**JALA UNIVERSITY**

**ENGENHARIA DE SOFTWARE COMERCIAL**

FELIPE DA CONCEIÇÃO ALVES

# A.1 LABORATÓRIO

semana 1

**DISCIPLINA:**  Programação 5

**PROFESSOR(A):** Felipe B.

**TUTOR**: Hector Tanabe

**GRUPO:** A

RECIFE

2024

# 

# SUMÁRIO

[**A.1 LABORATÓRIO 1**](#_7m6z0xff7csg)

[**SUMÁRIO 2**](#_hdih5b1u4gu4)

[**Relatório de Pesquisa sobre Ferramentas 3**](#_yaba5r5qxfng)

[Por que é importante usar a versão LTS do Node.js em projetos? 3](#_q0eac345z7yc)

[Três características destacadas do Node.js 18.x 3](#_gpn6nffy8djy)

[Vantagens do Angular CLI 16.x em relação às versões anteriores 4](#_i34bualy65pu)

[Melhorias recentes no Visual Studio Code (1.84 ou superior) 4](#_hn08npb5qa7x)

[Comparação entre ESLint, Prettier e Angular Language Service 4](#_om5kq01bz884)

[Conclusão 5](#_bt5vr2gu0f07)

# Relatório de Pesquisa sobre Ferramentas

## Por que é importante usar a versão LTS do Node.js em projetos?

A escolha de uma versão LTS (Long-Term Support) do Node.js é essencial para garantir estabilidade e segurança em projetos. Versões LTS são mantidas por um período prolongado, geralmente com atualizações regulares de segurança e correções de bugs críticos. Isso torna essas versões ideais para aplicações em produção, pois minimizam riscos associados a mudanças disruptivas ou introdução de funcionalidades experimentais. Além disso, muitas bibliotecas e frameworks populares do ecossistema Node.js priorizam compatibilidade com versões LTS, garantindo que os desenvolvedores tenham suporte confiável ao longo do ciclo de vida do projeto.

Usar versões LTS também facilita o gerenciamento de equipes e dependências, pois a estabilidade reduz a necessidade de frequentes atualizações de código para lidar com breaking changes. Isso permite que as equipes se concentrem no desenvolvimento de funcionalidades e na entrega de valor ao usuário, sem se preocupar constantemente com problemas de compatibilidade ou segurança.

## Três características destacadas do Node.js 18.x

1. **Fetch API integrada**: O Node.js 18 introduziu a API fetch como um recurso embutido, eliminando a necessidade de dependências externas, como node-fetch ou axios, para fazer requisições HTTP. Isso simplifica o código e melhora a compatibilidade com padrões modernos de JavaScript.
2. **Suporte aprimorado ao Módulo Teste (Test Runner)**: Esta versão incluiu um módulo nativo de teste experimental (node:test), permitindo que os desenvolvedores criem testes unitários diretamente no Node.js sem precisar de bibliotecas externas. Isso reduz a curva de aprendizado e promove práticas de desenvolvimento orientadas a testes (TDD).
3. **Atualizações no V8 e Melhorias de Performance**: Com o motor V8 atualizado, o Node.js 18 trouxe melhorias significativas em desempenho, especialmente para aplicativos que lidam com grande volume de operações assíncronas. A compatibilidade com novos recursos do ECMAScript também oferece uma base sólida para desenvolvimento moderno.

## Vantagens do Angular CLI 16.x em relação às versões anteriores

O Angular CLI 16.x introduz várias melhorias que otimizam o desenvolvimento de aplicativos Angular. Uma das principais vantagens é o suporte ao **Standalone Components**, que simplifica a estrutura de módulos ao permitir que componentes, diretivas e pipes sejam declarados e utilizados sem dependências explícitas de módulos. Isso reduz a complexidade do código e acelera o desenvolvimento.

Além disso, o CLI 16.x melhorou o desempenho de build e tempo de execução por meio de atualizações no compilador e no sistema de empacotamento. Recursos como detecção automática de alterações (Incremental Compilation) e suporte a bibliotecas modernas também contribuem para builds mais rápidos e eficientes. Finalmente, o CLI continua oferecendo suporte robusto para internacionalização (i18n), ferramentas de teste e integração com o Angular DevTools, aprimorando a experiência do desenvolvedor.

## Melhorias recentes no Visual Studio Code (1.84 ou superior)

A versão 1.84 do Visual Studio Code incluiu aprimoramentos significativos na usabilidade e desempenho. Um dos destaques é a funcionalidade de **Profiles**, que permite alternar rapidamente entre configurações personalizadas de extensões, atalhos e temas, ideal para desenvolvedores que trabalham em diferentes tipos de projetos ou equipes.

Outra melhoria notável é o suporte aprimorado a **Live Preview**, especialmente para projetos web. Essa funcionalidade permite visualizar mudanças no navegador em tempo real enquanto edita código. Além disso, atualizações no editor de terminais integrados e no sistema de busca tornaram o fluxo de trabalho mais ágil e responsivo.

## Comparação entre ESLint, Prettier e Angular Language Service

1. **ESLint**: Focado em análise estática de código, o ESLint ajuda a encontrar e corrigir problemas de sintaxe, estilo e possíveis bugs em JavaScript/TypeScript. Ele é altamente configurável, suportando regras personalizadas e integração com outros sistemas de linting. Sua principal vantagem é garantir qualidade e consistência no código, além de prevenir erros em tempo de desenvolvimento.
2. **Prettier**: É uma ferramenta de formatação de código opinativa que garante que todo o código siga um estilo consistente. Diferentemente do ESLint, o Prettier não detecta problemas lógicos, mas é ideal para evitar debates sobre estilo entre equipes, automatizando a formatação e melhorando a legibilidade.
3. **Angular Language Service**: Este serviço é específico para projetos Angular e fornece recursos como autocompletar, checagem de erros em tempo real e navegação por templates. Ele melhora a produtividade ao oferecer suporte direto ao desenvolvimento baseado em Angular, reduzindo erros relacionados à sintaxe ou ao uso incorreto de APIs do framework.

### Conclusão

As ferramentas investigadas neste relatório são pilares importantes no desenvolvimento de software moderno. Desde o uso de versões LTS do Node.js para garantir estabilidade, até as funcionalidades avançadas do Angular CLI, Visual Studio Code e serviços auxiliares como ESLint e Prettier, cada tecnologia desempenha um papel fundamental na construção de projetos robustos e eficientes. Entender suas características e vantagens permite escolher as soluções mais adequadas para cada contexto, otimizando tanto a experiência do desenvolvedor quanto o resultado final do projeto.

Toda documentação está disponível no repositório do github:  
  
<https://github.com/alves-py/Programa-o5_JalaUniversity>