

**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS  
ENGENHARIA DE COMPUTAÇÃO**

Beatriz Siqueira  
Eduardo Alves de Freitas  
Matheus Dutra Cerbino  
Sinval

**Trabalho Prático de Banco de Dados**  
Sistema de Controle de Envio de Trabalhos Acadêmicos

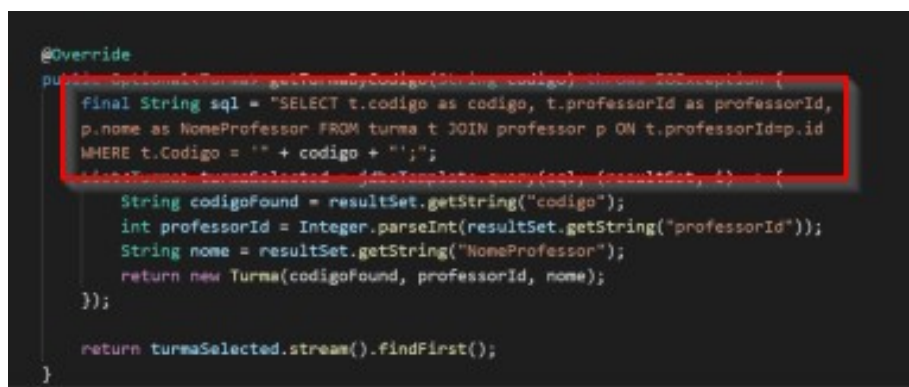
Belo Horizonte  
2020

## 1. Descrição Textual

O Sistema de Gestão de Submissão de Trabalhos feito pelo grupo, engloba um sistema de informação completo, contando com FrontEnd, BackEnd e acesso ao Banco de Dados. Eles se comunicam a fim de trazer ao usuário a experiência de criar turma, aluno, disciplina, curso e principalmente possibilitar os alunos a submeterem seus trabalhos. As regras de negócio que este sistema contempla, são as seguintes:

1. Notas só podem valer de 0 a 100;
2. A média para aprovação é 60 pontos;
3. Trabalho deve ser avaliado após ser enviado;
4. O trabalho não pode ser avaliado antes ou ao mesmo tempo de ser enviado;
5. Todo trabalho precisa de um professor avaliador;
6. Todo trabalho deve ter um ou mais alunos;
7. O trabalho deve pertencer a uma turma;
8. Toda turma deve ter um professor;
9. Toda turma pertence a uma disciplina;
10. Um trabalho pode ter sua nota alterada;

As tecnologias utilizadas foram SpringBoot para Java no Backend, Angular para JavaScript no Frontend, e Postgres no Banco de Dados. Essas foram as tecnologias escolhidas, visto que são atuais e familiares aos integrantes do grupo. Dessa forma, fizemos a implementação de rotas que faziam a comunicação entre o backend e o frontend, e que, a partir de strings no backend, conectavam com o banco de dados, como pode ser observado na imagem a seguir:

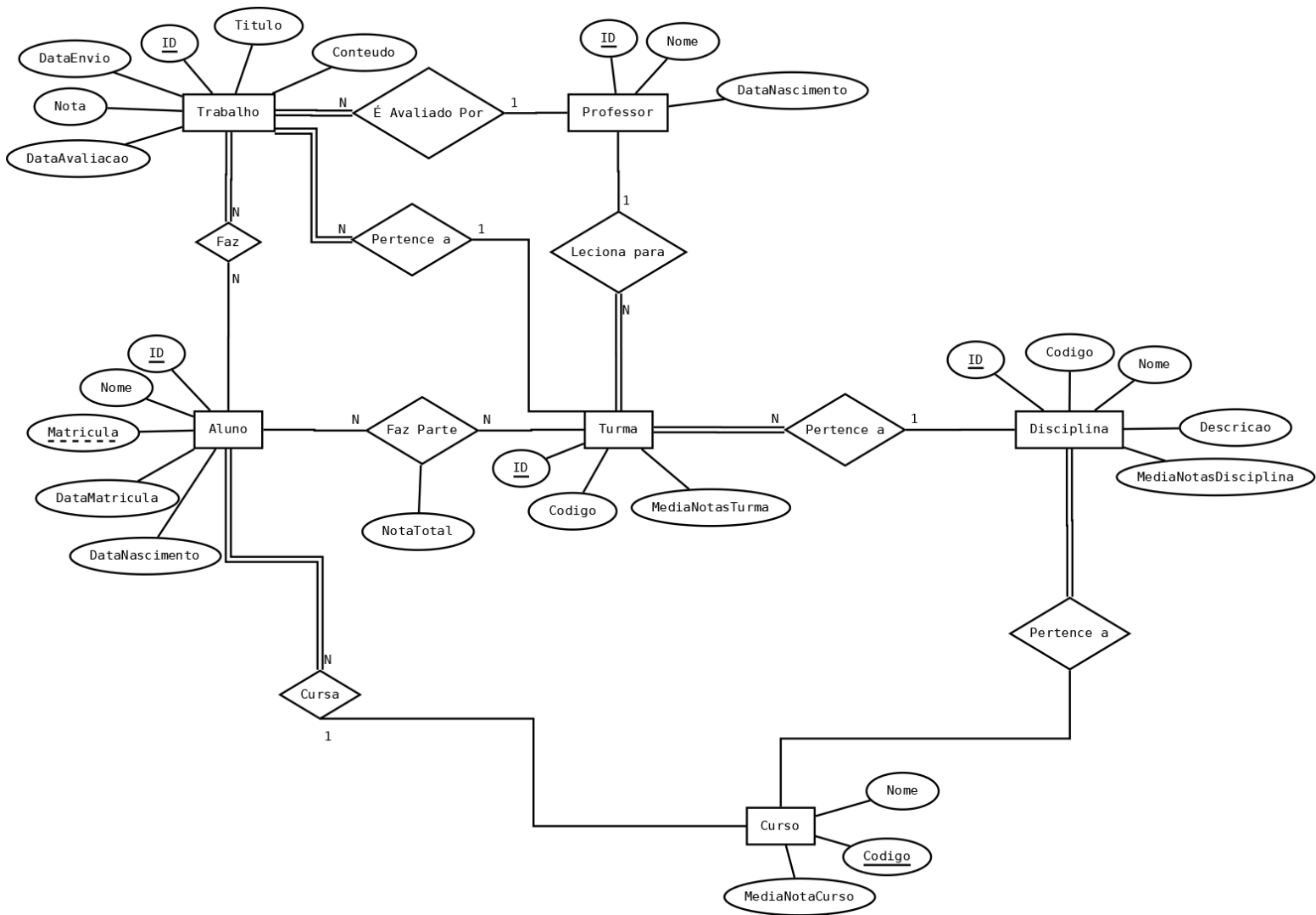


```
@Override
public Optional<Turma> getTurmaByCodigo(String codigo) throws IOException {
    final String sql = "SELECT t.codigo as codigo, t.professorId as professorId,
    p.nome as NomeProfessor FROM turma t JOIN professor p ON t.professorId=p.id
    WHERE t.Codigo = '" + codigo + "'";
    List<Turma> turmasSelected = jdbcTemplate.query(sql, (resultSet, i) -> {
        String codigoFound = resultSet.getString("codigo");
        int professorId = Integer.parseInt(resultSet.getString("professorId"));
        String nome = resultSet.getString("NomeProfessor");
        return new Turma(codigoFound, professorId, nome);
    });
    return turmasSelected.stream().findFirst();
}
```

Nesta imagem vê-se um exemplo de integração do backend com o Banco de Dados. As rotas principais são as rotas de get (Request Method "GET") e de insert(usam o Request Method "POST"). Elas basicamente mostram os dados que existem no banco de dados e inserem novos dados nas tabelas, respectivamente. Este trabalho foi interessante pois pudemos mexer, desde a construção até sua finalização de um sistema de submissão de trabalhos. Dessa forma vimos como os Bancos de Dados são construídos e planejados inicialmente em grandes sistemas, e como é necessário um bom planejamento, além de percebermos também a necessidade de uma boa estruturação na comunicação, e na coesão dos dados e triggers implementados.

**GitHub do Projeto:** <https://github.com/SinvalVJunior/Eldoom>

## 2. Diagramação do Banco em MER



### 3. Construção do Banco de Dados

create table if not exists curso

```
(
  id      serial not null
        constraint PK_Curso primary key,
  Codigo  text,
  Nome    text,
  MediaNotaCurso numeric(10,4)
);
```

create table if not exists aluno

```
(
  id      serial not null
        constraint PK_aluno primary key,
  Nome    text not null,
  Matricula integer not null
        constraint UQ_aluno_matricula unique,
  Cursold integer not null
        constraint FK_aluno_curso references curso,
  DataNascimento date,
  DataMatricula date
);
```

create table if not exists professor

```
(
  id      serial not null
        constraint PK_professor primary key,
  Nome    text not null,
  DataNascimento date
);
```

create table if not exists disciplina

```
(
  id      serial not null
        constraint PK_Disciplina primary key,
  Codigo  text,
  Nome    text,
  Descricao text,
  MediaNotasDisciplina numeric(10,4),
  Cursold integer
        constraint FK_disciplina_curso references curso
        on delete cascade
);
```

create table if not exists turma

```
(
  id      serial not null
        constraint PK_turma primary key,
  Codigo  text,
  MediaNotaTurma numeric(10,4),
  ProfessorId integer
        constraint FK_turma_professor_id references professor
        on delete set default,
  DisciplinaId integer
        constraint FK_turma_disciplina_id references disciplina
        on delete set default
);
```

create table if not exists aluno\_turma

```
(
  Alunold integer
    constraint FK_aluno_turma__aluno references aluno
      on delete cascade,
  Turmald integer
    constraint FK_aluno_turma__turma references turma
      on delete cascade,
  NotaTotal numeric(10, 4),
  constraint PK_aluno_turma primary key (Alunold, Turmald)
);
```

create table if not exists trabalho

```
(
  id serial not null
    constraint PK_trabalhos primary key,
  Titulo text,
  Conteudo text,
  ProfessorId integer not null
    constraint FK_trabalho_professor references professor
      on delete cascade,
  Turmald integer not null
    constraint FK_trabalho_turma references turma,
  DataEnvio timestamp not null,
  DataAvaliacao timestamp,
  Nota numeric(10, 4)
);
```

create table if not exists trabalho\_aluno

```
(
  Alunold integer
    constraint FK_trabalho_aluno_m2m__aluno references aluno
      on delete cascade,
  Trabalhold integer
    constraint FK_trabalho_aluno_m2m__trabalho references trabalho
      on delete cascade,
  constraint PK_trabalho_aluno primary key (Alunold, Trabalhold)
);
```

alter table disciplina

add constraint limite\_MediaDisciplina\_check

check (MediaNotasDisciplina >= 0 AND MediaNotasDisciplina <= 100);

alter table curso

add constraint limite\_MediaCurso\_check

check (MediaNotaCurso >= 0 AND MediaNotaCurso <= 100);

alter table turma

add constraint limite\_MediaTurma\_check

check (MediaNotaTurma >= 0 AND MediaNotaTurma <= 100);

alter table aluno\_turma

add constraint limite\_NotaFinal\_check

check (NotaTotal >= 0 AND NotaTotal <= 100);

--Trigger 1

create or replace function "Eldoom".*nota\_final\_procedure*() returns trigger  
language plpgsql

as

\$\$

BEGIN

UPDATE "Eldoom".aluno\_turma

SET NotaTotal= (SELECT avg(Trab.Nota)

```

        FROM "Eldoom".Trabalho Trab
        JOIN "Eldoom".trabalho_aluno TRAB_ALUN ON Trab.id = TRAB_ALUN.Trabalhold
        WHERE "Eldoom".aluno_turma.Alunold = TRAB_ALUN.Alunold
        AND "Eldoom".aluno_turma.Turmaid = Trab.turmaid
        GROUP BY "Eldoom".aluno_turma.Alunold, "Eldoom".aluno_turma.turmaid
        LIMIT 1)
WHERE (Alunold, Turmaid) = (SELECT TRAB_ALUN.Alunold, T.turmaid
        FROM "Eldoom".trabalho_aluno TRAB_ALUN
        JOIN "Eldoom".Trabalho T ON T.id = TRAB_ALUN.Trabalhold
        WHERE TRAB_ALUN.Trabalhold = NEW.id
        LIMIT 1);

RETURN NEW;
END ;
$$;

```

```

create trigger nota_final
after update of Nota
on "Eldoom".trabalho
for each row
execute
procedure "Eldoom".nota_final_procedure();

```

## --Trigger 2

```

create or replace function "Eldoom".media_turma_procedure() returns trigger
language plpgsql
as
$$
BEGIN
    UPDATE "Eldoom".turma
    SET MediaNotaTurma= (SELECT avg(aluno_turma.NotaTotal)
        FROM "Eldoom".aluno_turma aluno_turma
        WHERE aluno_turma.Turmaid = NEW.Turmaid
        GROUP BY aluno_turma.turmaid
        LIMIT 1)
    WHERE "Eldoom".turma.Turmaid = New.Turmaid;

    RETURN NEW;
END ;
$$;

```

```

create trigger media_turma
after update of NotaTotal
on "Eldoom".aluno_turma
for each row
execute
procedure "Eldoom".media_turma_procedure();

```

## --Trigger 3

```

create or replace function "Eldoom".media_disciplina_procedure() returns trigger
language plpgsql
as
$$
BEGIN
    UPDATE "Eldoom".disciplina
    SET MediaNotasDisciplina= (SELECT avg(turma.MediaNotaTurma)
        FROM "Eldoom".turma turma

```

```

        WHERE turma.id = NEW.id
        GROUP BY turma.id
        LIMIT 1)
WHERE "Eldoom".disciplina.id = New.id;

RETURN NEW;
END ;
$$;

```

```

create trigger media_disciplina
after update of MediaNotaTurma
on "Eldoom".turma
for each row
execute
procedure "Eldoom".media_disciplina_procedure();

```

#### --Trigger 4

```

create or replace function "Eldoom".media_curso_procedure() returns trigger
language plpgsql
as
$$
BEGIN
UPDATE "Eldoom".Curso
SET MediaNotaCurso= (SELECT avg(discip.MediaNotasDisciplina)
FROM "Eldoom".Disciplina discip
WHERE discip.id = NEW.id
GROUP BY discip.id
LIMIT 1)
WHERE "Eldoom".Curso.id = New.id;

RETURN NEW;
END ;
$$;

```

```

create trigger media_curso
after update of MediaNotasDisciplina
on "Eldoom".Disciplina
for each row
execute
procedure "Eldoom".media_Curso_procedure();

```

#### --Trigger 5

```

create or replace function "Eldoom".trabalho_data_avaliacao() returns trigger
language plpgsql
as
$$
BEGIN
IF ((NEW.dataavaliacao IS NOT NULL AND NEW.nota IS NULL) OR
(NEW.dataavaliacao IS NULL AND NEW.nota IS NOT NULL)) THEN
RAISE EXCEPTION USING MESSAGE = 'Para inserir a nota é necessario uma data de avaliacao';
END IF;

IF (NEW.dataavaliacao IS NOT NULL) THEN
IF (NEW.dataenvio IS NULL) THEN
IF (OLD.dataenvio IS NOT NULL AND NEW.dataavaliacao >= OLD.dataenvio) THEN
RETURN NEW;
ELSE
RAISE EXCEPTION USING MESSAGE = 'A data de avaliacao nao pode ocorrer antes da data de
envio';
END IF;

```

```
        ELSEIF (NEW.dataavaliacao < NEW.dataenvio) THEN
            RAISE EXCEPTION USING MESSAGE = 'A data de avaliacao nao pode ocorrer antes da data de
envio';
        END IF;
    END IF;
```

```
    RETURN NEW;
END;
$$;
```

```
create trigger trabalho_data_avaliacao
after insert or update of dataavaliacao, nota
on "Eldoom".trabalho
for each row
execute
    procedure "Eldoom".trabalho_data_avaliacao();
```



#### 4. Povoamento

##### --Inserção Curso

```
INSERT INTO curso (id, codigo, nome, medianotacurso) VALUES (default, 'ECOMP', 'Engenharia de Computacao', null);
```

```
INSERT INTO curso (id, codigo, nome, medianotacurso) VALUES (default, 'EELET', 'Engenharia Elétrica', null);
```

```
INSERT INTO curso (id, codigo, nome, medianotacurso) VALUES (default, 'EMEC', 'Engenharia Mecânica', null);
```

```
INSERT INTO curso (id, codigo, nome, medianotacurso) VALUES (default, 'LETR', 'Letras', null);
```

```
INSERT INTO curso (id, codigo, nome, medianotacurso) VALUES (default, 'ADM', 'Administracao', null);
```

##### --Inserção dos alunos

```
INSERT INTO aluno (id, nome, matricula, cursoid, DataNascimento, DataMatricula) VALUES (default, 'Jorge', 123456, 1, '1998-12-02 23:59:48.000000', '2007-11-20 23:59:48.000000');
```

```
INSERT INTO aluno (id, nome, matricula, cursoid, DataNascimento, DataMatricula) VALUES (default, 'Carlos', 234567, 1, '1995-08-08 23:59:48.000000', '2012-10-18 23:59:48.000000');
```

```
INSERT INTO aluno (id, nome, matricula, cursoid, DataNascimento, DataMatricula) VALUES (default, 'Marcus', 345678, 2, '1997-09-09 23:59:48.000000', '2010-06-17 23:59:48.000000');
```

```
INSERT INTO aluno (id, nome, matricula, cursoid, DataNascimento, DataMatricula) VALUES (default, 'Matheus', 456789, 3, '1993-07-05 23:59:48.000000', '2008-07-15 23:59:48.000000');
```

```
INSERT INTO aluno (id, nome, matricula, cursoid, DataNascimento, DataMatricula) VALUES (default, 'Lucas', 012345, 4, '2000-03-02 23:59:48.000000', '2009-05-16 23:59:48.000000');
```

##### --Inserção dos Professores

```
INSERT INTO professor (id, nome, DataNascimento) VALUES (default, 'Natalia', '1983-03-02 23:59:48.000000');
```

```
INSERT INTO professor (id, nome, DataNascimento) VALUES (default, 'Evandrino', '1966-09-15 23:59:48.000000');
```

```
INSERT INTO professor (id, nome, DataNascimento) VALUES (default, 'Edson', '1971-05-07 23:59:48.000000');
```

```
INSERT INTO professor (id, nome, DataNascimento) VALUES (default, 'Thiago', '1973-06-10 23:59:48.000000');
```

```
INSERT INTO professor (id, nome, DataNascimento) VALUES (default, 'Bruno', '1955-01-01 23:59:48.000000');
```

##### --Insercao Disciplina

```
INSERT INTO disciplina (id, codigo, nome, descricao, medianotasdisciplina, cursoid) VALUES (default, 'ECOMP101', 'Introdução a Eng. Comp', 'Introduz as possibilidades relacionadas ao curso', null, 1);
```

```
INSERT INTO disciplina (id, codigo, nome, descricao, medianotasdisciplina, cursoid) VALUES (default, 'ELET201', 'Circuitos', 'Permitir o entendimento e produção de circuitos elétricos', null, 2);
```

```
INSERT INTO disciplina (id, codigo, nome, descricao, medianotasdisciplina, cursoid) VALUES (default, 'EMEC301', 'AR Condicionados', 'Discutir o funcionamento mecânico de sistemas de ar condicionados', null, 3);
```

```
INSERT INTO disciplina (id, codigo, nome, descricao, medianotasdisciplina, cursoid) VALUES (default, 'LETR101', 'Alfabeto', 'Discutir a origem e funcionalidades do alfabeto ', null, 4);
```

```
INSERT INTO disciplina (id, codigo, nome, descricao, medianotasdisciplina, cursoid) VALUES (default, 'ADM202', 'Excel Avançado', 'Discutir conhecimentos sobre o uso do Excel', null, 5);
```

#### --Inserção de Turmas

```
INSERT INTO turma (id, codigo, professorid, disciplinaid) VALUES (default, '101', 1, 1);
```

```
INSERT INTO turma (id, codigo, professorid, disciplinaid) VALUES (default, '201', 2, 1);
```

```
INSERT INTO turma (id, codigo, professorid, disciplinaid) VALUES (default, '301', 3, 2);
```

```
INSERT INTO turma (id, codigo, professorid, disciplinaid) VALUES (default, '401', 4, 3);
```

```
INSERT INTO turma (id, codigo, professorid, disciplinaid) VALUES (default, '501', 5, 4);
```

#### --Inserção de Trabalhos

```
INSERT INTO trabalho (id, titulo, conteudo, professorid, turmaid, dataenvio, dataavaliacao, nota) VALUES (default, 'Estruturas de Dados', 'Filas, Listas e Pilhas', 1, 2, '2020-12-01 23:59:40.000000', '2020-12-02 13:00:23.000000', 100.0000);
```

```
INSERT INTO trabalho (id, titulo, conteudo, professorid, turmaid, dataenvio, dataavaliacao, nota) VALUES (default, 'Banco de Dados', 'Modelo Objeto-Relacional', 2, 4, '2020-12-02 23:59:48.000000', '2020-12-03 16:40:04.000000', 60.0000);
```

```
INSERT INTO trabalho (id, titulo, conteudo, professorid, turmaid, dataenvio, dataavaliacao, nota) VALUES (default, 'Modelagem de Software', 'Metodologia SCRUM', 3, 3, '2020-12-03 23:59:54.000000', '2020-12-13 23:59:26.000000', 57.0000);
```

```
INSERT INTO trabalho (id, titulo, conteudo, professorid, turmaid, dataenvio, dataavaliacao, nota) VALUES (default, 'Estruturas de Dados II', 'Arvore B', 4, 2, '2020-12-04 23:59:04.000000', '2020-12-13 23:59:31.000000', 95.0000);
```

```
INSERT INTO trabalho (id, titulo, conteudo, professorid, turmaid, dataenvio, dataavaliacao, nota) VALUES (default, 'Sistemas Operacionais', 'Kernel', 5, 1, '2020-12-05 23:59:08.000000', null, null);
```

#### --Inserção Aluno-Turma

```
INSERT INTO aluno_turma (alunoid, turmaid, notatotal) VALUES (2, 1, null);
```

```
INSERT INTO aluno_turma (alunoid, turmaid, notatotal) VALUES (3, 2, null);
```

```
INSERT INTO aluno_turma (alunoid, turmaid, notatotal) VALUES (4, 3, null);
```

```
INSERT INTO aluno_turma (alunoid, turmaid, notatotal) VALUES (5, 4, null);
```

```
INSERT INTO aluno_turma (alunoid, turmaid, notatotal) VALUES (1, 5, null);
```

#### --Inserção Trabalho-Aluno

```
INSERT INTO trabalho_aluno (alunoid, trabalhoid) VALUES (3, 1);
```

```
INSERT INTO trabalho_aluno (alunoid, trabalhoid) VALUES (2, 2);
```

```
INSERT INTO trabalho_aluno (alunoid, trabalhoid) VALUES (5, 3);
```

```
INSERT INTO trabalho_aluno (alunoid, trabalhoid) VALUES (1, 4);
```

```
INSERT INTO trabalho_aluno (alunoid, trabalhoid) VALUES (2, 5);
```

## **5. Sistema de Informação**