

 02

## Preparando o ambiente

Neste curso vamos retomar o projeto do curso anterior e continuar a partir dele. Se você fez a sua implementação e tem o projeto, pode aproveitar o código e somente adicionar os passos do tutorial a seguir; ou se preferir pode baixar o [repositório inicial \(<https://github.com/alura-cursos/1982-graphql>\)](https://github.com/alura-cursos/1982-graphql) do curso que vamos começar agora, com as implementações extras que faremos agora.

Neste curso vamos adicionar uma base de dados SQL ao projeto. Por motivos de praticidade e para não tirar o foco do curso, vamos usar o **SQLite**, um gerenciador de SQL de implementação rápida e bom para testes. Para conectar e fazer consultas ao banco, vamos usar o **dBeaver**.

Importante: Para ambos os casos, independentemente de você continuar com o

projeto do curso anterior ou baixar o repositório, é preciso instalar o dBeaver ou algum outro cliente de SQL de sua preferência, **ou** instalar as libs do SQLite caso prefira acessar o banco diretamente pelo terminal.

## Instalação do dBeaver

O dBeaver é uma ferramenta para conexão e gerenciamento de bancos, e pode ser usada com praticamente todos os gerenciadores de SQL mais usados: MySQL, PostgreSQL, SQLite, Oracle e etc.

Você pode fazer o download e a instalação da *versão Community* (gratuita e open source) direto na [página oficial \(https://dbeaver.io/download/\)](https://dbeaver.io/download/).

## Instalação do SQLite

Você pode baixar o pacote de instalação do SQLite na [página de downloads \(https://www.sqlite.org/download.html\)](https://www.sqlite.org/download.html) de acordo

com o seu sistema operacional, ou se preferir, pode seguir os passos abaixo.

Lembrando que, se preferir acessar o banco utilizando o dBeaver ou cliente similar, não é preciso instalar o SQLite.

## Instalação no Windows

1. Na [página de downloads](https://www.sqlite.org/download.html) (<https://www.sqlite.org/download.html>) baixe a opção “sqlite-tools”.
2. Crie uma pasta (por exemplo, `C:\sqlite`) e extraia o conteúdo do zip dentro dela. Agora a pasta deve conter três executáveis:  
`sqldiff.exe` , `sqlite3.exe` e `sqlite3-analyzer.exe` .
3. Abra o terminal do Windows, navegue até a pasta (no caso, `C:\sqlite`) e digite o comando `sqlite3`.
4. Você deve ter uma mensagem similar a esta:

```
SQLite version [versão e data]
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a
sqlite>
```

[COPIAR CÓDIGO](#)

Está tudo pronto para acessar arquivos SQLite pelo terminal.

## Instalação no Mac

A princípio, o SQLite já vem instalado por padrão no MacOs. Faça o teste digitando `sqlite3` direto no terminal. Caso apareça uma mensagem semelhante a:

```
SQLite version [número da versão]
Enter ".help" for usage hints.
sqlite>
```

[COPIAR CÓDIGO](#)

O SQLite está instalado e rodando. Caso contrário, você pode baixar o pacote de instalação para MacOs na [página de downloads](#) (<https://www.sqlite.org/download.html>) do SQLite.

## Instalação no Linux (Ubuntu)

Direto no terminal, digite o comando:

```
$ sudo apt update
```

[COPIAR CÓDIGO](#)

Em seguida, para instalar o SQLite:

```
$ sudo apt install sqlite3
```

[COPIAR CÓDIGO](#)

Confira a instalação com o comando:

```
$ sqlite3 --version
```

[COPIAR CÓDIGO](#)

Acesse o terminal do SQLite através do comando `sqlite3` . Você pode abrir diretamente um arquivo `.sqlite` ou `.db` com o comando `sqlite3 <caminho do arquivo> .`

---

## Começando do zero com o repositório inicial

Se preferir, você pode começar baixando o [repositório inicial \(<https://github.com/alura-cursos/1982-graphql>\)](https://github.com/alura-cursos/1982-graphql) que preparamos para este curso (na branch `config-inicial` ); o arquivo `database.db` já estará criado, e as tabelas populadas. Você só precisa:

1. Instalar em seu computador o SQLite e o dBeaver (instruções acima)
2. Instalar as dependências do projeto com `npm install`

```
3. Subir o servidor GraphQL com npm start e o
   json-server com npx json-server --watch
   ./api/data/dados.json
```

## Continuando do projeto do curso anterior

Se você vai continuar do projeto do curso anterior, precisa criar e popular o banco SQL que usaremos para expandir o projeto. Siga os passos abaixo.

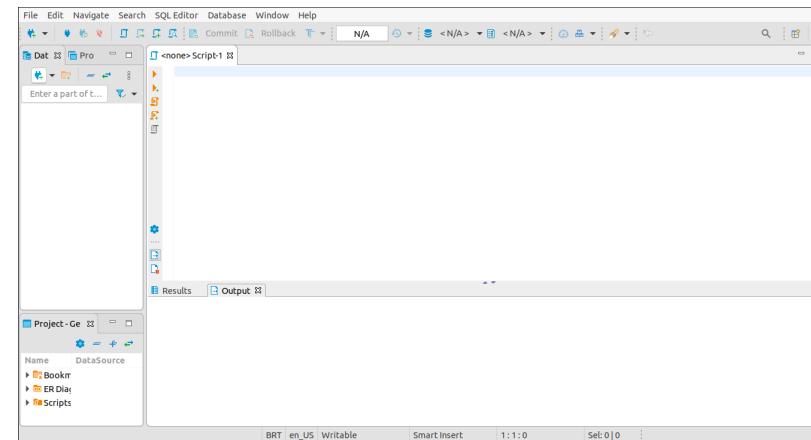
Se não quiser usar o dBeaver e preferir trabalhar com o SQLite diretamente no terminal, ou utilizando outro cliente, pode seguir apenas os passos 1 e 2; em seguida [acesse aqui \(https://github.com/alura-cursos/1982-graphql/blob/config-inicial/tabelas\\_sqlite.sql\)](https://github.com/alura-cursos/1982-graphql/blob/config-inicial/tabelas_sqlite.sql) o arquivo tabelas\_sqlite.sql com os scripts para criar e popular as tabelas que usaremos no curso.

1) Instale o SQLite no projeto com `npm install`

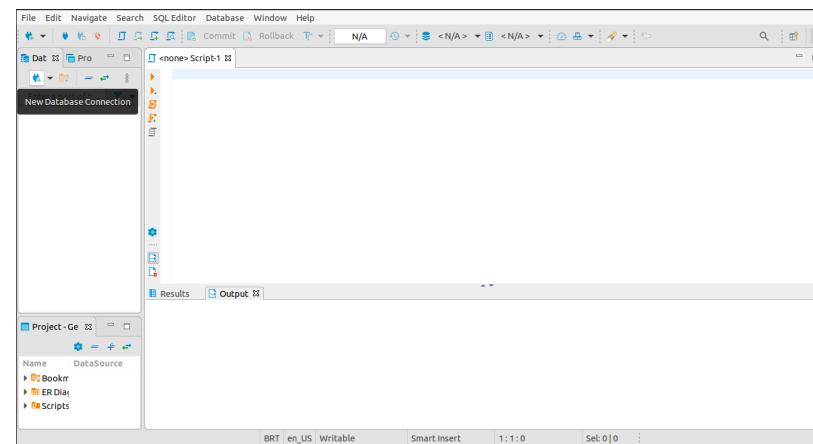
`sqlite3`

2) Crie um arquivo chamado `database.db` na pasta `./api/data` (mesma pasta onde está o JSON que usamos como base de dados para o tipo User)

3) Abra o dBeaver:



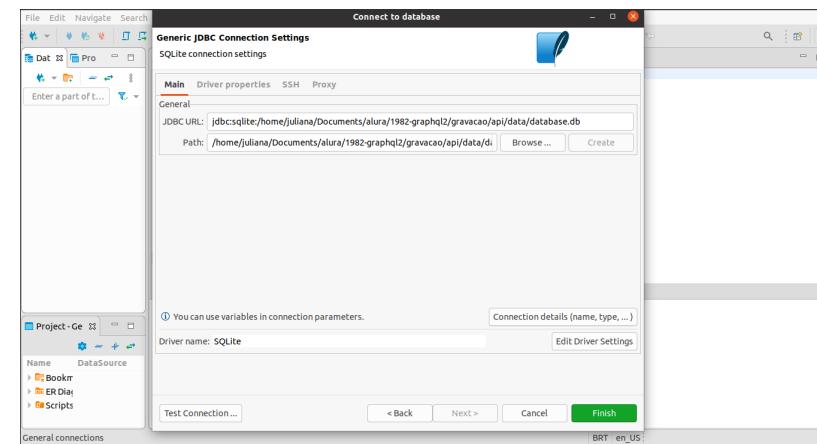
4) Clique em “new database connection” ou navegue pelo menu superior: Database > New Database Connection:



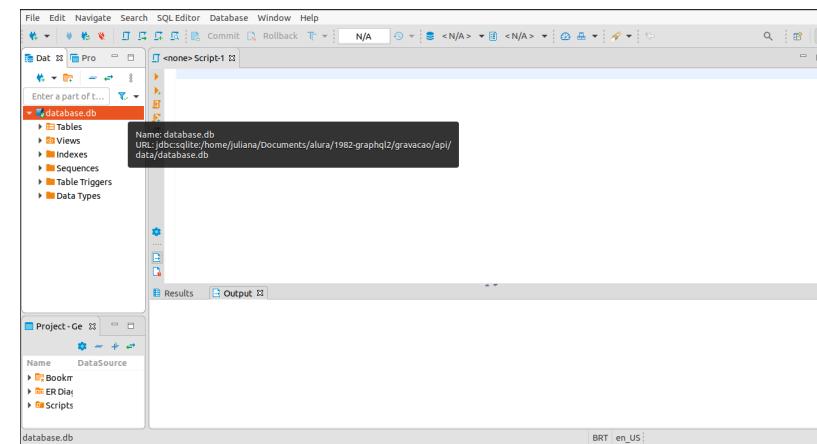
5) Selecione “SQLite” no menu de gerenciadores de SQL. Em seguida, clique no botão “next” na parte inferior do menu.



6) Na janela de menu “connection settings”, clique no botão “browse” e selecione o arquivo `database.db` na pasta do projeto.

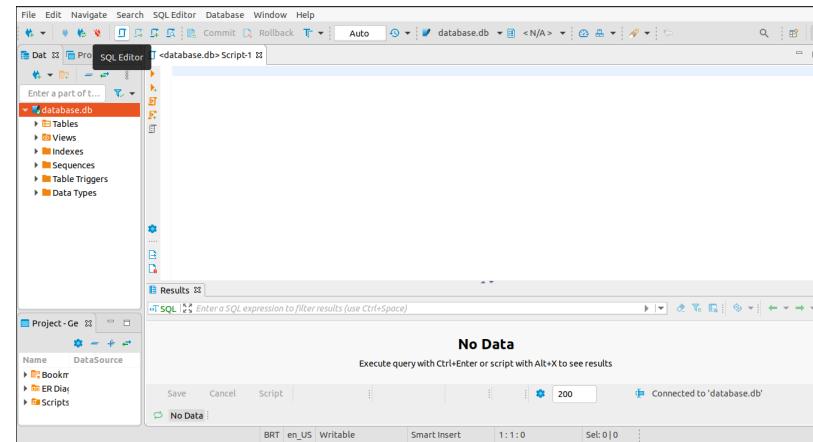


7) O arquivo `database.db` vai aparecer listado no menu à direita. Clique duas vezes em cima dele para abrir uma conexão, ou acesse no menu superior Database > Connect:



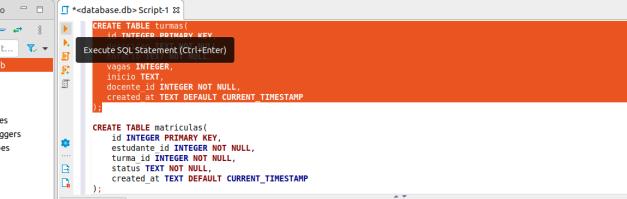
8) Clique em “SQL editor” para abrir uma nova janela do editor, ou acesse no menu superior: SQL Editor > New SQL Editor. Certifique-se que o

arquivo SQLite do projeto está conectado e selecionado. A aba do editor deve mencionar o nome do arquivo: <database.db>Script-1 (ou outro número de script).



9) Baixe e salve o arquivo `tabelas_sqlite.sql` que disponibilizamos [aqui](https://github.com/alura-cursos/1982-graphql/blob/config-inicial/tabelas_sqlite.sql) ([https://github.com/alura-cursos/1982-graphql/blob/config-inicial/tabelas\\_sqlite.sql](https://github.com/alura-cursos/1982-graphql/blob/config-inicial/tabelas_sqlite.sql)) usando o menu superior: SQL Editor > Import SQL Script > selecione o arquivo `tabelas_sqlite.sql` na pasta onde tiver salvo. Ou use o atalho `shift+ctrl+alt+0` para abrir o menu de importação.

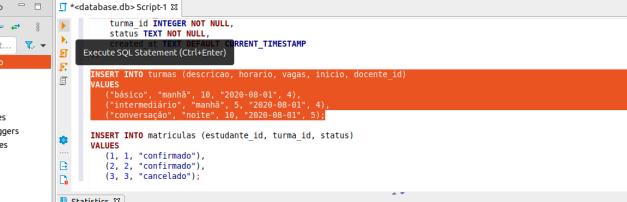
10) Selecione **somente** o statement CREATE TABLE turmas . Selecionar todas as 9 linhas, sem esquecer o ); . Clique em Execute SQL Statement (ou pressione ctrl+enter ) e confira a query gerada na aba inferior Statistics .



The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, Navigate, Search, SQL Editor, Database, Window, Help.
- Toolbars:** Standard toolbar with icons for New, Open, Save, Commit, Rollback, Auto, Undo, Redo, Database, <N/A>, and a magnifying glass.
- Left Panel:** Shows the current database is "database.db". It includes a search bar "Enter a part of..." and a tree view of the database structure: Tables, Views, indexes, Sequences, Table Triggers, and Data Types.
- Central Editor:** Displays two SQL scripts. The first script creates the "turmas" table with columns: id (INTEGER PRIMARY KEY), descricao (TEXT NOT NULL), horario (TEXT NOT NULL), vagas (INTEGER), and created\_at (TEXT DEFAULT CURRENT\_TIMESTAMP). The second script creates the "matriculas" table with columns: id (INTEGER PRIMARY KEY), estudiante\_id (INTEGER NOT NULL), turma\_id (INTEGER NOT NULL), status (TEXT NOT NULL), and created\_at (TEXT DEFAULT CURRENT\_TIMESTAMP).
- Bottom Bar:** Shows the current query is "CREATE TABLE turmas", the status is "Enter a SQL expression to filter results (use Ctrl+Space)", and the results table below.
- Results Table:** Shows the creation of the "turmas" table with 1 row affected, 200 rows updated, and 17ms execution time.

11) Selecione **somente** o statement `INSERT INTO turmas (todas as 5 linhas, sem esquecer o );`, clique em `Execute SQL Statement` (ou pressione `ctrl+enter`) e confira a query gerada na aba `Statistics`.



```
File Edit Navigate Search SQL Editor Database Window Help
Dat Commit Rollback Auto database.db <N/A> ...
Database.db
Tables Views Indexes Sequences Data Triggers Data Types
Enter a part of ...
Database.db
Tables Views Indexes Sequences Data Triggers Data Types
<database.db>
Script:1
  turma_id INTEGER NOT NULL,
  status TEXT NOT NULL,
  created_at CURRENT_TIMESTAMP
Execute SQL Statement (Ctrl+Enter)

INSERT INTO turmas (descricao, horario, vagas, inicio, docente_id)
VALUES
  ('básico', 'manhã', 10, '2020-08-01', 4),
  ('intermediário', 'manhã', 5, '2020-08-01', 4),
  ('conversão', 'noite', 10, '2020-08-01', 5);

INSERT INTO matriculas (estudante_id, turma_id, status)
VALUES
  (1, 1, 'confirmado'),
  (2, 2, 'confirmado'),
  (3, 3, 'cancelado');

Statistics
  INSERT INTO turmas (descricao, horario) Enter a SQL expression to filter results (use Ctrl+Space)
Name Value
Updated Rows 3
Query
  INSERT INTO turmas (descricao, horario, vagas, inicio, docente_id)
VALUES
  ('básico', 'manhã', 10, '2020-08-01', 4),
  ('intermediário', 'manhã', 5, '2020-08-01', 4),
  ('conversão', 'noite', 10, '2020-08-01', 5)
Finish time Fri Oct 16 15:53:40 BRT 2020
Save Cancel Script ...
BRT en_US Writable Smart Insert 19:1 [220] Sel: 220 | 5 3 row(s) updated - 10ms
```

12) Repita os passos 8 e 9 para os statements

```
CREATE TABLE matriculas e INSERT INTO  
matriculas .
```

13) As duas tabelas que usaremos no projeto já estão criadas e populadas com dados! Confira se deu tudo certo executando os statements `SELECT * FROM turmas;` e `SELECT * FROM matriculas;` em qualquer linha do editor SQL no dBeaver. Se quiser, pode criar mais registros também.

14) Se ainda não tiver feito isso, suba o servidor

GraphQL com `npm start` e o json-server com `npx json-server --watch ./api/data/dados.json`. Pode testar no playground em <http://localhost:4000/> (<http://localhost:4000/>) e conferir as queries que fizemos para Users no curso passado.

Tudo pronto para continuarmos!