

Prova III B – Técnicas de Programação II – DSM - Prof. Henrique Louro - 04/12/2025

- 1) Crie a pasta **Prova3** no local de sua preferência no computador;
- 2) Abra a pasta no VS Code e acesse o terminal;
- 3) No terminal, execute o comando **npm init -y** para criar o arquivo **package.json**;
- 4) No terminal, execute o comando **npm i express dotenv mongoose** para instalar os pacotes do Mongoose;
- 5) No terminal, execute o comando **npm i -D @types/express** para instalar o pacote que contém as definições de tipos do pacote express;
- 6) No terminal, execute o comando **npm i -D ts-node ts-node-dev typescript** para instalar os pacotes ts-node, ts-node-dev e Typescript como dependências de desenvolvimento;
- 7) No terminal, execute o comando **tsc --init** para criar o arquivo **tsconfig.json** de opções e configurações para o compilador TS;
- 8) Crie o arquivo **.gitignore** na raiz do projeto e coloque a linha para ignorar a pasta **node_modules**;
- 9) Crie o arquivo **.env** na raiz do projeto e coloque a seguinte variável de ambiente: **PORT = 3001**;
- 10) Coloque as seguintes propriedades no arquivo **package.json** dentro do grupo “scripts”:

```
"scripts": {
    "start": "ts-node ./src",
    "dev": "ts-node-dev ./src"
```

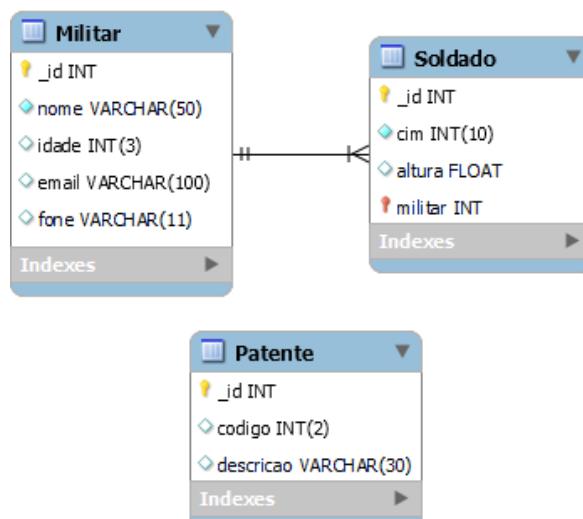
- 11) Crie a pasta **src** na raiz do projeto e dentro dela as pastas **controllers**, **models** e **routes** necessárias para organização dos códigos;
- 12) Coloque no arquivo **src/index.ts** o código para subir o servidor express;
- 13) Você pode encontrar os códigos necessários para conexão com o banco de dados MongoDB, bem como os arquivos principais do projeto, em meus repositórios no GitHub em:

<https://github.com/hdblouro/biblioteca>

<https://github.com/hdblouro/escola>

Inclusive, sugiro que clone um desses repositórios para fazer a prova;

- 14) O nome da coleção (banco) a ser criado no Mongo deverá ser: **p3tp2militar**;
- 15) Dado o MER a seguir, implemente-o em TypeScript com o Mongoose:



Prova III B – Técnicas de Programação II – DSM - Prof. Henrique Louro - 04/12/2025

- 16) Deverão ser criados os **Schemas, Controllers e Routes**, conforme modelo da revisão dada em sala de aula;
- 17) As informações a serem armazenadas nos campos dos documentos (tabelas) deverão ser validadas de acordo com o mostrado no MER e requisitos listados a seguir. Além disso, todas deverão ter mensagens informando ocorrências de erros, a serem capturadas nos **controllers**:
- 18) Validações dos campos dos documentos **Militar**:
- nome: não pode ser nulo;
 - idade: valor mínimo 18;
 - email:
 - não pode ser nulo;
 - deverá ser único;
 - validado pelo *validator* com requisitos informados abaixo:
 - O e-mail precisa ter pelo menos um "@" e um ponto. Utilize a expressão regular `/^[\^s@]+@[^\s@]+\.[^\s@]+$/` para fazer tal validação;
 - Só poderão ser aceitos e-mails de militares das forças armadas brasileira e devem conter `@eb` ou `@marinha` ou `@fab`, bem como o sufixo `".mil.br"`;
 - fone:
 - não pode ser nulo;
 - validado pelo *validator* com requisitos informados abaixo:
 - O número de telefone deverá ter de 10 a 11 dígitos entre 0 e 9. Utilize a expressão regular `/^[\^0-9]{10,11}\$/` para fazer tal validação;
 - Os dois primeiros dígitos do número de telefone devem indicar um DDD válido. Segue array numérico, com os números de DDDs válidos no Brasil:
`const ddd = [11,12,13,14,15,16,17,18,19,21,22,24,27,28,31,32,33,34,35,37,38,41,42,43,44,45,46,47,48,49,51,53,54,55,61,62,63,64,65,66,67,68,69,71,73,74,75,77,79,81,82,83,84,85,86,87,88,89,91,92,93,94,95,96,97,98,99];`
- 19) Validações dos campos dos documentos **Soldado**:
- cim:
 - não pode ser nulo;
 - deverá ser único;
 - altura: valores reais $\geq 1,62m$;
 - militar:
 - deverá ser do tipo: `mongoose.Schema.Types.ObjectId` (`Militar`);
 - não poderá ser nulo;
 - validado pelo *validator* que deverá verificar se o ID pertence a um documento Militar cadastrado no banco;
- 20) Validações dos campos dos documentos **Patente**:
- codigo: $0 < \text{codigo} \leq 20$;
 - descricao: não pode ser nulo;

Prova III B – Técnicas de Programação II – DSM - Prof. Henrique Louro - 04/12/2025

- 21) Logo após os **Schemas** criados coloque os códigos necessários para criar os modelos no Mongoose e faça suas exportações;
- 22) Crie os **Controllers** para cada **Schema** dentro da pasta **controllers** criada anteriormente. Cada **controller** deverá ter no mínimo os seguintes métodos: *create, list, delete* e *update*. Os métodos deverão ser implementados seguindo o padrão do Mongoose, vistos em sala de aula;
- 23) Os métodos **list** dos **controllers**, deverão listar o que se pede abaixo:
 - a. Militar: todos os militares cadastrados e seus dados, em ordem alfabética;
 - b. Soldado: todos os soldados cadastrados, seus dados e dos Militares a que estiverem relacionados;
 - c. Patente: todas as patentes cadastradas e seus dados, em ordem alfabética pela descrição;
- 24) Crie as rotas padrões no arquivo **index.ts** dentro da pasta **routes**, criada anteriormente, conforme modelo dado em sala de aula;
- 25) Crie as rotas para os métodos *create, list, delete* e *update* dos **Controllers**, para cada Schema, conforme modelo dado em sala de aula;
- 26) Crie comandos “curl” para testar a aplicação e suas rotas. Salve os comandos em um txt na raiz do projeto:
 - Cadastrar 2 militares;
 - Cadastrar 2 soldados, relacionados com um dos militares cada;
 - Cadastrar 2 patentes;
 - Alterar o cadastro de 1 militar;
 - Alterar o cadastro de 1 soldado;
 - Alterar o cadastro de 1 patente;
 - Listar os militares;
 - Listar os soldados associados aos seus militares;
 - Listar as patentes;
 - Excluir 1 soldado e o registro do militar relacionado;
 - Excluir 1 patente.
- 27) Suba o projeto em um repositório privado no GitHub compartilhado comigo. Informe na atividade do Teams o link desse repositório, para posterior avaliação.
- 28) Meu usuário no GitHub é: **hdblouro**
- 29) **Atenção:** Provas muito semelhantes serão objeto de uma avaliação mais rigorosa e se constatada a cola, os envolvidos terão suas notas zeradas.



Boa prova!