**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

André Avelar, Lucas Santos, Matheus Fontes e Rogélio Filho

**EDN Representações**

**Belo Horizonte – MG**

**2023**

* Software
  + **Nome do software que foi testado:** EDN Representações
  + **Principais funcionalidades:** O software é destinado a uma empresa que comercializa roupas e uniformes para outras instituições, sendo um sistema que permite a solicitação de orçamento e diálogo entre comerciantes e vendedores.

Além disso, a empresa detentora do projeto poderá adicionar os novos produtos que começaram a vender.

* + **Público principal (a quem se destina o software):** O software é destinado a empresas que desejam adquirir uniformes.
  + **Tecnologias utilizadas no Software com um breve descritivo para cada uma delas:**

***Spring Boot*** *->* Spring Boot é um framework Java que facilita a criação de aplicativos autônomos e prontos para produção. Ele simplifica o desenvolvimento, oferecendo um ambiente pré-configurado, abstraindo grande parte da configuração e permitindo o rápido desenvolvimento de aplicativos com Spring. Com Spring Boot, é possível criar aplicativos robustos, escaláveis e altamente eficientes com facilidade, usando convenções prontas e padrões de projeto.

***React*** *->* React é uma biblioteca JavaScript popular para construir interfaces usuário interativas. Desenvolvida pelo Facebook, ela permite criar componentes reutilizáveis que gerenciam seu próprio estado e podem ser compostos para criar interfaces complexas. O React utiliza uma abordagem baseada em componentes, onde cada componente é uma peça isolada que contém sua própria lógica e representação visual.

***Arquitetura MVC*** -> A arquitetura MVC (Model-View-Controller) é um padrão de projeto que separa uma aplicação em três componentes principais:

*Model (Modelo):* Representa a lógica de negócios e os dados da aplicação. Ele interage com o banco de dados, realiza operações de leitura e escrita, e fornece os dados para a View.

*View (Visão):* É a camada de apresentação da aplicação. É responsável pela exibição dos dados ao usuário final. Recebe informações do Modelo e as renderiza visualmente para o usuário.

*Controller (Controlador):* Atua como intermediário entre a View e o Model. Recebe as interações do usuário na interface (como cliques em botões) na View e traduz essas interações em ações no Modelo. Ele manipula as requisições do usuário e atualiza a View com os dados do Modelo.

* + ***Plataformas (se é Web, Mobile, Desktop, etc):*** O software é destinado a web.
  + **Tamanho (número de linhas de código, classes e métodos):** O software é bastante robusto, uma vez que se utiliza da arquitetura MVC para o backend, fazendo com que existam classes de Controller, Service e Repository para cada elemento existente no sistema.
* Testes:
  + **Texto Descritivo sobre Framework/Ferramenta utilizada:**

O JUnit é um framework de testes unitários para Java que permite a criação e execução de testes de forma automatizada, garantindo a verificação do comportamento esperado de cada parte do código.

O Mockito é uma biblioteca de mocking para Java, usada em conjunto com o JUnit, que permite criar objetos simulados (mocks) para simular o comportamento de componentes externos ou dependências durante os testes, facilitando o isolamento e a criação de cenários controlados para os testes unitários.

* + **Caracterização dos testes realizados (Se foi unitário, integração, *end to end*):** Os testes realizados foram unitários no ambiente de Spring Boot.
  + **Cálculo de Cobertura:** Para fazer o cálculo de cobertura, a ferramenta utilizada será o JaCoCo (Java Code Coverage).
  1. Adicione a dependência do JaCoCo no seu projeto (Maven ou Gradle).
  2. Execute os testes do seu projeto com o suporte do JaCoCo habilitado. Isso pode ser feito usando um comando específico do build ou através da configuração de plugins no seu ambiente de desenvolvimento.
  3. Após a execução dos testes, o JaCoCo gera relatórios de cobertura que detalham a porcentagem do código-fonte que foi coberta pelos testes.
  4. Você pode visualizar esses relatórios usando ferramentas de visualização específicas do JaCoCo ou integrando-os a ferramentas de CI/CD ou IDEs, para avaliar a cobertura de código alcançada.
  + Caso a cobertura não tenha atingido 100%, descrever quais arquivos não foram testados e por quê
  + Colocar trechos de códigos de exemplo (Código do software x Teste Desenvolvido)
  + Passo a passo descrevendo os comandos para testar a plataforma (o que instalar, o que digitar no terminal, etc)