

Projeto Final

Pedro Henrique de Brito Agnes, 18/0026305

Pedro Pessoa Ramos, 18/0026488

Introdução

Para o projeto final da disciplina de Banco de Dados, foi desenvolvido um *software* com o objetivo de auxiliar o professor no modelo de aulas remotas. Neste sistema, o professor seria capaz de consultar a acessibilidade dos alunos a recursos como internet, computador, dispositivos móveis, o que permite avaliar a situação de seus alunos para encontrar um formato de aula que não os prejudique. Também, entre diversas outras possibilidades, é possível manter registros da participação dos alunos em cada aula, seja síncrona ou assíncrona, já que pode cadastrar os tipos de participação como fazer exercícios e assistir a aula.

Detalhes Técnicos

Foram usadas a linguagem de programação PHP e um banco de dados com o SGBD MySQL para o desenvolvimento do sistema. Para a implementação da interface de usuário, foi desenvolvida uma API que serve como o CRUD do projeto, responsável por acessar cada tabela do banco para as operações de inserção, atualização, remoção e consulta. Já no banco, foram criadas 10 entidades, que totalizaram 13 tabelas seguindo o modelo proposto para aulas remotas, como informações do aluno, professor, participação das aulas, plataformas em que as aulas são dadas, entre outros.

O CRUD do projeto está funcionando em um servidor web e pode ser acessado pelo link <https://api2.opessoa.com.br/ProjetoBD>, assim como o banco também está no mesmo domínio. Após acessar o link, uma resposta no formato JSON deve ser recebida informando um erro, pois não foi informada uma classe. Existe uma classe para cada tabela, chamadas de *controllers*, que efetuam as operações de manipulação e consulta dos dados da tabela conforme solicitado pelo usuário. Como exemplo, se quisermos acessar o *controller* dos alunos, podemos chamar o `AlunosController` e, em seguida, a função que queremos utilizar dele, que são as mesmas para cada um: `get`, `insert`, `delete` e `update` (6).

O padrão de nomenclatura dos *controllers* é o nome da tabela em plural seguido por "*Controller*" e usando *camel case* para tabelas de nome composto como as de relacionamento. Acima foi mostrado como utilizar o comando `SELECT`, porém os comandos `INSERT`, `DELETE` e `UPDATE` precisam de parâmetros adicionais, que são os valores de cada coluna a inserir e no caso de excluir, devem ser passadas todas as chaves primárias, tudo separado por `/` e na ordem vista durante o `SELECT`. Já no `update`, são passadas todas as chaves assim como na remoção e, em seguida os parâmetros a serem alterados na ordem, assim como na inserção. Posteriormente serão mostrados exemplos de como efetuar cada operação (6).

Modelo Entidade Relacionamento

A seguir, podemos ver o MER do projeto, que retrata a ideia inicial do projeto, onde foram pensados e diagramados os relacionamentos, as entidades e os atributos e chaves primárias de cada. Aqui foi onde começou a estrutura do banco de dados desenvolvido.

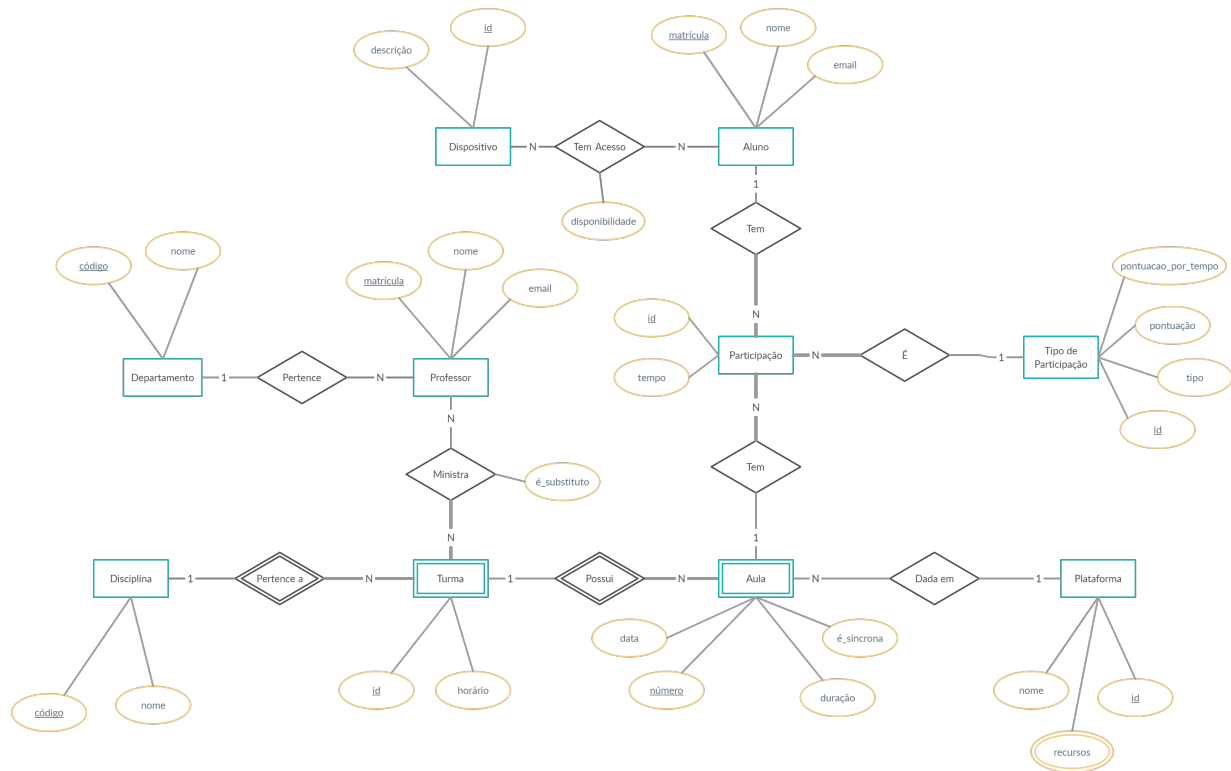


Figura 1: Modelo Entidade Relacionamento

Lista de entidades:

- Aluno
- Aula
- Departamento
- Disciplina
- Dispositivo
- Participação
- Plataforma
- Professor
- Tipo de Participação
- Turma

Modelo Relacional

Abaixo podemos ver o MR do projeto, onde foram especificados detalhes mais internos ao banco como os tipos de cada atributo e a organização das chaves estrangeiras e tabelas de relacionamento, assim como as tabelas para atributos multivalorados. Aqui foi onde começou a implementação do banco de dados desenvolvido. Na página 5, pode ser encontrada a lista completa das tabelas do banco, assim como o mapeamento de cada uma para a respectiva classe controladora.

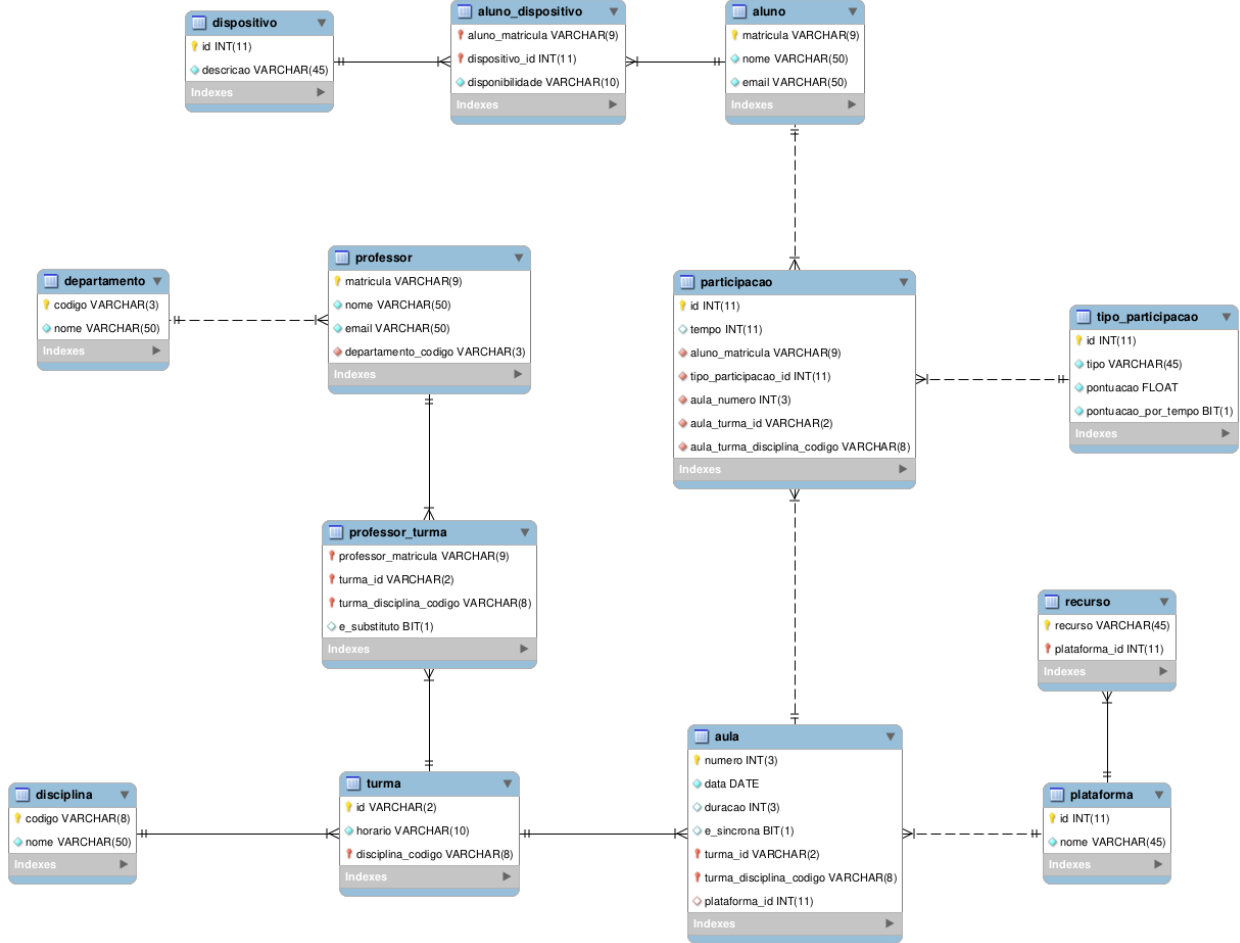


Figura 2: Modelo Relacional

Álgebra Relacional

Todas as combinações possíveis de alunos, dispositivos, e participação.

$$dispositivo \times aluno \times tipo_participacao \quad (1)$$

Todas as pessoas cadastradas (alunos e professores) mais o departamento do professor e dispositivo do aluno.

$$\sigma_{aluno.matricula=aluno_dispositivo.aluno_matricula \text{ AND } dispositivo.id=aluno_dispositivo.dispositivo_id} (aluno \times aluno_dispositivo \times dispositivo) \cup (professor \bowtie departamento) \quad (2)$$

Relação de professor com turma.

$$professor \bowtie professor_turma \bowtie turma \quad (3)$$

Todas as possíveis formas de passar aulas.

$$tipo_participacao \times plataforma \times recurso \quad (4)$$

Disciplinas que não tiveram aulas.

$$SemAula \leftarrow \pi_{turma_id, turma_disciplina_codigo}(\sigma_{group\ by\ turma_id\ having\ COUNT(*)=0}(aula)) \quad (5)$$
$$\sigma_{disciplina.codigo=SemAula.turma_disciplina_codigo}(SemAula)$$

Forma Normal

Avaliação das formas normais em cinco tabelas (**PEDRO PESSOA**).

Camada de Persistência

Abaixo temos um diagrama mostrando como a interface da API por meio da classe `mysqli` do php extendida por `Conn` no projeto acessa o banco de dados para realizar as operações do CRUD. Logo abaixo, temos uma tabela de mapeamento de cada tabela do banco para a respectiva classe controladora.

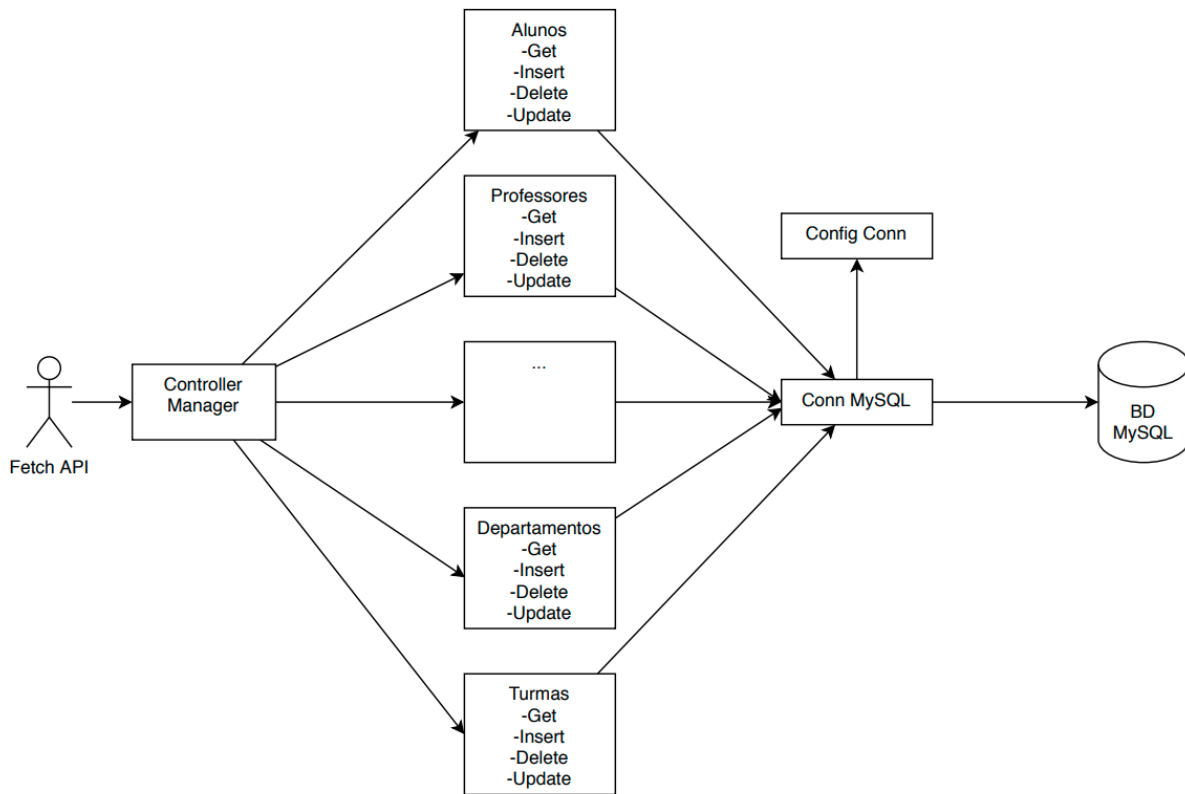


Figura 3: Diagrama de acesso a camada de persistência

Tabela	<i>Controller</i>
aluno	AlunosController
aluno_dispositivo	AlunosDispositivosController
aula	AulasController
departamento	DepartamentoController
disciplina	DisciplinasController
dispositivo	DispositivosController
participacao	ParticipacoesController
plataforma	PlataformasController
professor	ProfessoresController
professor_turma	ProfessoresTurmasController
recurso	RecursosController
tipo_participacao	TiposParticipacaoController
turma	TurmasController

Exemplos das Operações na tabela de Alunos

É importante mencionar que nem todos os atributos são obrigatórios durante a inserção e atualização de dados. Nesses casos, podem ser usadas duas barras seguidas separadas por um espaço para pular um parâmetro (/ /), ou se o parâmetro indesejado for o último, bastaria ignorar a sua existência por completo.

- SELECT - <https://api2.opessoa.com.br/ProjetoBD/AlunosController/get>
- INSERT - <https://api2.opessoa.com.br/ProjetoBD/AlunosController/insert/200012345/JosédosTestes/jt@tst.com>
- UPDATE - <https://api2.opessoa.com.br/ProjetoBD/AlunosController/update/200012345/JosephdosTestes>
- DELETE - <https://api2.opessoa.com.br/ProjetoBD/AlunosController/delete/200012345>

Considerações

Para o projeto desenvolvido, todo o código-fonte pode ser encontrado no github: <https://github.com/Pedenite/Projeto-BD-UnB>. Os scripts PHP estão divididos entre a pasta raiz e a pasta `controllers`, assim como todos os scripts SQL utilizados podem ser encontrados na pasta `sql`. Adicionalmente, este documento, o MR e o MER podem ser encontrados na pasta `documentacao` e existe uma descrição do trabalho na raiz do projeto no github, assim como instruções para executar localmente.

Para a execução do projeto em uma máquina local, deverá ser configurado um servidor Apache para o php, por exemplo, ou utilizar os comandos docker disponibilizados para rodar um container já configurado (mais informações no git). De todo modo, como não é qualquer um que possui acesso ao banco deste projeto, deve ser criado um banco local com os scripts disponibilizados e adicionado o seguinte arquivo `Config.php` na pasta `controllers`:

```
1 // Nome do arquivo: Config.php
2 <?php
3 define('HOST', 'localhost:3306'); // ou o host e porta alternativos
  usados
4 define('USUARIO', 'root'); // ou o usuário alternativo usado
5 define('SENHA', ''); // ou a senha definida
6 define('DB', 'opessoa08_ProjetoDB'); // ou o nome alternativo dado ao
  banco
```