## Bootcamp IGTI: Desenvolvedor(a) NODE.JS

# Trabalho Prático

| Módulo 4 | Tópicos Especiais Desenvolvimento Back End |
|---|---|

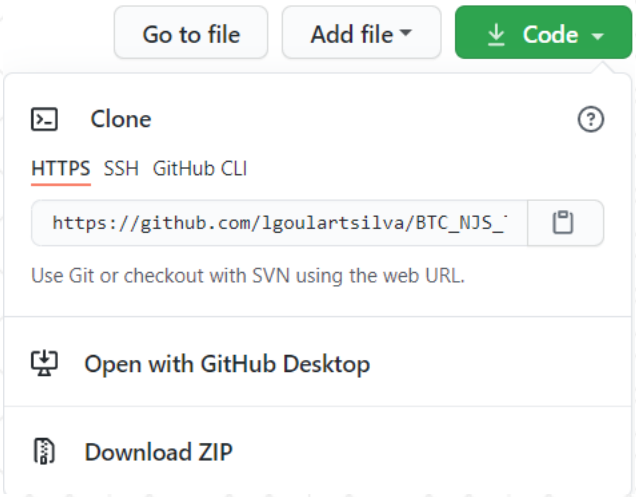
## Objetivos

Exercitar os seguintes conceitos trabalhados no Módulo:

- ✓ Reforçar os conceitos teóricos sobre o TDD.
- ✓ Reforçar os conceitos teóricos sobre a Pirâmide de Testes.
- ✓ Testes unitários utilizando a biblioteca Jest.
- ✓ Testes de integração utilizando as bibliotecas Jest e supertest.

## Enunciado

Deverá ser realizado o download do código fonte dos três projetos utilizados nas aulas práticas, disponível em https://github.com/lgoulartsilva/BTC_NJS_TA_PT1. A princípio, sinta-se à vontade para realizar o **download baixando o arquivo .zip**, ou **clonar o repositório utilizando o git** – caso não conheça o git, não se preocupe, pois em breve o assunto será abordado em nosso curso.




Após realizar o download será necessário realizar a instalação de cada um dos três projetos. Para isso é utilizado o comando **'npm install'.**

Após realizada a configuração inicial, siga as atividades propostas e observe o comportamento esperado para cada caso. Responda, então, às questões propostas de acordo com o comportamento observado.

## Atividades

Os alunos deverão desempenhar as seguintes atividades:

1. **ATIVIDADE 01:** para cada um dos três projetos, realize a execução dos testes com o script configurado no **package.json** do mesmo. Observe o comportamento para cada caso.

2. **ATIVIDADE 02:** seguindo o processo de TDD (ciclo vermelho-verde-refatora), a aplicação '01-calcula-valor' deverá ser alterada:
    a. Você deve criar funções de testes para cobrir os casos em que os parâmetros de entrada da função **calcularPrestacoes** são inválidos. Caso uma das regras, descritas a seguir, não seja atendida, a função deverá retornar um array vazio:
        i. **montante**: deve ser um número maior que 0.
        ii. **numeroParcelas:** deve ser um número inteiro maior que 0.

b. Você deve alterar a função **calcularPrestacoes** para que os novos testes possam ser atendidos.

Algumas observações:

- Não é necessário criar testes para números especiais, como **NaN, EPSILON,** entre outros. Números comuns são aceitáveis.
- Cada novo teste deve verificar um ponto. Por exemplo: ao calcular as prestações com montante negativo, deve retornar array vazio; e assim sucessivamente.

3. **ATIVIDADE 03:** altere o projeto ’03-consulta-credito’ acrescentando ao arquivo **app.js** uma nova rota **HTTP GET /cliente** que irá retornar uma lista com todos os clientes persistidos na base de dados. Altere também o arquivo **app.test.js**, criando um teste que retorne um JSON com todos os clientes persistidos na base da aplicação.