

**Objetivos:**

- Desenvolver um servidor utilizando Node.js com o framework Express;
- Criar rotas HTTP para consulta de dados;
- Realizar acesso ao banco de dados PostgreSQL;
- Realizar requisições HTTP;
- Estilizar componentes com Styled Components;
- Implementar React Context para o gerenciamento de dados;
- Utilizar o hook useEffect para lidar com efeitos colaterais.

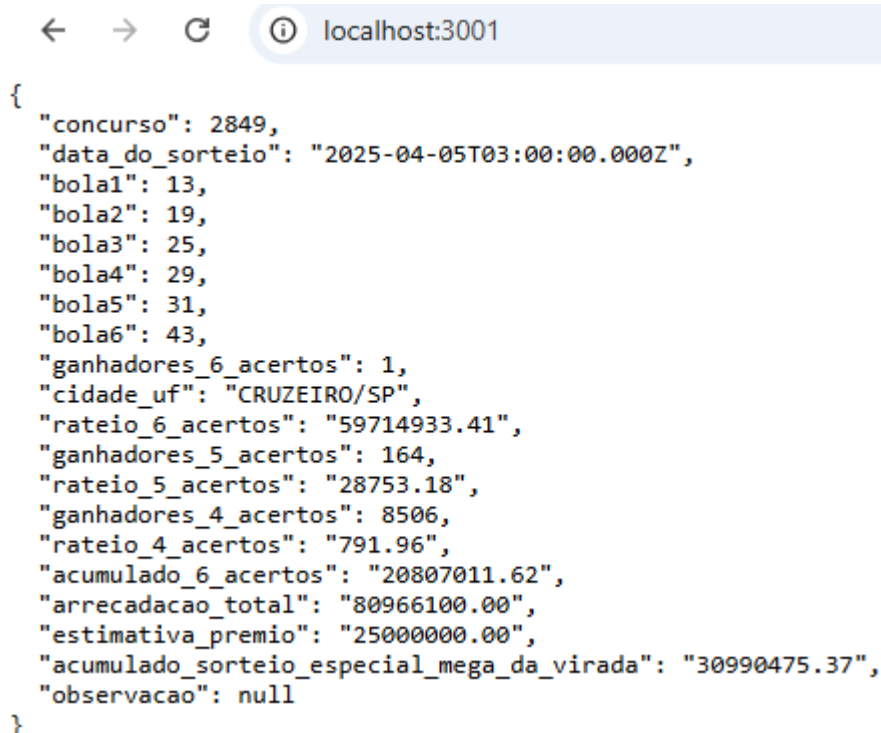
**Descrição da atividade:**

Desenvolver uma aplicação servidora que retorne os dados de todos os concursos da Mega-Sena, armazenados previamente em um banco de dados PostgreSQL. O acesso aos dados deverá ser integrado na aplicação front-end desenvolvida na atividade anterior.

- **Nota:** 0,25 pts. na média final
- **Data de entrega:** 21/mai
- **Forma de entrega:** individual e presencial (não será aceita entrega pelo Teams)

**Requisitos funcionais:**

- A aplicação servidora deverá disponibilizar as seguintes rotas:
  - Rota para retornar o concurso mais recente da Mega-Sena;



```
{
  "concurso": 2849,
  "data_do Sorteio": "2025-04-05T03:00:00.000Z",
  "bola1": 13,
  "bola2": 19,
  "bola3": 25,
  "bola4": 29,
  "bola5": 31,
  "bola6": 43,
  "ganhadores_6_acertos": 1,
  "cidade_uf": "CRUZEIRO/SP",
  "rateio_6_acertos": "59714933.41",
  "ganhadores_5_acertos": 164,
  "rateio_5_acertos": "28753.18",
  "ganhadores_4_acertos": 8506,
  "rateio_4_acertos": "791.96",
  "acumulado_6_acertos": "20807011.62",
  "arrecadacao_total": "80966100.00",
  "estimativa_premio": "25000000.00",
  "acumulado Sorteio Especial Mega da Virada": "30990475.37",
  "observacao": null
}
```

- Rota para retornar os dados de um concurso específico, identificado pelo número.

← → ↻ ⓘ localhost:3001/2848

```
{
  "concurso": 2848,
  "data_do Sorteio": "2025-04-03T03:00:00.000Z",
  "bola1": 5,
  "bola2": 14,
  "bola3": 19,
  "bola4": 29,
  "bola5": 30,
  "bola6": 54,
  "ganhadores_6_acertos": 0,
  "cidade_uf": null,
  "rateio_6_acertos": "0.00",
  "ganhadores_5_acertos": 73,
  "rateio_5_acertos": "50494.97",
  "ganhadores_4_acertos": 5003,
  "rateio_4_acertos": "1052.54",
  "acumulado_6_acertos": "51028444.89",
  "arrecadacao_total": "63291355.00",
  "estimativa_premio": "60000000.00",
  "acumulado Sorteio Especial Mega da Virada": "29749548.43",
  "observacao": null
}
```

A rota deverá retornar uma mensagem se o concurso não existir.

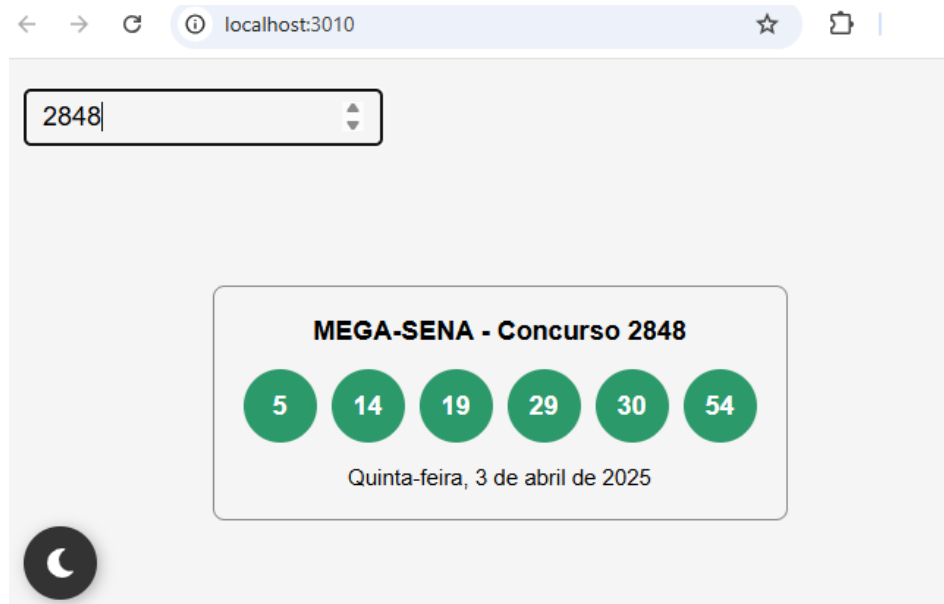
← → ↻ ⓘ localhost:3001/3000

```
{
  "message": "Não existem dados do concurso 3000"
}
```

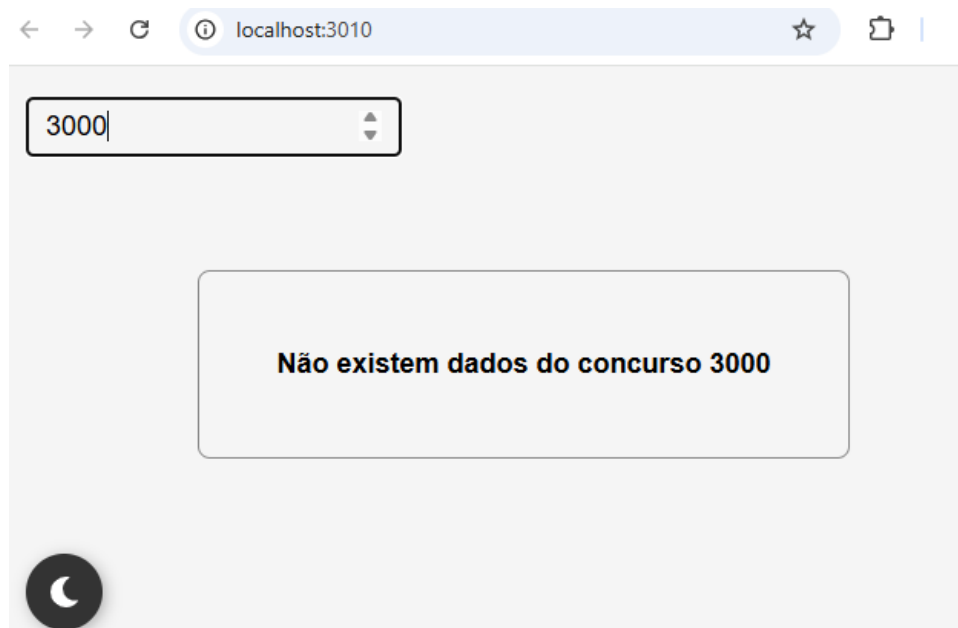
- ii. A aplicação front-end deverá ser inicializada exibindo o concurso mais atual. O concurso deverá ser obtido no servidor.



- iii. A aplicação front-end deverá ter um campo para o usuário fornecer o número do concurso desejado. No exemplo a seguir o usuário forneceu o número de concurso 2848.



- iv. A aplicação front-end deverá exibir uma mensagem quando não existir o concurso fornecido pelo usuário. No exemplo a seguir o usuário forneceu o número de concurso 3000.



**Requisitos não-funcionais:**

- A aplicação servidora deverá ser executada na porta 3001;
- Os parâmetros de conexão com o BD devem ser definidos estar no arquivo `.env`.

**Passos para implementação da aplicação servidora**

**Passo 1 – Criar o BD no PostgreSQL e carregar os dados**

- No pgAdmin, crie um BD de nome `bdaula` ou outro nome de sua preferência;
- No pgAdmin, crie a tabela `megasena` executando o seguinte comando SQL:

```
DROP TABLE IF EXISTS megasena;  
CREATE TABLE megasena (  
    concurso INTEGER NOT NULL,  
    data_do Sorteio DATE NOT NULL,  
    bola1 INTEGER NOT NULL,  
    bola2 INTEGER NOT NULL,  
    bola3 INTEGER NOT NULL,  
    bola4 INTEGER NOT NULL,  
    bola5 INTEGER NOT NULL,  
    bola6 INTEGER NOT NULL,  
    ganhadores_6_acertos INTEGER NOT NULL,  
    cidade_uf VARCHAR(510) NULL,  
    rateio_6_acertos DECIMAL NOT NULL,  
    ganhadores_5_acertos INTEGER NOT NULL,  
    rateio_5_acertos DECIMAL NOT NULL,  
    ganhadores_4_acertos INTEGER NOT NULL,  
    rateio_4_acertos DECIMAL NOT NULL,  
    acumulado_6_acertos DECIMAL NOT NULL,  
    arrecadacao_total DECIMAL NOT NULL,  
    estimativa_premio DECIMAL NOT NULL,  
    acumulado Sorteio Especial mega da virada DECIMAL NOT NULL,  
    observacao VARCHAR(255) NULL,  
    PRIMARY KEY(concurso)  
);
```

- c) No pgAdmin, carregue os dados do arquivo `megasena.csv` com o seguinte comando SQL (ajuste o caminho do arquivo conforme o seu sistema):

```
COPY megasena  
FROM 'D:\pasta\pasta/megasena.csv'  
WITH (  
    FORMAT csv,  
    DELIMITER ';',  
    HEADER,  
    NULL 'NULL'  
);
```

O arquivo `megasena.csv` foi obtido a partir do site oficial da Caixa Econômica Federal (<https://loterias.caixa.gov.br/Paginas/Mega-Sena.aspx>) e foi adaptado para estar no formato adequado de importação para o PostgreSQL.

- d) No pgAdmin, verifique se os dados foram corretamente importados com o comando:

```
select * from megasena;
```

## Passo 2 – Criar e configurar o servidor Node.js com Express

- a) Crie uma pasta de nome `server` ou outro nome de sua preferência;  
b) No terminal, execute o seguinte comando para iniciar um projeto Node.js:

```
npm init -y
```

- c) Instale as dependências necessárias:

```
npm i express dotenv cors pg
```

- d) Instale as dependências de desenvolvimento:

```
npm i ts-node ts-node-dev @types/express @types/cors @types/pg -D
```

- e) Inicialize o arquivo de configuração do TypeScript:

```
npx tsc --init
```

- f) Crie o arquivo `.env` na raiz do projeto e defina as seguintes variáveis de ambiente:

```
PORT = 3001
BD_HOST = localhost
BD_USER = postgres # Altere conforme o seu usuário
BD_PASSWORD = 123 # Altere conforme a sua senha
BD_DATABASE = bdaula # Nome do banco criado no pgAdmin
BD_PORT = 5432 # Porta padrão do PostgreSQL
```

- g) Crie a pasta `src` e, dentro dela, o arquivo `index.ts` (inicialmente pode deixá-lo vazio);

- h) Adicione os seguintes scripts no arquivo `package.json` para facilitar a execução do projeto:

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

- i) Crie o arquivo `db.ts` na pasta `src/controllers` com o código responsável por configurar e exportar a conexão com o BD:

```
import { Pool } from "pg";
import dotenv from "dotenv";

// Carrega as variáveis de ambiente do arquivo .env
dotenv.config();

export default new Pool({
  host: process.env.BD_HOST,
  user: process.env.BD_USER,
  password: process.env.BD_PASSWORD,
  database: process.env.BD_DATABASE,
  port: Number(process.env.BD_PORT)
});
```