
**UNIVERSIDADE SALVADOR – CAMPUS PROFESSOR BARROS**
**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**CAUAN BRANDÃO SILVA LEITE**
**(RA: 1272225908)**

**EGRINALDO FERREIRA DE CERQUEIRA JUNIOR**
**(RA: 1272226313)**

**GABRIEL VITOR CEDRAZ CARNEIRO**
**(RA: 12723118390)**

**HENRIQUE CAVALCANTI ROCHA**
**(RA: 12722117519)**

**JACKSON REIS SILVA SANTOS**
**(RA: 1272223855)**

**RINALDO RABELO DA SILVA JUNIOR**
**(RA: 1272220875)**

# ATIVIDADE AVALIATIVA 3

# DOCUMENTAÇÃO DO PROJETO

Salvador - BA

**CAUAN BRANDÃO SILVA LEITE**
**(RA: 1272225908)**

**EGRINALDO FERREIRA DE CERQUEIRA JUNIOR**
**(RA: 1272226313)**

**GABRIEL VITOR CEDRAZ CARNEIRO**
**(RA: 12723118390)**

**HENRIQUE CAVALCANTI ROCHA**
**(RA: 12722117519)**

**JACKSON REIS SILVA SANTOS**
**(RA: 1272223855)**

**RINALDO RABELO DA SILVA JUNIOR**
**(RA: 1272220875)**

# ATIVIDADE AVALIATIVA 3

# DOCUMENTAÇÃO DO PROJETO

Atividade avaliativa da unidade curricular Sistemas distribuídos e Mobile da Universidade Salvador – UNIFACS, campus Professor Barros.

Docente: Adailton de Jesus Cerqueira Junior

Salvador - BA

2023

**SUMÁRIO**

## 1. DESCRIÇÃO DOS REQUERIMENTOS DE SOFTWARE PARA EXECUTAR A APLICAÇÃO.

Para realizar o projeto, escolhemos implementar a solução utilizando API.

Toda a nossa API é feita baseada na arquitetura REST utilizando de comunicações HTTP para troca e envio de dados utilizando JSON. Desenvolvemos a API com Javascript, utilizando o framework Express para criar o servidor HTTP, express validator para validação de inputs de entrada do usuário, prisma para criação e consultas com o banco de dados, nodemon para refresh automático baseado nas alterações e o Swagger para documentação técnica da nossa API.
Para o banco de dados, utilizamos o SQLite. Toda nossa aplicação roda na porta 3000 e corresponde as chamadas no endereço: http://localhost:3000
Para conseguir rodar o projeto, é importante e fundamental seguir o seguinte passo a passo:

1 - Com o projeto clonado e aberto no editor de código de sua preferência, no terminal, executar o comando “npm install” para instalar as dependências do projeto.

2 – Executar o comando “npx prisma migrate dev” para sincronizar o banco de dados com as migrações realizadas até o seguinte momento da entrega do projeto.

3 - Executar o comando “npm run dev” para rodar o servidor, uma mensagem será exibida no terminal para indicar que a aplicação está ativa.

4 - Se desejar, rodar o comando “npx prisma studio” para abrir um mini gerenciador do banco de dados no browser, somente para acompanhamento dos dados.

IMPORTANTE: Como extra, fizemos uma documentação técnica com a ajuda de uma blibioteca chamada Swagger, ela é bem utilizada no mercado atual e é muito intuitivo e fácil utilizá-la. Ela mostra como pode ser feitos as chamadas aos respectivos endpoints e o que se espera receber no objeto de entrada, e bem como os retornos possíveis para o objeto de saída. Para ter acesso a essa documentação, basta digitar no browser de sua preferência o seguinte endereço: http://localhost:3000/api-docs (Com a aplicação rodando).

## 2. JUSTIFICATIVA DA ESCOLHA DA TECNOLOGIA

Optamos por integrar uma API com JavaScript em nosso projeto acadêmico devido à eficiência assíncrona da linguagem, facilitando interações suaves. A ampla adoção do JavaScript e suas bibliotecas simplificam o desenvolvimento, promovendo modularidade e manutenção fácil. A portabilidade do JavaScript amplia nosso alcance, enquanto a escalabilidade é assegurada pela solidez do ecossistema. Essa escolha estratégica visa garantir eficácia, usabilidade e preparação para futuras expansões. Também utilizando o NodeJS que é responsável por executar o servidor HTTP que hospeda a API, permitindo a comunicação entre o cliente e o servidor por meio de solicitações HTTP.

## 3. APRESENTAÇÃO E DETALHAMENTO SOBRE A ARQUITETURA, ESTRATÉGIA E ALTORITMOS.

Basicamente, utilizamos na nossa API uma estrutura de pastas bastante similar com o modelo de arquitetura MVC, onde na nossa pasta de Controllers temos a responsabilidade de receber as requisições vinda do usuário e dar um retorno baseado com as validações.

A camada de Service é responsável por chamar a camada de repository para acesso com banco de dados. Além disso, ela é responsável por tratamento de dados, e é nela que está inserida toda a regra de negócio por dentro da aplicação, como um exemplo prático: Nós não podemos deixar a venda acontecer se o produto não tem estoque ou está com o estoque zerado, com isso, a camada de service consegue validar isso antes de realizar a venda.

A camada de Repository é responsável pelo acesso e manipulação com o banco de dados. Para esse projeto, utilizamos o prisma, uma ORM bastante prática e fácil para acesso, utilizando a linguaguem do javascript nativo para fazer queries ao banco de dados.

Como uma melhoria, implementamos mais uma camada adicional. A camada de Validators, essa camada na API é responsável por implementar um middleware antes da requisição chegar na controller de fato, para validar os dados de input do usuário. É importante essa camada para garantir que os dados que estejam chegando a minha camada de controller esteja correto e conciso com a documentação apresentada.

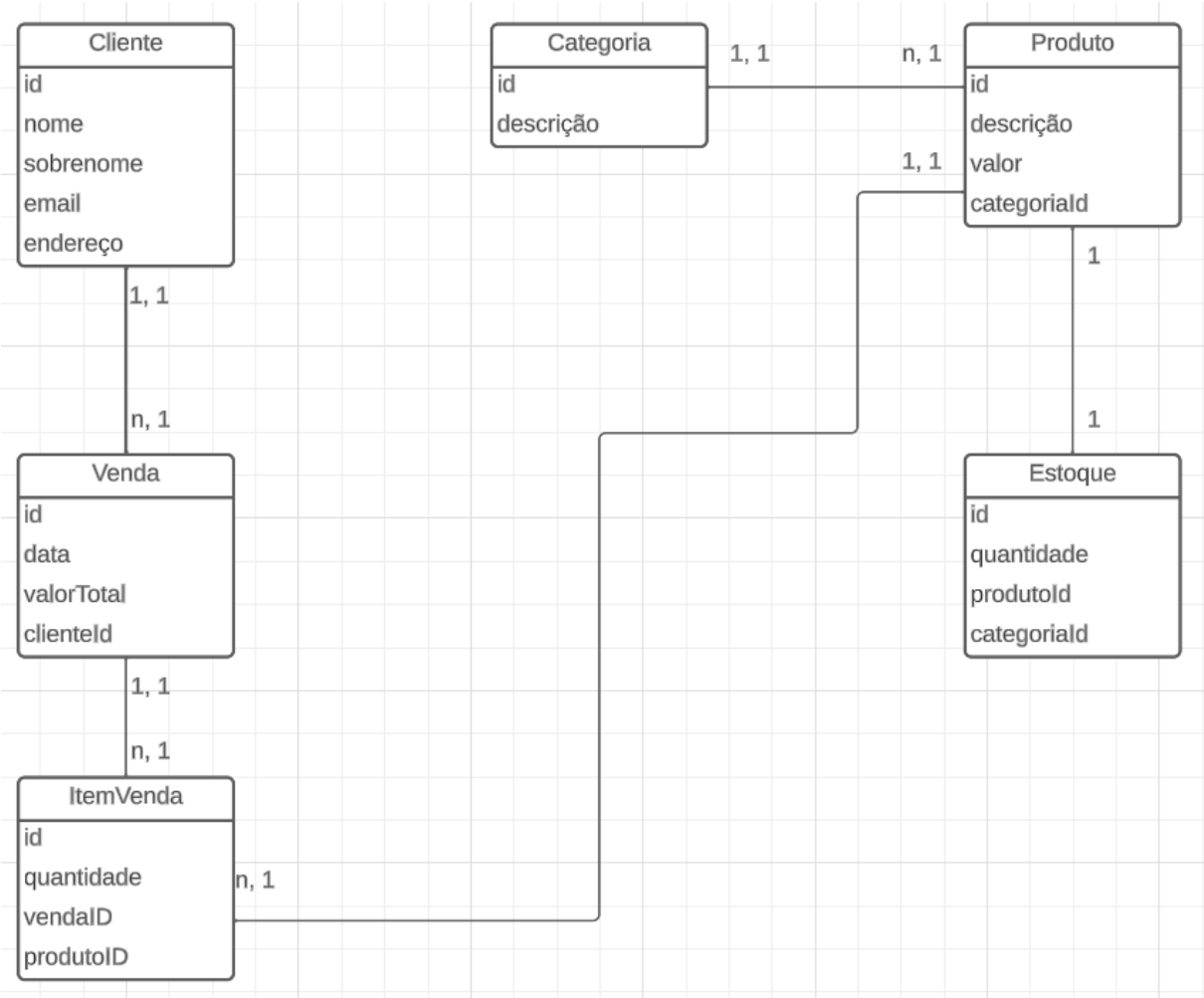
## 4. PROJETO DE IMPLEMENTAÇÃO

Para desenvolvermos a aplicação nos reunimos e criamos a UML para estruturar os comportamentos e interações das entidades, visando facilitar a compreensão do desenvolvimento, reduzir erros e melhorar a produtividade.

Os Relacionamentos da UML são:

- Cliente: 1:N com Venda (um cliente pode ter várias vendas)
- Categoria: 1:N com Produto (uma categoria pode ter vários produtos)
- Produto: 1:1 com Categoria (um produto pertence a uma categoria)
  1:N com ItemVenda (um produto pode estar em vários itens de venda)
- Estoque: 1:1 com Produto (um estoque pertence a um produto)
- Venda:1:1 com Cliente (uma venda pertence a um cliente)
  1:N com ItemVenda (uma venda pode ter vários itens de venda)
- ItemVenda: 1:1 com Venda (um item de venda pertence a uma venda)
  1:1 com Produto (um item de venda se refere a um produto)

Imagem: UML do sistema (desenvolvido no Lucidchart)




Autoria própria

**REFERÊNCIAS:**


- https://www.lucidchart.com
- https://coodesh.com
- https://www.prisma.io/
- https://swagger.io