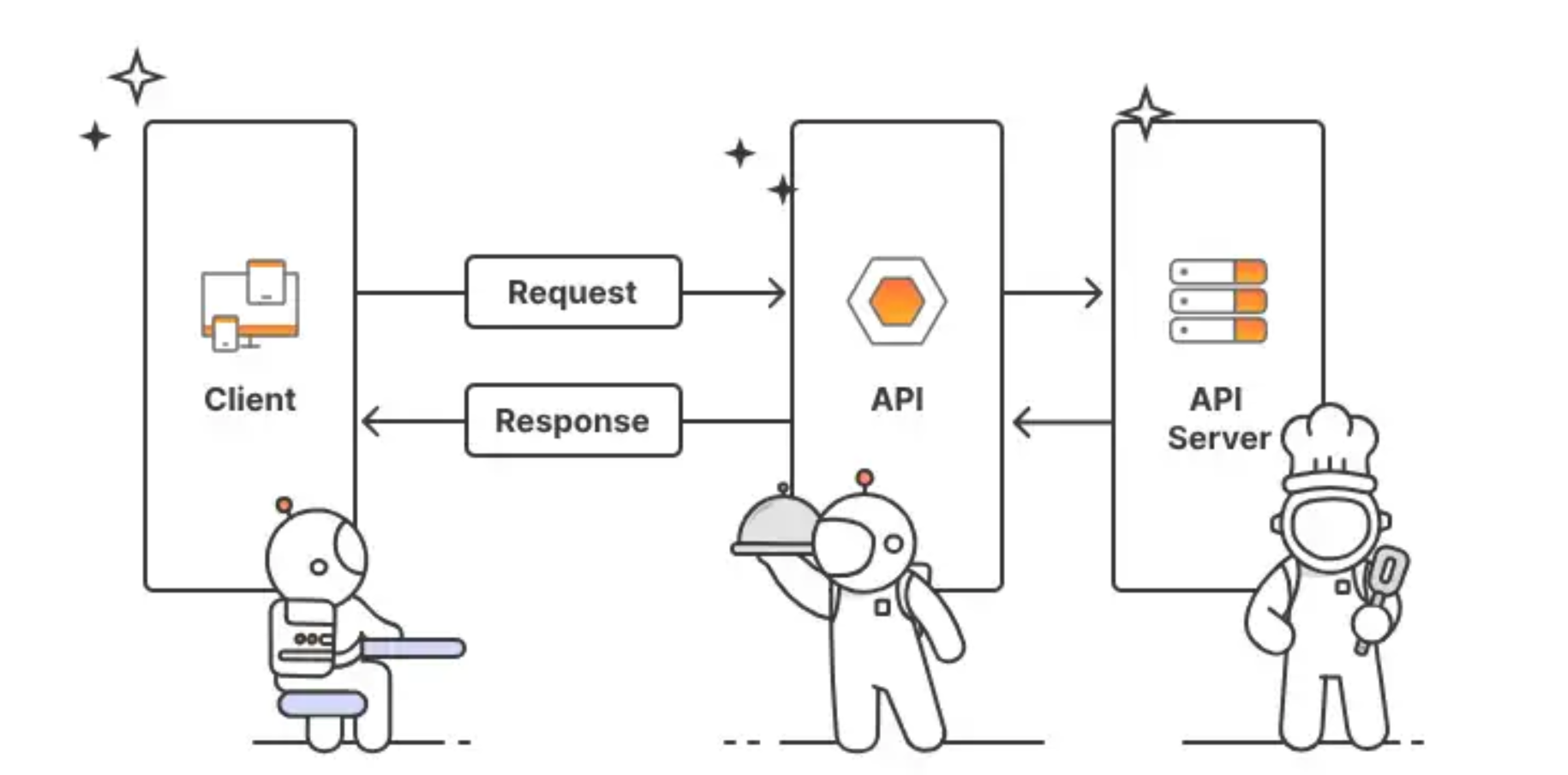
Material Didático: Requisições HTTP e APIs com Express.js

# 1. O que é uma API?

API (Interface de Programação de Aplicações) é um conjunto de regras que permite que diferentes sistemas de software se comuniquem entre si. No caso de APIs web, elas permitem que clientes (como navegadores ou aplicativos móveis) se conectem a servidores para buscar ou enviar dados.



# 2. O que é o Express.js?

O Express.js é um framework leve para Node.js que facilita a criação de servidores web e APIs. Ele ajuda a organizar melhor o código e criar rotas para diferentes funcionalidades.

Obs.: entenderemos mais sobre um que é um framework mais para frente, apenas devemos entender que um framework não adiciona funcionalidades não existentes no nosso código, apenas “facilita” e padroniza a forma que construímos uma API em node.

# 3. Entendendo o Protocolo HTTP

HTTP (HyperText Transfer Protocol) é o protocolo que regula como as requisições e respostas são feitas na web. Ele define métodos que usamos para interagir com o servidor. Os principais métodos são:

- GET: Busca informações do servidor.

- POST: Envia dados para o servidor (geralmente para criar novos recursos).

- PUT: Atualiza recursos no servidor.

- DELETE: Remove recursos do servidor.

# 4. Métodos HTTP com Express.js

|  |  |  |
| --- | --- | --- |
| Método | Finalidade | Exemplo no Express |
| GET | Obter dados do servidor (consultar) | app.get('/users', (req, res) => {...}) |
| POST | Enviar dados para criar algo | app.post('/users', (req, res) => {...}) |
| PUT | Atualizar um recurso existente | app.put('/users/:id', (req, res) => {...}) |
| DELETE | Excluir um recurso | app.delete('/users/:id', (req, res) => {...}) |

# 5. Exemplo Prático: Criando uma API Simples

## 5.1 Configuração do Projeto

1. Criar um diretório de projeto:  
 mkdir api-express (ou outro nome para sua pasta)  
 cd api-express

2. Inicializar o projeto Node.js:  
 npm init -y

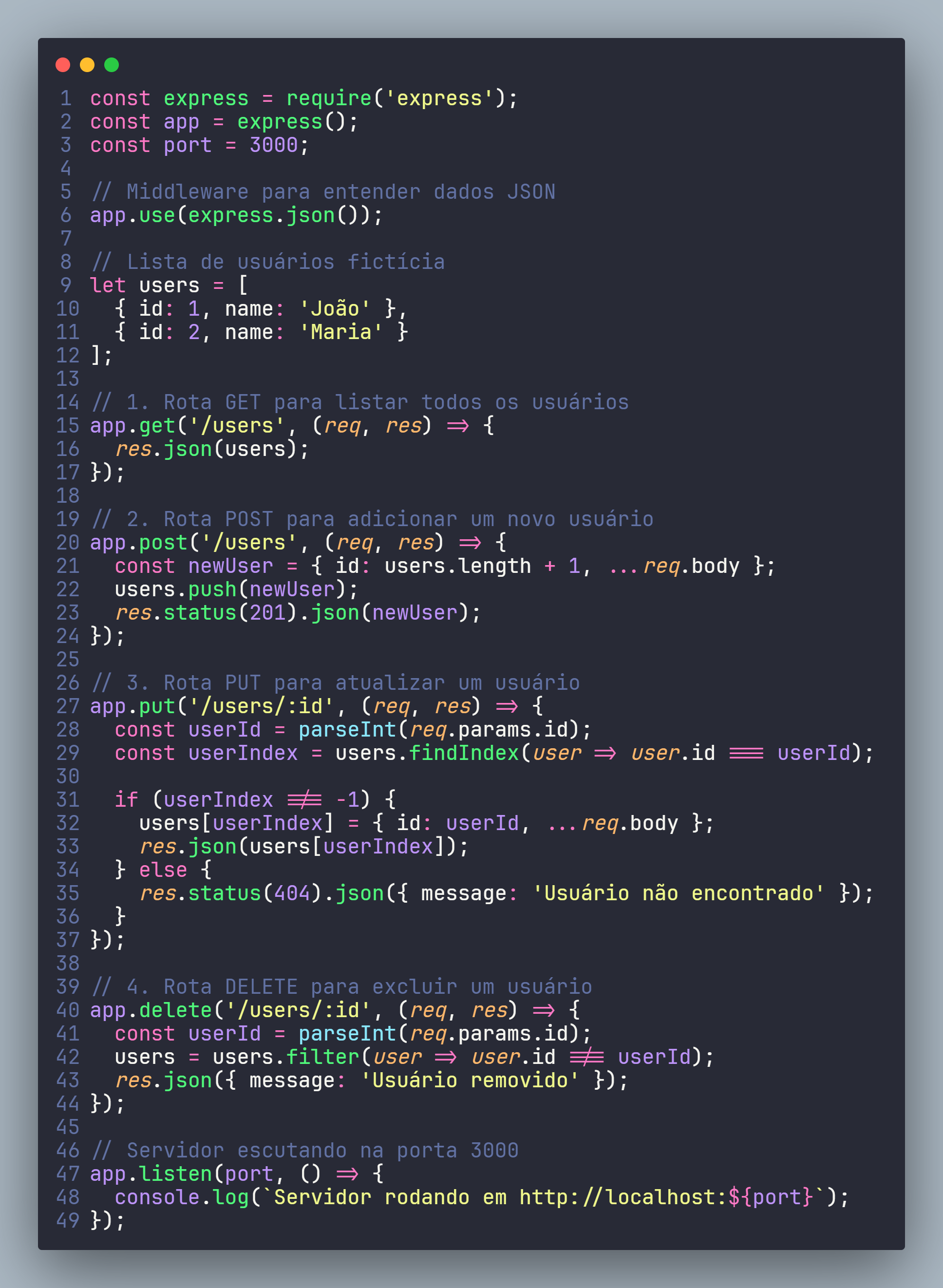
3. Instalar o Express.js:  
 npm install express

## 5.2 Estrutura Básica do Servidor

Criar o arquivo `server.js` e criar uma api (sem funções)



Como adicionar as rotas básicas, como GET, POST, PUT e DELETE.



## 5.3 Explicando o Código

• Rota GET: Retorna todos os usuários na lista. Ideal para listar ou consultar dados.  
• Rota POST: Adiciona um novo usuário. No corpo da requisição (request body), enviamos os dados do novo usuário.  
• Rota PUT: Atualiza os dados de um usuário existente. O ID do usuário é passado como parte da URL.  
• Rota DELETE: Remove um usuário com base no ID fornecido na URL.

## 5.3 O que é um CRUD?

CRUD é um acrônimo que representa as quatro operações básicas utilizadas em bancos de dados e APIs:

* **C**reate (Criar): Inserir ou adicionar novos dados no sistema.
* **R**ead (Ler): Recuperar ou consultar dados existentes.
* **U**pdate (Atualizar): Modificar ou atualizar dados existentes.
* **D**elete (Deletar): Remover dados do sistema.

Essas operações são comuns em qualquer aplicação que lida com persistência de dados e são fundamentais para a criação de sistemas dinâmicos que interagem com bases de dados.

# 6. Testando a API

Para testar nossa API, podemos usar ferramentas como Postman ou Insomnia, mas também podemos testar usando o curl no terminal:

1. Testar GET:  
 curl http://localhost:3000/users

2. Testar POST (Adicionando um usuário):  
 curl -X POST -H "Content-Type: application/json" -d '{"name": "Carlos"}' http://localhost:3000/users

3. Testar PUT (Atualizando um usuário):  
 curl -X PUT -H "Content-Type: application/json" -d '{"name": "Ana"}' http://localhost:3000/users/1

4. Testar DELETE (Remover um usuário):  
 curl -X DELETE http://localhost:3000/users/1

# 7. Resumo

- GET: Usado para buscar recursos.  
- POST: Usado para criar novos recursos.  
- PUT: Usado para atualizar recursos.  
- DELETE: Usado para deletar recursos.  
O Express.js facilita a criação de APIs organizadas e com rotas claras.

# 8. Tarefa

1. Crie uma rota GET para um novo recurso, como /products.
   1. Quando o cliente faz uma requisição GET para /products, o servidor precisa consultar os dados disponíveis (que podem estar em um banco de dados ou em uma variável) e devolvê-los no formato correto (geralmente em JSON).
2. Implemente uma rota POST para adicionar novos produtos.
   1. Quando o cliente faz uma requisição POST para /products, o servidor precisa ler os dados enviados no corpo da requisição (geralmente no formato JSON).
   2. Em seguida, ele adiciona esse novo produto à lista de produtos e confirma que a adição foi realizada.
3. Implemente uma rota PUT para atualizar um produto específico.
   1. Quando o cliente faz uma requisição PUT para /products/:id, o servidor precisa primeiro identificar qual produto está sendo atualizado (pelo ID).
   2. Depois, ele lê os novos dados fornecidos no corpo da requisição e atualiza o produto correspondente.
4. Teste todas as rotas usando o Postman ou outra ferramenta de sua escolha.
   1. Para testar o GET, você fará uma requisição para /products e verá se a lista de produtos é exibida corretamente.
   2. Para testar o POST, você enviará um novo produto no corpo da requisição para /products e verá se ele é adicionado à lista.
   3. Para testar o PUT, você enviará um ID de produto específico junto com as informações atualizadas no corpo da requisição e verificará se o produto foi alterado.

