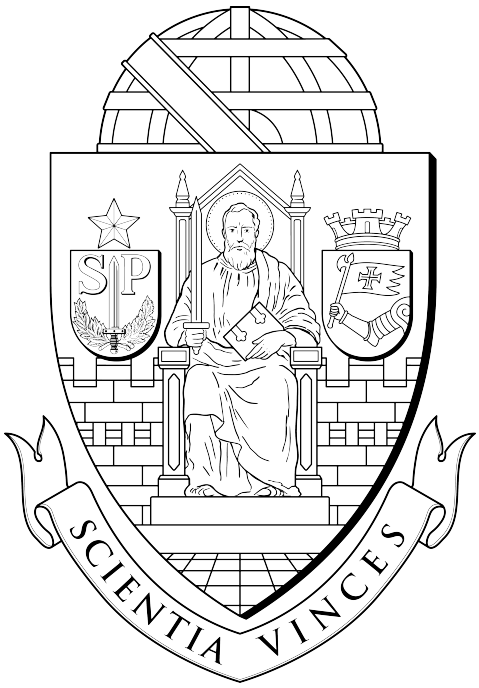

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
SCC0240 - Bases de Dados

Projeto - Produtora

Sistema de Gerenciamento de um Telejornal

Discentes	Fabio Fogarin Destro	10284667
	Paulo André de Oliveira Carneiro	10295304
	Renata Vinhaga dos Anjos	10295263
	Vitor Henrique Gratiere Torres	10284952

Docente Elaine Parros Machado de Sousa

São Carlos, 14 de Junho de 2019

Parte I

1 Descrição do Problema

Uma produtora de mídia foi contratada para gerenciar a produção de um programa televisivo com foco em notícias frias (sem urgência), como por exemplo o programa Fantástico da Globo. A produção deve oferecer a organização de recursos para a gravação de uma matéria, bem como os profissionais envolvidos, locais de gravação, equipamentos e edição.

Esse programa de TV conta com um acervo de pautas (resumos de possíveis notícias), cada uma contendo um título único, a data de inclusão, um resumo e links de referência. O acervo é alimentado pela equipe de pesquisa. O acervo pode ser acessado pelos jornalistas, que são encarregados para trabalhar em uma notícia, realizando o trabalho de campo, levantando informações e incluindo o texto de uma matéria produzida. Essa matéria passa pelo processo de revisão de um produtor responsável. O produtor pode aprovar o roteiro ou adicionar um ou mais comentários de revisão para o jornalista.

Com a matéria aprovada, iniciam-se as gravações para a matéria. Cada matéria pode conter diversas gravações em dias e horários distintos. Além disso, o produtor pode entrar em contato com uma ou mais pessoas que serão entrevistadas em uma gravação do conteúdo, que conta com um ou mais repórteres. Cada repórter é definido por CPF, nome e telefone. Um entrevistado pode ser apenas uma testemunha ou um especialista necessário para embasamento da matéria.

Para cada gravação, são alocados profissionais de audiovisual e equipamentos necessários. Os tipos de equipamentos utilizados são: câmeras, microfones e iluminação. Cada equipamento possui uma identificação com tipo, marca, ano e número de patrimônio. Um equipamento fica armazenado em uma sala da produtora, identificada pela composição do seu número, andar e bloco.

O produtor pode definir também o local para cada gravação da notícia, podendo ser uma gravação em um cenário ou em um endereço externo. Um cenário é localizado pelo número da sala, bloco e andar. Uma locação externa possui um endereço com logradouro, número, CEP, cidade e estado.

Cada gravação gera arquivos de vídeo que são então entregues para um editor que irá polir e juntar o material para montar o material final que irá ao ar. A edição ocorre em uma sala de edição identificada pela composição do seu número, andar e bloco. O editor deve reservar a sala em um dia e período. Ao fim do processo de edição, é gerado um material final, que contém as gravações de uma matéria e foi editado por um editor.

2 Consultas Relevantes

- Quais pessoas trabalharam ou participaram de uma matéria que foi gravada;
- Qual o cenário mais utilizado nas gravações;
- Qual o tipo de equipamento com menor disponibilidade (número de equipamentos totais de um tipo - número de equipamentos reservados desse tipo);
- Qual a porcentagem de pautas catalogadas que viraram gravação;
- Quantos roteiros cada jornalista escreveu por mês;
- Qual a porcentagem de participação de matérias gravadas de um jornalista específico.

3 Projeto Conceitual

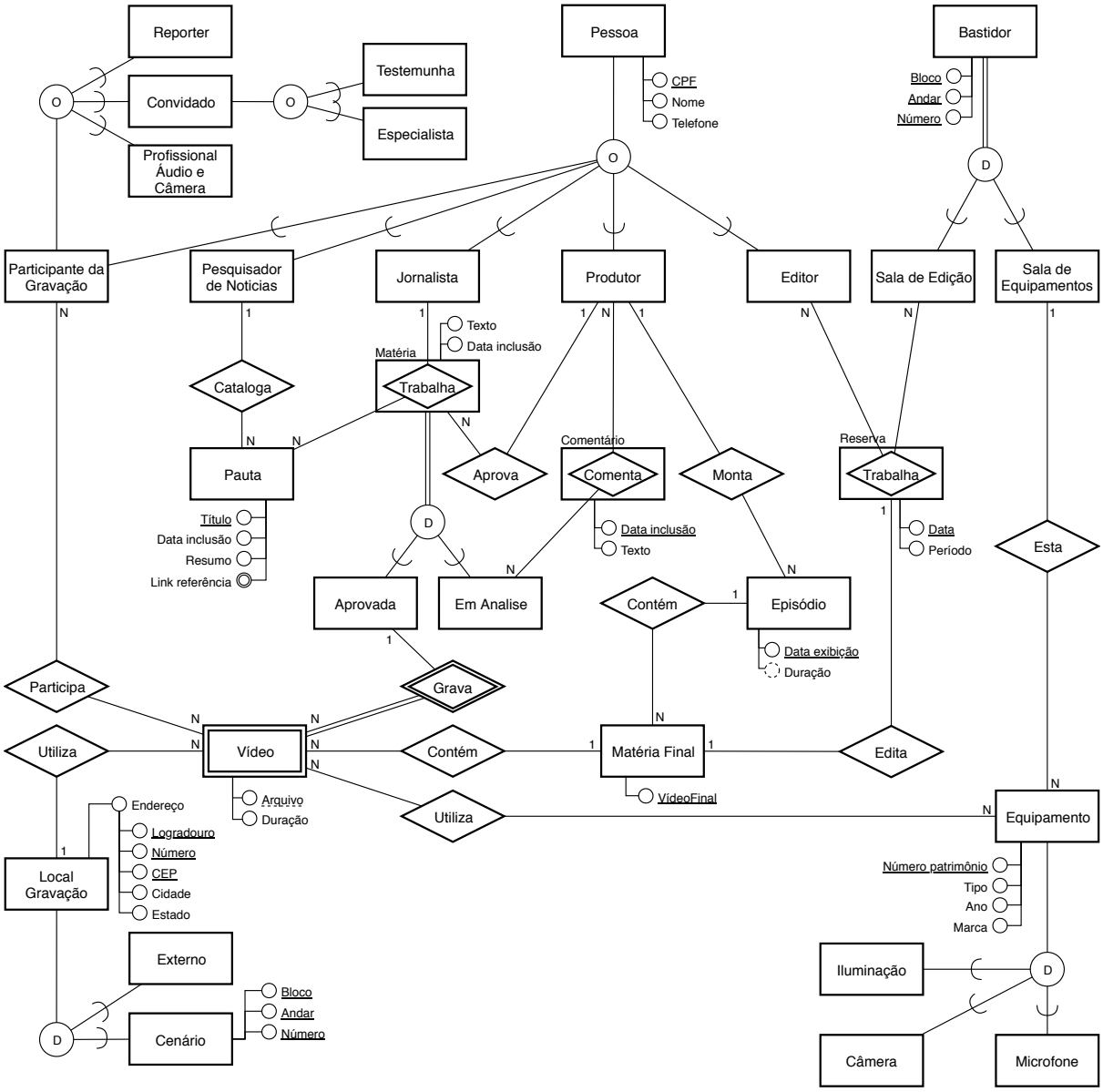

Figura 1: Diagrama Entidade-Relacionamento

Parte II

1 Projeto Lógico

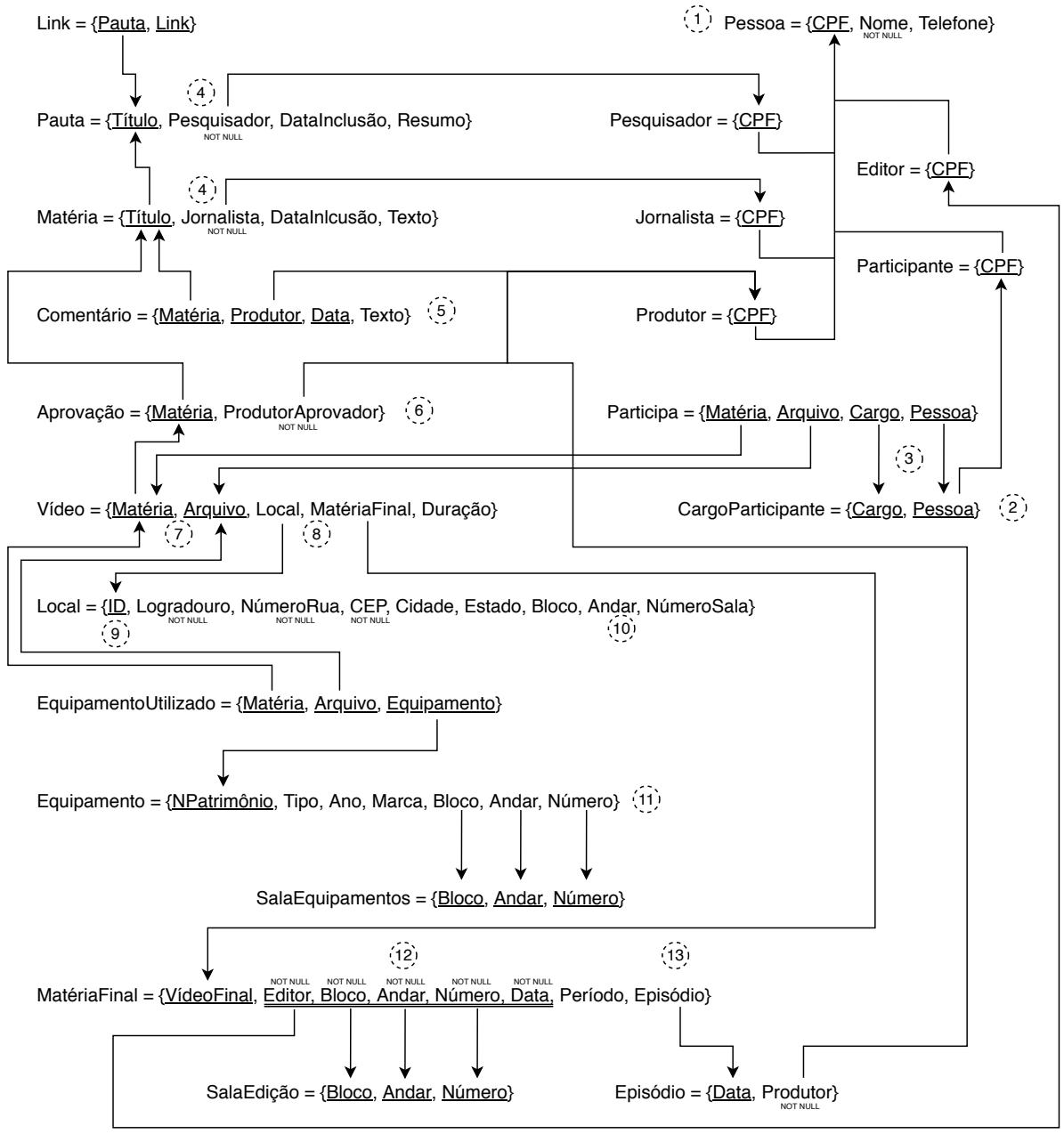

Figura 2: Diagrama Relacional

2 Considerações

1. Para a entidade Pessoa, mapeamos uma entidade geral, e para suas heranças mapeamos apenas as que têm relações para garantir restrições de relacionamentos com cargos específico, decidimos por esta opção para centralizar as informações e poder dar vários cargos para a mesma pessoa;

2. Para discriminar os participantes criamos a relação CargoParticipante, note que isso também mantém o overlap da sub-herança que descrevemos no MER, porém aqui o cargo não está mapeado em uma relação e sim no atributo porque esse cargo não gera restrições em relações, é apenas informativo. Fizemos isso também para manter os registros de Repórteres, por exemplo, que ainda não participaram de uma gravação;

3. A relação Participa poderia receber a chave estrangeira da relação Participante, porém decidimos pegar as chaves de CargoParticipante para quando montarmos os créditos conseguirmos saber qual papel a pessoa representou na gravação de uma matéria, note que a mesma pessoa pode representar papéis diferentes na mesma gravação;

4. Para manter as cardinalidades de uma Pauta é Catalogada por um Pesquisador e um Matéria é Trabalhada por um Jornalista atrelamos a cada Pauta e Matéria seu respectivo Pesquisador e Jornalista, ambos não nulos. Assim também garantimos a volta um Pesquisador escreve n Pautas e um Jornalista Trabalha em n Matérias;

5. Mapeamos a agregação Comentário com as chaves Matéria, Produtor e Data, assim o mesmo produtor pode fazer n comentários na mesma matéria;

6. Criamos a relação Aprovação para que seja possível recuperar o Produtor que aprovou que determinada matéria esteja pronta para gravação, e só de uma matéria aprovada podem ser gravados vídeos;

7. No MER, definimos "Arquivo" como chave parcial da entidade Vídeo, no Relacional, a relação Vídeo tem as chaves Matéria (herdada por ser entidade fraca) e Arquivo (sua chave parcial);

8. Colocamos Local e MatériaFinal na relação Vídeo para garantir as cardinalidades 1:N. Note também que respeita às relações parciais podendo ou não ter um Local e podendo ou não participar de uma MatériaFinal;

9. Primeiramente, pensamos em mapear as entidades Local (geral) e Cenário (específica) e não mapear Externo (específica) por esta não ter atributos específicos, e a relação Cenário herdaria um Local, então Vídeo teria que ter um campo para o Local e um campo para o Cenário, isso implicaria em inconsistências porque um Cenário também teria (por herança) um Local, logo Vídeo teria dois Locais que poderiam ser diferentes. Não queríamos criar entidades como VídeoExterno e VídeoCenário e duplicar todos as relações para Vídeo, por isso definimos que a relação Local teria um ID único. Também teria as informações de endereço e localização do cenário já que Cenário também precisa dessas informações e os últimos três campos (três inteiros) podem ficar nulos quando em Locais Externos sem uma grande perda para o banco;

10. Sabemos também que Logradouro, NúmeroRua, CEP, Bloco, Andar e NúmeroSala definem uma chave secundária (aceitando valores nulos para os últimos três atributos), porém não quisemos carregar o banco com essa verificação, essa inconsistência não é crítica;

11. Só decidimos mapear a entidade genérica Equipamento porque as específicas não têm[1] nem atributos específico, nem relacionamentos;

12. Mapeamos a Reserva da Sala de Edição na MatériaFinal porque uma Reserva só Edita uma única MatériaFinal, aqui precisamos definir a chave secundária porque não podem ocorrer duas Reservas na mesma Sala na mesma Data, nem um Editor Reservar duas Sala na mesma Data. Não conseguimos tratar as coincidências de intervalos de tempo, deixamos isso a cargo da aplicação;

13. Se uma MatériaFinal for selecionada para aparecer em um Episódio ela receberá a data do Episódio para qual ela foi selecionada, caso contrário, não irá ao ar e terá esse atributo nulo.

[1] O Novo Acordo Ortográfico da Língua Portuguesa assinado em 1990 e que entrou em vigor em Janeiro de 2016 manteve o acento diferencial que distingue tempos verbais singular e plural de verbos como, por exemplo, tem/têm e convém/convêm. Referências: bit.ly/31rbhyp, bit.ly/2wROP3h, bit.ly/2Idkzqa e bit.ly/2IdABQA

Parte III

1 Descrição da Implementação

Arquivos códigos-fonte submitados no Escaninho do Tidia do aluno Fabio Fogarin Destro (10284667).

Utilizamos `PostgreSQL` e `node.js`, sendo estes necessários para a execução do sistema. O sistema inteiro foi escrito e testado em `GNU Linux Ubuntu` e `macOS`.

Para executar, dentro da pasta `server/` execute `npm install` para instalar as dependências do projeto, então `npm start` para abrir o servidor. É necesário também configurar variáveis de ambiente de usuário e senha do banco usando os seguintes comandos `export USER=$your username$` e `export PASS=$your password$`.

1.1 DDLs

Criação da relação Materia_Final. Acessível em `server/bdsetup.js`.

```
CREATE TABLE MATERIA_FINAL (
    VIDEO_FINAL VARCHAR(50),
    EDITOR CHAR(11) NOT NULL,
    BLOCO INTEGER NOT NULL,
    ANDAR INTEGER NOT NULL,
    NUMERO INTEGER NOT NULL,
    DATA TIMESTAMP NOT NULL,
    PERIODO INTERVAL NOT NULL,
    EPISODIO DATE,

    CONSTRAINT PK_MATERIA_FINAL
        PRIMARY KEY (VIDEO_FINAL),
    CONSTRAINT FK_MATERIA_FINAL_EDITOR
        FOREIGN KEY (EDITOR)
        REFERENCES EDITOR(CPF),
    CONSTRAINT FK_MATERIA_FINAL_SALA_EDICAO
        FOREIGN KEY (BLOCO, ANDAR, NUMERO)
        REFERENCES SALA_EDICAO (BLOCO, ANDAR, NUMERO),
    CONSTRAINT SK_MATERIA_FINAL
        UNIQUE (EDITOR, BLOCO, ANDAR, NUMERO, DATA),
    CONSTRAINT FK_MATERIA_FINAL_EPISODIO
        FOREIGN KEY (EPISODIO)
        REFERENCES EPISODIO (DATA)
        ON DELETE SET NULL
);
```

Definição da agregação Comentario. Acessível em `server/bdsetup.js`.

```
CREATE TABLE COMENTARIO(
    MATERIA VARCHAR(100),
    PRODUTOR CHAR(11),
    DATA_INCLUSAO TIMESTAMP DEFAULT NOW(),
    TEXTO TEXT,

    CONSTRAINT PK_COMENTARIO
```

```
        PRIMARY KEY (MATERIA, PRODUTOR, DATA_INCLUSAO),
    CONSTRAINT FK_COMENTARIO_PRODUTOR
        FOREIGN KEY (PRODUTOR)
        REFERENCES PRODUTOR(CPF),
    CONSTRAINT FK_COMENTARIO_MATERIA
        FOREIGN KEY(MATERIA)
        REFERENCES MATERIA(TITULO)
        ON DELETE CASCADE
);
```

Alguns exemplos de deleção de tabelas. Acessível em `server/bdsetup.js`.

```
DROP TABLE IF EXISTS COMENTARIO;
DROP TABLE IF EXISTS EQUIPAMENTO_UTILIZADO;
DROP TABLE IF EXISTS PARTICIPA;
DROP TABLE IF EXISTS VIDEO
```

1.2 DMLs

Mostra quantas matérias cada jornalista escreveu por mês. Acessível em `server/routes/materias.js`.

```
SELECT M.JORNALISTA, EXTRACT(YEAR FROM M.DATA_INCLUSAO) AS YEAR,
        EXTRACT(MONTH FROM M.DATA_INCLUSAO) AS MONTH, COUNT(*)
    FROM MATERIA M
    GROUP BY M.JORNALISTA, EXTRACT(YEAR FROM M.DATA_INCLUSAO),
        EXTRACT(MONTH FROM M.DATA_INCLUSAO)
```

Mostra todos as matérias finais que forão ao ar no dia "data". Acessível em `server/routes/materias_finais.js`.

```
SELECT DISTINCT V.MATERIA
    FROM MATERIA_FINAL MF
    JOIN VIDEO V
        ON V.MATERIA_FINAL = MF.VIDEO_FINAL
    WHERE TO_DATE(MF.EPISODIO, 'YYYY-MM-DD') = ${request.params.data}
```

Retorna a porcentagem de pautas que foram efetivamente gravadas. Acessível em `server/routes/-pautas.js`.

```
let SQL =   SELECT COUNT(*)
                FROM VIDEO V
                JOIN PAUTA P
                    ON V.MATERIA = P.TITULO

let results = await conexao.query(SQL);
let qtd_video = results.rows[0].count;

SQL =   SELECT COUNT(*)
            FROM PAUTA

results = await conexao.query(SQL);
let qtd_pautas = results.rows[0].count;
response.status(200).json((qtd_video/qtd_pautas * 100).tofixed(2))
```

Pesquisa quais os cenários mais utilizados. Acessível em `server/routes/local.js`.

```
SELECT L.BLOCO, L.ANDAR, L.NUMERO_SALA, COUNT(*) AS QTDLOCACOES
    FROM VIDEO V
    JOIN LOCAL L
        ON L.ID = V.LOCAL
    WHERE L.BLOCO IS NOT NULL AND L.ANDAR IS NOT NULL
        AND L.NUMERO_SALA IS NOT NULL
    GROUP BY L.BLOCO, L.ANDAR, L.NUMERO_SALA
    ORDER BY QTDLOCACOES
```

Retorna todas as pessoas que trabalharam em uma matéria final ("titulo") desde o pesquisador até o editor. Acessível em `server/routes/materias_finais.js`.

```
SELECT V.MATERIA, PPES.NOME AS PESQUISADOR, PJOR.NOME AS JORNALISTA,
        PPRO.NOME AS PRODUTOR, PPAR.NOME AS PARTICIPANTE

    FROM VIDEO V

    JOIN APROVACAO A
        ON V.MATERIA = A.MATERIA
    JOIN PESSOA PPRO -- PESSOA "PRODUTOR"
        ON PPRO.CPF = A.PRODUTORAPROVADOR

    JOIN MATERIA M
        ON V.MATERIA = M.TITULO
    JOIN PESSOA PJOR -- PESSOA "JORNALISTA"
        ON PJOR.CPF = M.JORNALISTA

    JOIN PAUTA PA
        ON V.MATERIA = PA.TITULO
    JOIN PESSOA PPES -- PESSOA "PESQUISADOR"
        ON PPES.CPF = PA.PESQUISADOR

    LEFT JOIN PARTICIPA P
        ON V.MATERIA = P.MATERIA AND V.ARQUIVO = P.ARQUIVO
    LEFT JOIN PESSOA PPAR -- PESSOA "PARTICIPANTE"
        ON PPAR.CPF = P.PESSOA

    WHERE V.MATERIA = ${request.params.titulo}
```

Retorna todos os funcionarios da Produtora e, se houverem, seus respectivos cargos. Acessível em `server/routes/pessoas.js`.

```
SELECT PES.CPF, PES.NOME, PES.TEL, PESQ.CPF AS PESQUISADOR,
        JOR.CPF AS JORNALISTA, PROD.CPF AS PRODUTOR, ED.CPF AS EDITOR
    FROM PESSOA PES
    LEFT JOIN PESQUISADOR PESQ
        ON PES.CPF = PESQ.CPF
    LEFT JOIN JORNALISTA JOR
        ON PES.CPF = JOR.CPF
    LEFT JOIN PRODUTOR PROD
        ON PES.CPF = PROD.CPF
    LEFT JOIN EDITOR ED
        ON PES.CPF = ED.CPF
```

Insere um novo equipamento na relação, note que `Equipamento` tem o campo `ID` do tipo `SERIAL` o qual não inserimos manualmente por que o SGBD já trata seu incremento se assumido `DEFAULT`. Acessível em `server/routes/equipamentos.js`.

```
INSERT INTO
    EQUIPAMENTO ("TIPO", "ANO", "MARCA", "BLOCO", "ANDAR", "NUMERO")
    VALUES ($1, $2, $S3, $S4, $S5, $S6)
values: [tipo, ano, marca, bloco, andar, numero]
```

Retorna quantas vezes cada tipo de equipamento foi utilizado. Acessível em `server/routes/equipamentos.js`.

```
SELECT E.TIPO, COUNT(*)
    FROM EQUIPAMENTO_UTILIZADO EU
    JOIN EQUIPAMENTO E
        ON E.NPATRIMONIO = EU.EQUIPAMENTO
    GROUP BY E.TIPO
    ORDER BY COUNT(*)
```

2 API

O sistema foi desenvolvido utilizando o conceito de `API RESTful` utilizando os metodos HTTP: `GET` (requisita dados), `POST` (insere dados), `PUT` (atualiza dados) e `DELETE` (deleta dados).

Para fazer requisições ao servidor é utilizado o URL da API como o modelo `localhost:port/-api/route`, as rotas do servidor do sistema estão dividas de acordo com a categoria de dados que serão tratados. Aqui serão descritos a lista de rotas e possíveis requisições:

2.1 /api/pessoas

Descritas no arquivo `servidor/routes/pessoas.js`.

`GET: /`

Parâmetros: Nenhum.

Retorno:
- `200:` Todas as pessoas cadastradas no banco de dados e seus cargos.
- `404:` Nada encontrado.

`GET: /filtros?cargo1=1[&cargoN=1]`

Parâmetros: Filtros por query string.

Retorno:
- `200:` Todas as pessoas cadastradas no banco de dadosque atendem aos filtros aplicados na página HTML.
- `404:` Nada encontrado.

`GET: /quantidade/:cargo`

Parâmetros: Cargo requerido, e.g. Jornalista.

Retorno:
- 200: Número de pessoas cadastradas no banco de dados que atendem ao cargo especificado.
- 404: Nada encontrado.

`POST: /`

Parâmetros: CPF, Nome, Telefone, Cargos, definidos no formulário da página HTML.

Retorno:
- 200: Operação bem sucedida.
- 400: Falha ao inserir dados.

`PUT: /`

Parâmetros: Nome, Telefone, Cargos, definidos no formulário da página HTML.

Retorno:
- 200: Operação bem sucedida.
- 400: Falha ao atualizar dados.

2.2 /api/comentarios

Descritas no arquivo `servidor/routes/comentarios.js`.

`GET: /`

Parâmetros: Nenhum.

Retorno:
- 200: Todos os comentários de todas as matérias.
- 404: Nada encontrado.

`POST: /`

Parâmetros: Matéria, produtor, texto do comentário, inseridos por formulario na página.

Retorno:
- 200: Operação bem sucedida.
- 400: Falha ao inserir comentario.

`DELETE: /`

Parâmetros: Matéria, produtor e timestamp do comentário a ser apagado inseridos por formulário na página.

Retorno:
- 200: Operação bem sucedida.
- 400: Falha ao remover comentario.

2.3 /api/pautas

Descritas no arquivo `servidor/routes/pautas.js`.

`GET: /`

Parâmetros: Nenhum.

Retorno:
- `200:` Todas as pautas.
- `404:` Nada encontrado.

`GET: /pesquisador/:cpf`

Parâmetros: CPF de um pesquisador.

Retorno:
- `200:` Todas as pautas escritas por um dado pesquisador.
- `404:` Nada encontrado.

`GET: /link/:titulo`

Parâmetros: Título de uma pauta.

Retorno:
- `200:` Todas os links de uma dada pauta.
- `404:` Nada encontrado.

`GET: /semMateria`

Parâmetros: Nenhum.

Retorno:
- `200:` Todas as pautas que não estão associadas a uma matéria.
- `404:` Nada encontrado.

`GET: /semMateria/:cpf`

Parâmetros: CPF de um pesquisador.

Retorno:
- `200:` Todas as pautas que não estão associadas a uma matéria de um dado pesquisador.
- `404:` Nada encontrado.

`GET: /porcentagemPautasGravadas`

Parâmetros: Nenhum.

Retorno:
- `200`: Quantidade de pautas que foram gravadas dividido pela quantidade total de pautas catalogadas
- `404`: Nada encontrado.

`POST: /`

Parâmetros: Título, pesquisador e resumo, inseridos por formulário na página.

Retorno:
- `200`: Operação bem sucedida.
- `400`: Falha ao inserir pauta.

`POST: /link`

Parâmetros: Título da pauta e link inseridos por formulário na página.

Retorno:
- `200`: Operação bem sucedida.
- `400`: Falha ao inserir link.

`DELETE: /`

Parâmetros: Título da pauta a ser deletada inserido por formulário na página.

Retorno:
- `200`: Operação bem sucedida.
- `400`: Falha ao remover pauta.

2.4 /api/locais

Descritas no arquivo `servidor/routes/locais.js`.

`GET: /`

Parâmetros: Nenhum.

Retorno:
- `200`: Todos os locais.
- `404`: Nada encontrado.

`GET: /cenarios_mais_utilizados`

Parâmetros: Nenhum.

Retorno:
- `200`: Local mais utilizado e a quantidade de locações.
- `404`: Nada encontrado.

`POST: /`

Parâmetros: Logradouro, Numero_rua, Cep, Cidade, Estado, Bloco, Andar, Numero_sala.

Retorno:
- `200`: Local inserido com sucesso.
- `404`: Falha ao inserir local. Erro do banco.

`DELETE: /:id`

Parâmetros: ID de um local a ser deletado.

Retorno:
`200`: Operação bem sucedida.
`400`: Falha a remover local.

2.5 /api/equipamentos

Descritas no arquivo `servidor/routes/equipamentos.js`.

`GET: /`

Parâmetros: Nenhum.

Retorno:
`200`: Todos os equipamentos.
`404`: Nada encontrado.

`GET: /utilizados`

Parâmetros: Nenhum.

Retorno:
`200`: Todos os equipamentos que foram utilizados .
`404`: Nada encontrado.

`GET: /quantidadeUtilizada/maior`

Parâmetros: Nenhum.

Retorno:
`200`: Tipo de de equipamento que foi mais utilizado.
`404`: Nada encontrado.

`GET: /quantidadeUtilizada/total`

Parâmetros: Nenhum.

Retorno:
`200`: Quantas vezes cada tipo de equipamento foi utilizado.
`404`: Nada encontrado.

`GET: /estoque`

Parâmetros: Nenhum.

Retorno:
`200`: Retorna o número de equipamentos por tipo disponíveis no estoque.
`404`: Nada encontrado.

`POST: /`

Parâmetros: Tipo, Ano, Marca, Bloco, Andar, Numero.

Retorno:
- `200`: Equipamento inserido com sucesso.
- `404`: Falha ao inserir o equipamento.

`POST`: `/equipamentoUtilizado`

Parâmetros: Matéria, arquivo onde um equipamento foi utilizado.

Retorno:
- `200`: Operação bem sucedida.
- `404`: Falha ao inserir o equipamento.

2.6 /api/materias

Descritas no arquivo `servidor/routes/materias.js`.

`GET`: `/`

Parâmetros: Nenhum.

Retorno:
- `200`: Todas as matérias ordenadas por data de inclusão.
- `404`: Não encontrado.

`GET`: `/quantidade/:mes`

Parâmetros: Mês requisitado.

Retorno:
- `200`: Quantidade de matérias escritas em dado mês.
- `404`: Não encontrado.

`GET`: `/jornalista/:cpf`

Parâmetros: CPF de um jornalista.

Retorno:
- `200`: Todas as matérias escritas por um dado jornalista.
- `404`: Não encontrado.

`GET`: `/jornalistaspormes`

Parâmetros: Nenhum.

Retorno:
- `200`: Quantas matérias cada jornalista escreveu por mês.
- `404`: Não encontrado.

`GET`: `/quantidade/:jornalista`

Parâmetros: CPF de um dado jornalista.

Retorno:
- 200: Quantidade de matérias escritas por um jornalista.
- 404: Nada encontrado.

`POST: /`

Parâmetros: Titulo, jornalista e texto inseridos em formulário na página.

Retorno:
- 200: Matéria inserida com sucesso.
- 400: Falha ao inserir matéria.

`DELETE: /:id`

Parâmetros: Titulo de uma matéria a ser deletada.

Retorno:
- 200: Matéria removida com sucesso.
- 400: Falha ao remover matéria.

2.7 /api/materias_finais

Descritas no arquivo `servidor/routes/materias_finais.js`.

`GET: /`

Parâmetros: Nenhum.

Retorno:
- 200: Todas as matérias finais.
- 404: Não encontrado.

`GET: /episodio/:data`

Parâmetros: Data.

Retorno:
- 200: Mostra todas as matérias finais que foram ao ar no dia requisitado.
- 404: Não encontrado.

`GET: /creditos/:título`

Parâmetros: Título.

Retorno:
- 200: Retorna o nome de todas as pessoas que trabalharam em uma determinada matéria final
- 404: Não encontrado.

`GET: /quantidade/:jornalista`

Parâmetros: Jornalista.

Retorno:
- `200`: Retorna a quantidade de matérias finais que um determinado jornalista participou.
- `404`: Não encontrado.

`POST: /`

Parâmetros: Video_final, Editor, Bloco, Andar, Numero, Data, Periodo, Episodio.

Retorno:
- `200`: Insere uma matéria final.
- `400`: Erro ao inserir matéria.

`DELETE: /:video_final`

Parâmetros: Video_final.

Retorno:
- `200`: Matéria deletada com sucesso.
- `400`: Falha ao deletar matéria.

3 Conclusão

No começo tivemos dificuldade em direcionar o foco do que seria o trabalho, era muito abstrato, a abstração diminuiu ao definirmos o MER. Depois o problema foi simplificá-lo a um nível factível ao tempo remanescente da graduação mantendo, ainda, sua essência. Para a terceira parte: foi muito útil a criação de Consultas Relevantes na primeira parte; e foi difícil instalar `PostgreSQL` em nossos notebooks. Quanto à implementação da interface, a consideramos completamente fora do escopo da disciplina, tivemos a sorte de ter um aluno em nosso grupo que já tinha conhecimentos em full-stack, apesar de interessante e proveitoso pensamos que seria mais pertinente um simples script que invocasse as queries, sendo elas, efetivamente, o intuito da matéria.