

Documentação Trabalho Prático - Banco de Dados II

André Felipe Magalhães Silva, Anna Luiza Pereira Rosa,
Daniel Henrique Ferreira Carvalho,
José Luiz Corrêa Junior, Tiago Rafael Amaral Reis

¹Instituto de Matemática e Computação – Universidade Federal de Itajubá (UNIFEI)
Caixa Postal 50 – 37500 903 – Itajubá – MG – Brasil

{andre, anna.luiza, danielhfc01, juninhopc, d2021021338}@unifei.edu.br

1. Descrição do Trabalho

Este documento descreve os processos utilizados no desenvolvimento do Trabalho Prático na disciplina de Banco de Dados II do curso de graduação em Ciência da Computação da Universidade Federal de Itajubá no 1º semestre de 2022.

O trabalho foi dividido em duas etapas. Na primeira foi desenvolvida uma aplicação que consome dados da API Web do Spotify, que foram tratados e carregados no banco de dados relacional PostgreSQL utilizando o *driver* Psycopg. Já na segunda, foi desenvolvida uma aplicação utilizando o framework Angular que consome os dados do banco de dados através de uma API contruída com NestJS e TypeORM Além disso, foi utilizado o software JMeter para realizar as medidas de performance do sistema gerenciador de banco de dados.

2. Primeira Etapa

2.1. PostgreSQL e Psycopg

PostgreSQL é uma ferramenta que atua como Sistema de Gerenciamento de Banco de Dados (SGBD) relacionados. Seu foco é permitir a implementação da linguagem SQL em estruturas, garantindo um trabalho com os padrões desse tipo de ordenação dos dados. É um sistema que lida bem com altos volumes de solicitações e com cargas de trabalho grandes, ou seja, funciona muito bem para sites com intensidade de acesso.

Para o reconhecimento do banco através de uma linguagem de programação, são utilizados *drivers* de conexão. Uma das maneiras de fazer isso com a linguagem de programação Python é com o adaptador Psycopg. Esse *driver* converte variáveis Python em valores SQL usando seus tipos e determinando a função usada para converter o objeto em uma representação de string adequada para PostgreSQL; muito desses tipos padrões já são adaptados à representação SQL correta .

2.2. Extração de Dados da API e Carregamento do Banco

Utilizando a documentação disponível no site Spotify for Developers, a extração foi feita da seguinte forma:

3. Segunda Etapa

3.1. Tecnologias

3.1.1. Angular, Node.js e TypeScript

Angular é uma plataforma e framework para construção de interface de aplicações utilizando HTML, CSS e TypeScript. Dentre os elementos que tornam essa ferramenta interessante, pode-se destacar os componentes, templates, roteamento, módulos, injeção de dependências, ferramentas de automatização, etc., além de ser *open source*.

O Node.js pode ser definido como um ambiente de execução Javascript *server-side* que permite executar aplicações desenvolvidas com a linguagem de forma autônoma, sem depender de um navegador. Já o Typescript é definido como um *superset* da linguagem de programação Javascript, melhorando o suporte à programação orientada a objetos com várias possibilidades que a linguagem pura não dispõe; além da possibilidade de aplicar tipagem estática à programação com os conceitos de encapsulamento, herança, abstração e polimorfismo.

3.1.2. NestJS e TypeORM

NestJS é uma estrutura para criar aplicativos Node.js eficientes e escaláveis do lado do servidor. Ele utiliza uma arquitetura bem definida e pronta para uso que inclui o uso de bibliotecas já consolidadas e testadas, diminuindo as decisões, amenizando inconsistências no código e facilitando o entendimento de quem está olhando o projeto pela primeira vez. Assim, é possível criar aplicações testáveis, escaláveis, acopladas e fáceis de manter.

Um ORM ou Mapeamento Objeto-Relacional é uma técnica utilizada para fazer o mapeamento entre sistemas orientados a objetos e bancos de dados relacionais. Nessa técnica, as tabelas do banco são representadas por classes e os registros das tabelas são instâncias dessas classes. Dessa forma, é possível manipular o banco SQL sem a necessidade de escrever SQL de fato, utilizando-se a própria linguagem de programação.


O TypeORM é um ORM que pode ser executados em diversas plataformas; É muito influenciado por outros ORM's, a exemplo do Hibernate, e entre suas principais características, estão: o suporte a transações sólidas (ACID), relacionamentos, e *lazy loading* (carregamento adiantado ou tardio), replicação de leitura, entre várias outras. É um dos ORMs mais maduros disponíveis para Node.js e como é escrito em TypeScript, funciona muito bem com a estrutura Nest.

3.2. API - Backend

As APIs proporcionam a integração entre sistemas que possuem linguagem totalmente distintas de maneira ágil. As possibilidades disponibilizadas pelo uso das APIs proporcionam para os desenvolvedores de softwares e aplicativos a possibilidade de conectar tecnologias heterogêneas, como diferentes bancos de dados, por exemplo.

Gerada uma aplicação Nest e com os devidos pacotes instalados, foi criado um arquivo .env que contém os dados do banco. Os dados necessários para a configuração da conexão entre a API e o banco são o tipo do SGBD, o *host*, a porta, o usuário e senha, o nome do banco de dados, as entidades (nesse caso, as tabelas que serão utilizadas) e uma

variável de sincronização automática. Uma vez definidos os dados, a conexão foi feita através do seguinte trecho de código:



```
1  @Module({
2    imports: [
3      ConfigModule.forRoot(),
4      TypeOrmModule.forRoot({
5        type: 'postgres',
6        host: process.env.DB_HOST,
7        port: parseInt(process.env.DB_PORT),
8        username: process.env.DB_USERNAME,
9        password: process.env.DB_PASSWORD,
10       database: process.env.DB_DATABASE,
11       entities: [__dirname + '/*.entity{.ts,.js}'],
12       synchronize: (process.env.DB_SYNCHRONIZE === 'true'),
13     }),
14     ArtistModule,
15     TrackModule,
16     AlbumModule,
17     ArtistAlbumModule,
18     ArtistGenresModule,
19     ArtistTrackModule
20   ],
21   controllers: [AppController],
22   providers: [AppService],
23 })
24 export class AppModule {}
```

Figura 1. Conexão entre a API e o banco

Os Modules importados na configuração são referentes às tabelas utilizadas no trabalho. Como já citado, com o uso do ORM, as tabelas do banco passam a ser tratadas como classes ou entidades. Isso significa que cada um dos atributos dessas entidades precisam ser definidos de acordo com a sua classificação dentro do SGBD antes de serem utilizados pela entidade de fato.

É possível configurar desde o tipo da coluna e o tipo de registro até os relacionamentos. Um exemplo de definição de uma entidade dentro do ORM é apresentada na Figura 2. Nesse exemplo, a coluna 'id' é definida como *primary key* (PK) ou chave primária da tabela e as demais como colunas simples, respeitando o tipos dos dados.

Além disso, nessa modelagem, cada instância de *track* está relacionada a uma instância de *album* (1 : 1). Isso faz com que seja necessário apontar o relacionamento entre essas entidades, o que é feito por meio de uma *foreign key* (FK) ou chave estrangeira. Nesse caso, o atributo 'album_id' da entidade *track* referencia a PK da entidade *album* por meio do decorator *@JoinColumn*, se transformando em uma FK.

```
1 import { Album } from "src/album/entities/album.entity";
2 import { Column, Entity, JoinColumn, OneToOne, PrimaryColumn } from "typeorm";
3
4 @Entity()
5 export class Track {
6     @PrimaryColumn({ name: 'id', type: 'varchar'})
7     id: string;
8
9     @Column({ name: 'name', type: 'varchar'})
10    name: string;
11
12    @Column({ name: 'duration', type: 'int'})
13    duration: number;
14
15    @Column({ name: 'explicit', type: 'boolean'})
16    explicit: boolean;
17
18    @Column({ name: 'track_number', type: 'int'})
19    track_number: number;
20
21    @Column({ name: 'qtd_artistas', type: 'int'})
22    qtd_artistas: number;
23
24    @OneToOne(() => Album)
25    @JoinColumn({ name: 'album_id'})
26    album_id: Album;
27 }
```

Figura 2. Definição na entidade *track*

3.3. Aplicação - Frontend

4. Análise de Performance - JMeter

5. Relatório AD-HOC

6. Conclusão

Referências

Carneiro, A. (2021). Criando uma API com NestJS. Disponível em:
<https://www.sidechannel.blog/criando-uma-api-com-nestjs/>.

Spotify (2022). Spotify Web API. Disponível em:
<https://developer.spotify.com/documentation/web-api/reference/>.

TypeORM (2022). Disponível em: <https://typeorm.io/>.