Tipo: Trabalho em grupo (mínimo 2 – máximo 4 estudantes, grupo deve ser o mesmo na N2 e N3)
Ferramentas: Node.js, Mocha, Chai, Sinon, Chai-http, Insomnia

Tema: Projeto autoral baseado em APIs https://jsonplaceholder.typicode.com/ e https://my-json-server.typicode.com/

Objetivos

- Desenvolver um projeto Node.js **do zero**, com estrutura de pastas definida.
- Criar funções simples no código-fonte (não é necessário que o software seja funcional) que permitam a aplicação de diferentes tipos de testes.
- Aplicar testes **unitários, de API e de integração** com Mocha + Chai + Sinon + Chai-http.
- Compreender e demonstrar o uso de **assert, expect e should** no Mocha/Chai.
- Progredir em etapas de complexidade crescente, de acordo com a faixa de nota desejada (6–10).

Estrutura mínima do projeto

- src/ → código-fonte (funções a serem testadas)
- test/ → arquivos de testes
- package.json configurado para executar testes com npm test
- **Importante:** o projeto não deve usar módulos ES (usar require/module.exports)

Definição do tema do projeto

O **JSONPlaceholder** e o **My JSON Server** simulam endpoints típicos de um **mini-sistema de rede social/blog**.
Os endpoints mais comuns são:

- /posts → publicações (artigos, mensagens, posts)
- /comments → comentários em posts

- /users → usuários do sistema
- /albums → álbuns de fotos
- /photos → fotos de cada álbum
- /todos → tarefas pendentes

T**ema-base do projeto**:

“Miniaplicativo de Rede Social / Blog”, com funções para manipular **usuários, posts, comentários, álbuns, fotos e tarefas**.

Exemplo de funções esperadas em src/:

- src/users.js → funções para buscar/filtrar usuários.
- src/posts.js → funções para listar posts, criar post fake, associar post a usuário.
- src/comments.js → funções para listar comentários de um post.
- src/todos.js → funções para marcar tarefas como concluídas.

Critérios de avaliação por faixa de nota

Nota 6

Entregas obrigatórias:

- Projeto criado e estruturado (src/ e test/).
- Funções implementadas com tema da rede social/blog.
- Testes unitários:
    - **mínimo 5 asserts diferentes** (não apenas strictEqual → usar também deepEqual, throws, doesNotThrow, match, etc.).
    - **mínimo 5 expects diferentes** (não apenas .to.equal → usar .to.have.property, .to.be.a, .to.contain, .to.have.lengthOf, etc.).
    - **mínimo 5 should diferentes**.
    - Total mínimo: **20 testes unitários** só nessa parte.

- Testes mockados/stubados de API com Sinon (mínimo **5 testes**).

Avaliação:

- Abrir projeto no VS Code.
- Rodar npm test e todos os testes passarem
- Conferir variações de asserts/expect/should.

Nota 7

Além da nota 6:

- Criar **apresentação PowerPoint** + versão em PDF contendo:
    - Cada assert, expect e should utilizados.
    - Explicação objetiva de funcionamento de cada um.
    - Bloco de código do projeto mostrando seu uso.

Avaliação:

- Conferir correspondência entre apresentação e testes reais do projeto.
- **Formato de entrega:** .pptx + .pdf.

Nota 8

Além da nota 7:

- Criar **novo arquivo de testes de API** com chamadas **reais** ao https://jsonplaceholder.typicode.com/
- Implementar **todos os endpoints disponíveis** no site.
- Criar **vídeo (YouTube ou OneDrive)** contendo:
    - Explicação **função por função de teste** (unitários, mockados, APIs).
    - Explicação **linha por linha** de cada teste.
    - Demonstração no código-fonte de qual função é testada.

Ao apresentar cada função de teste (unitário, mockado/stub, API), o vídeo deve também mostrar a função correspondente no código-fonte testado e explicá-la.

- Para **cada teste**:
    1. Mostrar o nome do teste e explicar linha a linha o que ele verifica.
    2. Navegar no editor até a função real em src/ que é alvo do teste.

3. Explicar a função de produção (o que faz, entradas/saídas, casos contemplados).
4. Deixar claro como o teste valida o comportamento dessa função (asserções usadas e por quê).

Avaliação:

- Rodar npm test → todos os testes devem passar.
- Conferir se todos os endpoints do JSONPlaceholder foram testados.
- Conferir link de vídeo.
- Estimativa: **+20 ou mais funções de teste adicionais** em relação à nota 6.
- Tempo médio esperado de vídeo: **35–50 min**, considerando ~1–2 min por teste.

Nota 9

Além da nota 8:

- Criar **novo arquivo de testes de integração** com https://my-json-server.typicode.com/
- Testar endpoints definidos no arquivo db.json (versão gratuita/sem login).

Avaliação:

- Rodar npm test → todos os testes devem passar.
- Conferir se os testes cobrem corretamente os endpoints simulados.
- Estimativa: **+10 a 15 funções de teste adicionais** em relação à nota 8.

Nota 10

Além da nota 9:

- Utilizar **Insomnia** para executar os testes de API (JSONPlaceholder) e integração (My JSON Server).
- Criar **vídeo explicativo** contendo:
    - Demonstração de cada teste executado no Insomnia.
    - Explicação do que é um **header** (mostrando elementos comuns).
    - Explicação do que retorna no **body**.

    - Comparação com as funções de teste criadas no projeto (nota 8 e 9).

Além de demonstrar a execução dos testes de API/integração no Insomnia, para cada cenário exibido deve haver:

- referência ao arquivo de teste correspondente no projeto (onde aquele cenário está automatizado) e
- referência à função de produção em src/ exercitada pela chamada (ex.: o handler/serviço que monta a requisição/trata a resposta), com uma breve explicação do papel dessa função no fluxo.

Forma de Entrega

- Projeto completo compactado (.zip) com todas as dependências (node_modules incluído).
- Materiais adicionais conforme faixa escolhida (pptx, pdf, vídeos via link).
- Enviar via plataforma TEAMS na atividade N2 respeitando a data e horário de entrega