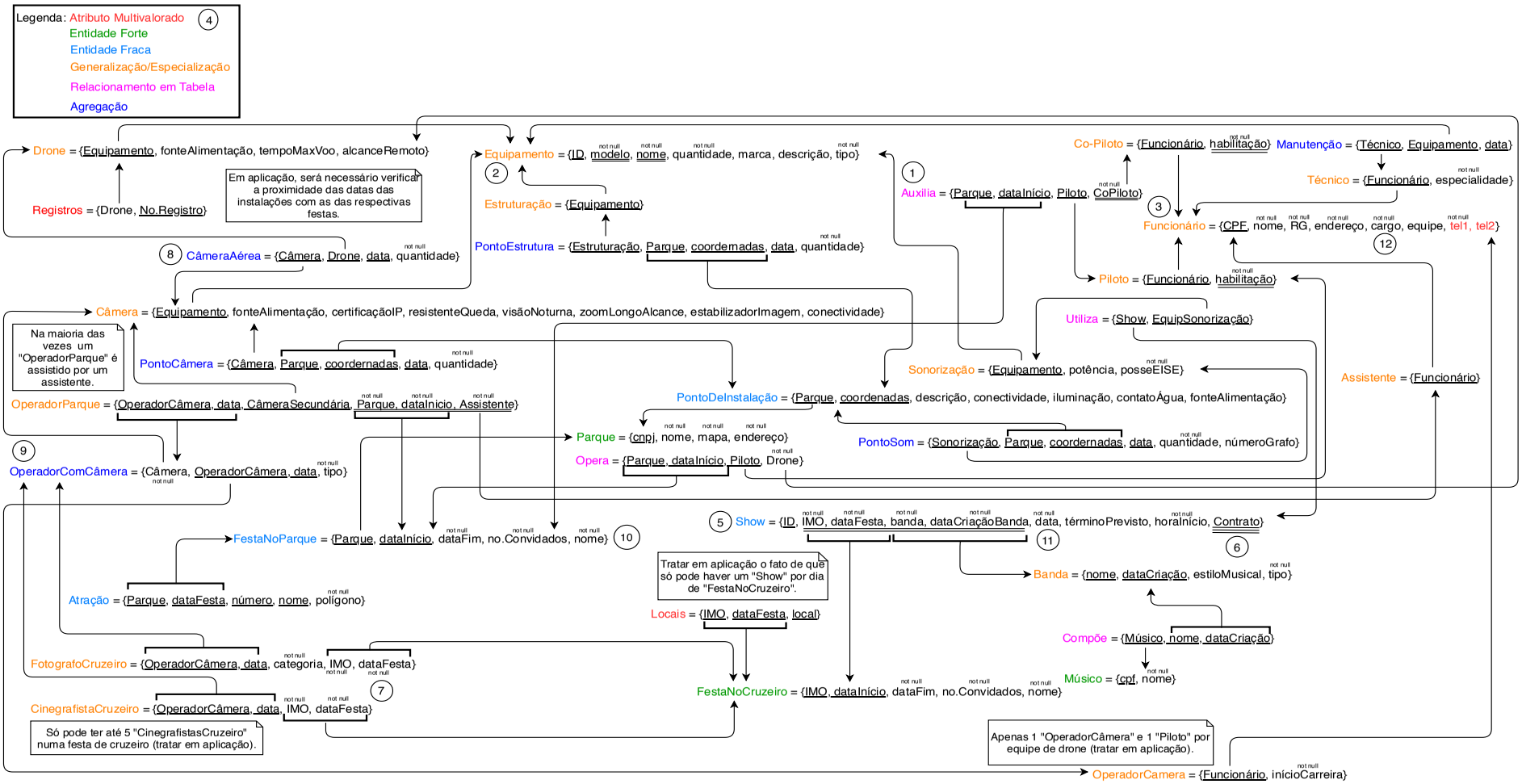
Segue o Diagrama Entidade-Relacionamento elaborado a partir do problema descrito na seção anterior.


Azul - Só tem no Cruzeiro
Vermelho - Cruzeiro e Parque
Amarelo - Só tem no Parque
Making Of
Realiza
Cinegrafista
nomeFesta
nomeCinegrafista
Possui
Nota adicional: só pode ter até 5 cinegrafistas numa festa de cruzeiro.
Retrata
Álbum
Realiza
Fotógrafo
categoria
nomeFesta
nomeFotografo
categoriaFotografo
D
cargo
Técnico
especialidade
função
Funcionário
nome
cpf
rg
telefone
endereço
função
equipe
habilitação
Assistente
Co-Piloto
habilitação
Auxilia
Piloto
Um 'Operador-Parque' é identificado por 'data', 'CPF', e 'Câmera'.
Festa no Cruzeiro
dataInício
dataFim
no.Convidados
nome
IMO
local
Operador-Cruzeiro
cargo
Operador-Parque
Registra
Festa no Parque
Opera
O número de um contrato é único.
tipo
OperadorComCâmera
Utiliza
Operador de Câmera
início Carreira
dataInício
dataFim
no.Convidados
nome
endereço
Possui
Atração
número
nome
polígono
Manutenção
Manutenção
data
data
tipo
Contrato
número
Realiza
Contratada
Acontece
Nota adicional: apenas 1 operador de câmera e 1 piloto por equipe de drone.
Em aplicação, será necessário verificar a proximidade das datas das instalações com as das respectivas festas.
fonteAlimentação
certificaçãoIP
resistenteQueda
visãoNoturna
zoomLongoAlcance
estabilizadorImagem
conectividade
Um 'Operador de Câmera' e a 'data' identificam um 'OperadorComCâmera'.
quantidade
data
CâmeraAérea
Equipa
Câmera
Drone
nome
dataCriação
estiloMusical
tipo
Banda
Compõe
Músico
nome
cpf
Particular
Realiza
Equipamento
modelo
nome
quantidade
marca
descrição
tipo
PontoCâmera
Fixa
data
quantidade
cnpj
nome
mapa
endereço
Parque
Possui
registro
fonteAlimentação
tempoMaxVoo
alcanceRemoto
Nota adicional: atributo "registro" de "Drone" são únicos.
data
quantidade
PontoEstrutura
Estruturação
Possui
Ponto de Instalação
descrição
coordenadas
conectividade
iluminação
contatoÁgua
fonteAlimentação
términoPrevisto
horaInício
data
Show
Utiliza
Sonorização
potência
posseEISE
númeroGrafo
quantidade
data
Contém
PontoSom

3 Modelo Relacional (MR)

3.1 Representação do Modelo Relacional

Segue o Modelo Relacional elaborado a partir do Diagrama Entidade-Relacionamento apresentado na seção anterior.


Legenda: Atributo Multivalorado
Entidade Forte
Entidade Fraca
Generalização/Especialização
Relacionamento em Tabela
Agregação
4
Drone = {Equipamento, fonteAlimentação, tempoMaxVoo, alcanceRemoto}
Registros = {Drone, No.Registro}
Em aplicação, será necessário verificar a proximidade das datas das instalações com as das respectivas festas.
Equipamento = {ID, modelo, nome, quantidade, marca, descrição, tipo}
2
Estruturação = {Equipamento}
PontoEstrutura = {Estruturação, Parque, coordernadas, data, quantidade}
1
Auxilia = {Parque, dataInício, Piloto, CoPiloto}
Co-Piloto = {Funcionário, habilitação}
Manutenção = {Técnico, Equipamento, data}
Técnico = {Funcionário, especialidade}
3
Funcionário = {CPF, nome, RG, endereço, cargo, equipe, tel1, tel2}
12
8
CâmeraAérea = {Câmera, Drone, data, quantidade}
Piloto = {Funcionário, habilitação}
Câmera = {Equipamento, fonteAlimentação, certificaçãoIP, resisteteQueda, visãoNoturna, zoomLongoAlcance, estabilizadorImagem, conectividade}
Utiliza = {Show, EquipSonorização}
Na maioria das vezes um "OperadorParque" é assistido por um assistente.
PontoCâmera = {Câmera, Parque, coordernadas, data, quantidade}
Sonorização = {Equipamento, potência, posseEISE}
Assistente = {Funcionário}
OperadorParque = {OperadorCâmera, data, CâmeraSecundária, Parque, dataInicio, Assistente}
PontoDeInstalação = {Parque, coordenadas, descrição, conectividade, iluminação, contatoÁgua, fonteAlimentação}
Parque = {cnpj, nome, mapa, endereço}
PontoSom = {Sonorização, Parque, coordernadas, data, quantidade, númeroGrafo}
Opera = {Parque, dataInício, Piloto, Drone}
9
OperadorComCâmera = {Câmera, OperadorCâmera, data, tipo}
5
Show = {ID, IMO, dataFesta, banda, dataCriaçãoBanda, data, términoPrevisto, horaInício, Contrato}
11
6
FestaNoParque = {Parque, dataInício, dataFim, no.Convidados, nome}
10
Tratar em aplicação o fato de que só pode haver um "Show" por dia de "FestaNoCruzeiro".
Banda = {nome, dataCriação, estiloMusical, tipo}
Atração = {Parque, dataFesta, número, nome, polígono}
Locais = {IMO, dataFesta, local}
Compõe = {Músico, nome, dataCriação}
Músico = {cpf, nome}
FotografoCruzeiro = {OperadorCâmera, data, categoria, IMO, dataFesta}
7
FestaNoCruzeiro = {IMO, dataInício, dataFim, no.Convidados, nome}
CinegrafistaCruzeiro = {OperadorCâmera, data, IMO, dataFesta}
Só pode ter até 5 "CinegrafistasCruzeiro" numa festa de cruzeiro (tratar em aplicação).
Apenas 1 "OperadorCâmera" e 1 "Piloto" por equipe de drone (tratar em aplicação).
OperadorCamera = {Funcionário, inícioCarreira}

3.2 Justificativas de modelagem

1. Era possível embarcar as relações "Piloto" e "Co-Piloto" na relação "FestaNoParque". Porém, como a proporção de festas no parque que são registradas por drones não é alta, isso geraria uma grande quantidade de valores nulos na relação "FestaNoParque", nos campos "Piloto" e "Co-Piloto". Portanto, para reduzir custo espacial e estando dispostos a perder no desempenho, optou-se por manter a relação "Auxilia" separada.

2. A relação "Equipamento" originalmente possuía chave primária muito grande, composta pelo seu modelo e seu nome, e além disso era um *hub*[1]. Portanto, haveria muita replicação de valores correspondentes a suas chaves primárias e sobrecarregaria a estrutura de índices correspondente. Então, para reduzir o custo espacial, a criação de um ID.

3. Optou-se por criar uma tabela para cada tipo de funcionário, pois os conjuntos de entidades específicas (CEEs) participam de muitas relações, ao passo que o conjunto de entidade generalizada (CEG) não participa de qualquer relação diretamente. Poderíamos ter colocado todos os atributos dos CEEs na tabela do CEG, poupando espaço e tornando as consultas mais rápidas, mas assim todas as inserções nas relações em que os CEEs participam deveriam ser acompanhadas pela checagem de se aquele Funcionário (que é o CEG) tem de fato o cargo correto para participar da relação em questão, o que só poderia ser tratado em aplicação. Da maneira que foi mapeada, essa checagem é feita pelo próprio banco, pois as chaves estrangeiras das relações apontam diretamente para a tabela do funcionário que possui o cargo correto.

4. Para os atributos multivalorados "Locais" da entidade "Festa no Cruzeiro" e "Registros" da entidade "Drone", foi utilizado a modelagem em tabelas separadas, dado que é desconhecido a quantidade máxima de elementos por instância e, portanto, é muito complicado estimar o tamanho máximo que cada atributo poderia assumir caso fosse mapeado dentro da relação que o contém ("FestaNoCruzeiro" e "Drone", respectivamente). O atributo multivalorado "telefone" da entidade "Funcionário" foi mapeado na própria relação "Funcionário", através de duas colunas "tel1" e "tel2", pois foi suposto que a quantidade de números de telefone por funcionário é bastante parecida entre eles (entre 1 e 2) e todo funcionário deve ter ao menos um número de telefone registrado.

5. Optou-se por estabelecer um ID para o conjunto de entidades "Show" pois sua chave é muito grande, o que geraria grande desperdício de espaço na tabela "Utiliza", relativa à relação

[1] Em teoria dos grafos, *hub* é um nó com muitas arestas incidentes.

Show→Utiliza→Sonorização, principalmente por ser uma relação N-N. Dessa forma, ganhamos espaço; no entanto, a presença de somente o ID na tabela Utiliza nos impede de recuperar facilmente informações sobre os shows que utilizaram um dado equipamento de sonorização, a qual passa a requerer uma operação de união com a tabela Show. Porém, essa consulta não parece relevante e provavelmente seria pouco executada, então deixamos o ID como chave de Show. Além disso, este mapeamento já resolve automaticamente o mapeamento do relacionamento "Realiza" entre as entidades "(Banda) Particular", "Banda" e "Festa no Cruzeiro".

6. O "Contrato", apesar de ser uma chave terciária, pode assumir valores nulos, dado que apenas as "Bandas" do tipo "Contratada" possuem um número de contrato associadas a si. Bandas do tipo "Particular" terão valor nulo neste campo, o que não é um problema de custo espacial, pois supõe-se que os shows de bandas particulares serão bastante raros, dado que são usadas unicamente para realizar *covers* para cobrir possíveis problemas com as bandas contratadas.

7. Pelo fato de ter vários cinegrafistas em cada festa, preferiu-se mapear o relacionamento 1:N com *making-of* diretamente na tabela CinegrafistaCruzeiro, sendo o *making-of* representado pelas chaves de FestaNoCruzeiro já que está é uma entidade fraca, escolhemos mapear *making-of* desta forma pois uma tabela a mais para isso resultaria apenas em informação redundante e sem nada a acrescentar, já que está entidade não tem atributo próprio. Assim, ganhamos em eficiência ao realizar uma junção da tabela CinegrafistaCruzeiro com a tabela FestaNoCruzeiro por um lado e perdemos a semântica do *making-of* na mesma por outro. Porém, ressaltamos que essa perda não seria um problema pois saber as informações sobre o *making-of* realizado por um cinegrafista é uma consulta pouco frequente, sendo mais frequente a consulta para verificar se em uma determinada data o cinegrafista já está trabalhando em uma festa ou não. Em relação aos atributos derivados, podemos justificar o seu não mapeamento pois estas informações são facilmente obtidas através de views ou consultas que podem representar tais dados. A mesma justificativa vale para o relacionamento de Álbum com FotografoCruzeiro e FestaNoCruzeiro.

8. Para os diferentes tipos de equipamento e também "Ponto de Instalação", os atributos informativos (*e.g.* conectividade, fonte de alimentação) são muito importantes, mas optou-se por não colocá-los como *Not Null* pois parece inconveniente forçar a determinação desses valores no momento da adição de um novo equipamento no banco. Em outras palavras, parece mais razoável permitir que essas informações sejam adicionadas incrementalmente, após a inserção do equipamento no banco.

9. No MER, foi especificado que a “Câmera” não identifica a agregação “OperadorComCâmera”. Neste caso, a relação “OperadorComCâmera”, no MR, é identificada unicamente pela composição das chaves “CPF” (da relação “OperadorCâmera”) e “data”. A entidade “Operador-Parque”, entretanto, deve ser identificada pelas chaves anteriormente citadas e também pela “Câmera”, dado que os mesmos podem trabalhar com diferentes modelos de câmera em uma única festa.

10. Não é possível mapear o atributo derivado “endereço” da entidade “Festa no Parque” na relação “FestaNoParque” sem a perda de consistência ou tratamento em aplicação, dado que tal atributo não é chave e, portanto, poderia exibir valores diferentes entre as relações “Parque” e “FestaNoParque”. Isso gera uma pequena queda de desempenho, dado que o atributo “endereço” é provavelmente relevante em toda consulta de “FestaNoParque”, sendo agora necessário a junção desta informação que agora está presente apenas na relação “Parque”. Entretanto, prefere-se o ganho de consistência neste caso.

11. Apesar do atributo ’data’ da entidade Show aparecer como chave no MER, ela não é chave no MR pois, como especificado no texto, deseja-se que em uma festa no cruzeiro cada dia tenha um show diferente. (festa, banda) é uma chave que garante essa restrição desejada.

12. Um uso frequente do banco de dados será alocar pilotos em equipes. No entanto, somente pode haver um piloto em cada equipe, o que torna necessário realizar uma consulta para verificar se uma dada equipe já não tem um piloto nela. Se a tabela funcionário não possuir o atributo de critério, a consulta mencionada requereria uma operação de *join* entre as tabelas “Funcionário” e “Piloto”, ou seja, algo como

    ```
    SELECT * FROM piloto NATURAL JOIN funcionario
                WHERE equipe = ‘drone1’;
    ```

    e se retornar vazio significa que pode-se inserir um piloto na equipe 1. No entanto, se o atributo de critério for mantido, pode-se recuperar a informação desejada sem realizar um *join*, tal como

    ```
    SELECT * FROM funcionario WHERE cargo = ‘piloto’
                AND equipe = ‘drone1’;
    ```

    e, novamente, se retornar vazio pode-se inserir um piloto na equipe ‘drone1’. Optou-se por deixar o atributo de critério na tabela funcionario para ganhar em eficiência, mas a consistência do banco de dados deverá ser garantida no nível da aplicação. De forma análoga, esse mesmo tipo de consulta será necessária também aos co-pilotos, operador de câmera no parque e assistente.

4 Implementação do Sistema

4.1 Tecnologias Utilizadas

- O ambiente de desenvolvimento majoritariamente utilizado foi o Ubuntu 18.04 LTS (Bionic Beaver).
  - Esse ambiente pode ser encontrado em `https://github.com/brunoxd/devMachine` se desejar utilizar.
- Como SGBD relacional o grupo optou pelo PostgreSQL em sua versão 10.4.
- A linguagem de programação escolhida pelo grupo para desenvolvimento da aplicação foi C++, sendo o seu compilador g++ em sua versão 7.3.0.
- A interface gráfica da aplicação foi desenvolvida utilizando o framework multiplataforma Qt em sua versão 5.

4.2 Dependências da Aplicação

Por ter sido desenvolvido pensando em ambiente Linux, nossa aplicação possui como dependências de execução as seguintes bibliotecas:

- postgresql
- postgresql-contrib
- g++
- qt5-default
- libqt5sql5-psql

Para ambientes de execução Linux baseados em Debian e devidamente atualizados, nosso script de configuração de ambiente para aplicação ($config.sh$) procura tentar instalar essas dependências caso não se encontrem já instaladas no sistema.

4.3 Execução do Programa

Junto aos arquivos de código fonte da aplicação se encontra um shell script “config.sh” responsável por automatizar todo o processo de instalação de dependências, criação e alimentação da base de dados e compilação da aplicação para execução, basta apenas executá-lo dentro da pasta do projeto com o comando:

```
$ ./config.sh
```

Porém, caso deseje configurar manualmente o ambiente para execução da aplicação, segue aqui todos os passos necessários:

```
# Destrói a base de dados com o nome do usuário local.
$ sudo su -- postgres -c "psql -c \"DROP DATABASE $(whoami);\""
# Cria uma base de dados limpa com o nome do usuário local.
$ sudo su -- postgres -c "psql -c \"CREATE DATABASE $(whoami);\""
# Insere na base de dados a estrutura do banco.
$ psql $(whoami) < sqlFiles/schemas.sql
# Alimenta a base de dados com as inserções previamente geradas.
$ psql $(whoami) < sqlFiles/inserts.sql
# Prepara o código para compilação.
$ qmake -makefile projeto_bd.pro
# Compila o programa silenciosamente.
$ make -s
# Executa a aplicação.
$ ./projeto_bd
```

Caso utilize o script “config.sh” e não tenha sucesso, um log dos erros ocorridos pode ser encontrado no diretório “logs” do projeto.

4.4 Alimentação da Base de Dados

Dada a complexidade de nossa base de dados, optamos por gerar o script de inserções automaticamente, isso só foi possível graças a um programa feito em Python de autoria própria do grupo. A aplicação em questão foi feita inicialmente pensada para atender o problema das inserções no nosso projeto, porém ela acabou se tornando genérica para qualquer esquema de criação da base de dados. Algumas adaptações foram necessárias fora dessa ferramenta, a mais impactante foi a de disjunção que foi garantida por meio de querys diretamente no banco de dados, posteriormente foi feito o dump da base de dados com as disjunções corrigidas e assim gerou-se o “inserts.sql”.

Por padrão, a configuração do banco por meio do script “config.sh” apenas insere no banco o script de inserção “inserts.sql” previamente gerado pelo grupo. Porém, caso deseje gerar um novo

script de inserções, execute o seguinte comando:

```
$ ./scripts/developer/config.sh NUM_INSERTS
```

Em que NUM_INSERTS corresponde ao número de inserções por tabela que desejar, caso não insira esse parâmetro, o padrão considerado pelo programa é de 20 inserções por tabela.

Caso se interesse em saber mais a respeito desta ferramenta, ela pode ser encontrada em: `https://github.com/FelSiq/bllsht-my-database`

4.5 Trechos do Código Fonte

Todas as funções mencionadas aqui podem ser encontradas em "/src/database/eisedatabase.cpp"

Figura 1: Consulta que busca todas as câmeras disponíveis na base de dados em uma determinada data. A disponibilidade da câmera é verificada se a quantidade de vezes que a câmera foi utilizada em uma determinada data é menor do que a quantidade total de modelos.

```
StringPairVectorList EISEDatabase::getCamerasData(QString partyStartDate)
{

 const static QString query = QString("SELECT E.*, CASE WHEN D.QTD_UTILIZADA IS NULL THEN '0' "
      "ELSE D.QTD_UTILIZADA END AS QTD_UTILIZADA, CAM.* FROM EQUIPAMENTO E LEFT JOIN( "
    "SELECT C.CAMERA, SUM(C.QTD) AS QTD_UTILIZADA "
        "FROM( SELECT CAMERA, DATA, COUNT(*) AS QTD FROM OPCOMCAMERA "
                "WHERE DATA BETWEEN TO_DATE('%1', 'YYYY-MM-DD') AND TO_DATE('%1', 'YYYY-MM-DD') "
                "GROUP BY(CAMERA, DATA) "
            "UNION ALL SELECT IDCAMERASECUNDARIA AS CAMERA, DATA, COUNT(*) AS QTD "
                "FROM OPPARQUE WHERE DATA BETWEEN TO_DATE('%1', 'YYYY-MM-DD') AND TO_DATE('%1', 'YYYY-MM-DD') "
                "GROUP BY(CAMERA, DATA) "
            "UNION ALL "
            "SELECT IDCAMERA AS CAMERA, DATA, SUM(QUANTIDADE) AS QTD "
                "FROM PONTOCAMERA "
                "WHERE DATA BETWEEN TO_DATE('%1', 'YYYY-MM-DD') AND TO_DATE('%1', 'YYYY-MM-DD') "
                "GROUP BY(CAMERA, DATA) "
            "UNION ALL "
            "SELECT CAMERA, DATA, SUM(QUANTIDADE) AS QTD "
                "FROM CAMERAAEREA WHERE DATA BETWEEN TO_DATE('%1', 'YYYY-MM-DD') AND TO_DATE('%1', 'YYYY-MM-DD') "
                "GROUP BY(CAMERA, DATA) "
        ") AS C GROUP BY(C.CAMERA) "
    ") AS D ON E.ID = D.CAMERA JOIN CAMERA CAM ON E.ID = CAM.ID WHERE(D.QTD_UTILIZADA "
    "< E.QUANTIDADE OR D.QTD_UTILIZADA IS NULL) AND TIPO = 'CAMERA';").arg(partyStartDate);

    QSqlQuery rows = m_database.exec(query);
    StringPairVectorList items;

    while(rows.next()){
        QVector<QPair<QString,QString> > vec;

        vec.append( {"Id", rows.value(0).toString()} );
        vec.append( {"Modelo", rows.value(1).toString()} );
        vec.append( {"Nome", rows.value(2).toString()} );
        vec.append( {"Qtd", rows.value(3).toString()} );
        vec.append( {"Marca", rows.value(4).toString()} );
        vec.append( {"Qtd utlizada", rows.value(7).toString()} );

        items.append(vec);
    }

    return items;
}
```

Figura 2: Apresenta apenas os operadores de câmera disponíveis em uma determinada data para uma festa, para isso é necessário buscar quais operadores de câmera já estão trabalhando em uma determinada festa, essa informação é obtida por meio da correspondência entre as datas.

```
311 StringPairVectorList EISEDatabase::getEmployeesData(QString partyStartDate, QString partyEndDate)
312 {
313
314     const static QString query = QString("SELECT OP.CPF, F.NOME, OP.INICIOCARREIRA, F.TEL1 FROM OPCAMERA OP JOIN ( "
315     "SELECT CPFOPCAMERA AS CPF FROM FOTOGRAFOCRUZEIRO JOIN FESTANOCRUZEIRO ON IMOFESTA = IMO AND DATAFESTA = DATAINICIO "
316     "WHERE TO_DATE('%1', 'YYYY-MM-DD') < DATAINICIO OR TO_DATE('%2', 'YYYY-MM-DD') > DATAFIM "
317     "UNION SELECT CPFOPCAMERA AS CPF FROM CINEGRAFISTACRUZEIRO JOIN FESTANOCRUZEIRO ON IMOFESTA = IMO AND DATAFESTA = DATAINICIO
318     "WHERE TO_DATE('%1', 'YYYY-MM-DD') < DATAINICIO "
319     "OR TO_DATE('%2', 'YYYY-MM-DD') > DATAFIM UNION SELECT OP.CPFOPCAMERA AS CPF "
320     "FROM OPPARQUE OP JOIN FESTANOPARQUE FP ON OP.CNPJPARQUE = FP.CNPJPARQUE AND OP.DATAINICIOPARQUE = FP.DATAINICIO "
321     "WHERE TO_DATE('%1', 'YYYY-MM-DD') < FP.DATAINICIO OR TO_DATE('%2', 'YYYY-MM-DD') > FP.DATAFIM "
322     ") AS DT ON OP.CPF IN (DT.CPF) JOIN FUNCIONARIO F ON OP.CPF = F.CPF WHERE F.CARGO = 'OPCAMERA';").arg(partyStartDate,
323     partyEndDate);
324
325     const static QString query2 = "SELECT * FROM funcionario;";
326
327     QSqlQuery rows = m_database.exec(query);
328     StringPairVectorList items;
329
330     while(rows.next()){
331         QVector<QPair<QString,QString> > vec;
332
333         vec.append( {"CPF", rows.value(0).toString()} );
334         vec.append( {"Nome", rows.value(1).toString()} );
335         vec.append( {"Inicio Carreira", rows.value(2).toString()} );
336         vec.append( {"Telefone", rows.value(3).toString()} );
337
338         items.append(vec);
339     }
340
341     return items;
342 }
```

Figura 3: Dado um músico de nome M, quero saber as banda onde ele já trabalhou, seu cpf, o contrato dos shows que essas bandas já realizaram, e o nome da festa no cruzeiro associada. Se o músico está cadastrado, mas não realizou nenhum show, isso também deve ser mostrado.

```
21  ReportTextData EISEDatabase::getReport1(){
22      /*
23      Dado um músico de nome M, quero saber as banda onde ele já trabalhou,
24      seu cpf, o contrato dos shows que essas bandas já realizaram,
25      e o nome da festa no cruzeiro associada.
26      Se o músico está cadastrado, mas não realizou nenhum show,
27      isso deve também ser mostrado.
28      */
29
30      const static QString query =
31      "SELECT M.nome AS nome_musico,\n\
32          M.cpf AS cpf_musico,\n\
33          C.nomeBanda AS nome_banda,\n\
34          S.contrato AS show_contrato,\n\
35          FC.nome AS festa_nome\n\
36      FROM musico M\n\
37      LEFT JOIN compoe C\n\
38          ON M.cpf = C.cpfmusico\n\
39      LEFT JOIN show S\n\
40          ON (S.nomebanda, S.dataCriacaoBanda) = (C.nomeBanda, C.dataCriacaoBanda)\n\
41      LEFT JOIN festaNoCruzeiro FC\n\
42      ON (S.imo, S.datafesta) = (FC.imo, FC.datainicio)\n\
43      ORDER BY M.nome;";
44
45      QSqlQuery rows = m_database.exec(query);
46      StringPairVectorList items;
47      while(rows.next()){
48          QVector<QPair<QString,QString> > vec;
49
50          vec.append( {"Nome do Músico", rows.value(0).toString()} );
51          vec.append( {"CPF do Músico", rows.value(1).toString()} );
52          vec.append( {"Nome da Banda", rows.value(2).toString()} );
53          vec.append( {"Contrato do Show", rows.value(3).toString()} );
54          vec.append( {"Nome da Festa", rows.value(4).toString()} );
55
56          items.append(vec);
57      }
58
```

4.6 Entrega

Entrega realizada no Tidia-Ae 4.0 por: Bruno Mendes da Costa

5 Considerações Finais

No começo deste projeto tivemos muita dificuldade em entender o que realmente estava sendo proposto, a descrição do projeto não deixou muito explícito que somente eventos do tipo festa seriam aceitos, nesse meio tempo acabamos tendo que recomeçar o projeto pois havíamos escolhido outro tipo de evento.

Outro ponto que gerou mais dúvidas foi a exigência de que o projeto atendesse a 2 tipos de festas distintos. Refizemos pensando em atender esse requisito, mas ao levar para revisão, após explicação da professora, entendemos que o que estava sendo solicitado não era bem isso, que na verdade o projeto deveria ter um conjunto de entidades que pertencessem a um determinado tipo de festa e outro conjunto que não pertencesse, além disso, deveria haver uma interseção entre esse conjuntos de entidades que ligassem o projeto como um todo.

No geral, para o nosso grupo, iniciar o projeto foi bem complicado, mas passando essa fase inicial conseguimos nos sair bem, até a parte 3.

Desenvolver uma aplicação não é fácil, ainda mais em um semestre que já cobra grandes projetos, o tempo se torna curto, tivemos que tirar um tempo para aprender a mexer com a interface gráfica, a qual não tínhamos experiência prévia e nem com aplicações que utilizassem banco de dados, assim, acabamos não conseguindo implementar apropriadamente tudo o que prometemos, mas a parte do banco em si foi bem implementada, amarramos tudo o que foi possível no banco para tentar deixar a aplicação mais simples, e mesmo assim, a complexidade de desenvolver uma aplicação desse porte se mostrou acima do esperado.

Este projeto como um todo foi de grande aprendizado, a elaboração dele agregou muito mais conhecimento do que estudar para uma prova ou resolver algum exercício de sala, acreditamos que essa experiência seja fundamental para a nossa formação, não só como aluno mas como profissional.