**PÓS-GRADUAÇÃO EM DESENVOLVIMENTO FULL STACK**

**TRABALHO DE CONCLUSÃO DE CURSO: MEMORIAL**

MATHEUS ANDREY RODRIGUES

Orientador: Daniel Antônio Callegari

2024



**SUMÁRIO**

[1. Trajetória Pessoal 3](#_Toc160563585)

[1.1. História de vida 3](#_Toc160563586)

[1.2. Áreas selecionadas 4](#_Toc160563587)

[2. Organização do Portfólio 5](#_Toc160563588)

[2.1. Fundamentos de computação e algoritmos 6](#_Toc160563589)

[2.2. Arquitetura *Client-Side* 7](#_Toc160563590)

[2.3. Arquitetura *Server-Side* 8](#_Toc160563591)

[2.4. Micro Serviços 9](#_Toc160563592)

[2.5. Banco de dados 9](#_Toc160563593)

[2.6. Segurança de software 10](#_Toc160563594)

[2.7. DevOps 11](#_Toc160563595)

[2.8. Qualidade e Testes de Software 12](#_Toc160563596)

[3. Recursos explorados e suas aplicações 13](#_Toc160563597)

[4. Lições aprendidas durante a formação 22](#_Toc160563598)

[5. Competências e expectativas 24](#_Toc160563599)

[5.1. Competências Adquiridas 24](#_Toc160563600)

[5.2. Expectativas em relação ao curso 24](#_Toc160563601)

[5.3. Próximos passos 25](#_Toc160563602)

[Referências 26](#_Toc160563603)

# Trajetória Pessoal

Minha trajetória pessoal é apresentada em duas partes: (a) minha história de vida e (b) as áreas selecionadas. Em minha história de vida detalho minha vida antes a realização deste curso, bem como os motivos que me trouxeram até ele e as contribuições técnicas e comportamentais que a realização deste curso me ofereceu. Ao tratar sobre minhas áreas de interesse, darei enfoque a algumas tecnologias que me chamaram a atenção e que entendo serem importantes na construção de minha vida profissional.

## História de vida

Ao longo dos meus 23 anos de vida, minha jornada tem sido permeada por uma fascinação constante pela tecnologia e uma busca incessante por compreender os caminhos desse universo dinâmico. Este relato não é apenas uma retrospectiva pessoal, mas também um olhar para o futuro que delineia minha escolha profissional e as linhas de atuação que planejo seguir.

No entanto, o verdadeiro ponto de inflexão que direcionou minha trajetória rumo à área de Tecnologia da Informação (TI) foi a influência do meu irmão mais velho, onde temos uma diferença de 10 anos de idade. Desde muito cedo, como comentado anteriormente, tinha sede em descobrir novas coisas sobre tecnologia e eu o acompanhava enquanto realizava algumas manutenções gerais nos computadores em nossa casa. Essa exposição precoce a ambientes tecnológicos e seus estudos dedicados ao desenvolvimento de software foram cruciais para despertar em mim uma determinação de seguir esse mesmo caminho.

Concluí o curso técnico em informática na ETEC Polivalente de Americana, obtive a graduação em Sistemas de Informação pela PUC Campinas e, desde os meus 17 anos, venho atuando como desenvolvedor de software. Ao longo da minha carreira, tive a oportunidade de contribuir em duas outras empresas como desenvolvedor de software, e atualmente já graduado, exerço a função de Analista de Sistemas Pleno, em um instituto de tecnologia chamado Venturus. Durante meu dia a dia de trabalho senti que precisava de uma nova especialização, me tornar um desenvolvedor Full Stack para conseguir superar os desafios que me eram expostos no trabalho e alcançar novas oportunidades de crescimento em minha carreira.

## Áreas selecionadas

Em minha constante busca por aprendizado e já considerando a boa experiência no mercado de desenvolvimento de software, identifiquei claramente os pontos cruciais e necessários para alcançar novas oportunidades de crescimento pessoal e profissional. Nesse contexto, as áreas estratégicas selecionadas são arquitetura server/cliente *side*, arquitetura de software em nuvem, micro serviços, banco de dados Relacional/NoSQL, segurança de software, qualidade e testes de software e DevOps.

Ao compreender a relevância dessas áreas que estão em constante evolução, me comprometido em aprofundar meus conhecimentos e habilidades, visando não apenas acompanhar as demandas do mercado, mas também antecipar os futuros desafios que posso enfrentar por já estar atuando na área. Desta forma busquei algo que pudesse suprir as informações que eu buscava, de uma forma simples e clara e foi aí que encontrei a pós-graduação de desenvolvimento FullStack da PUC RS. Desde o início a orientação estratégica que pude encontrar nas aulas da pós-graduação refletiram muito no meu dia a dia de trabalho, já podendo contribuir de uma maneira significativa juntamente a minha equipe e fortalecendo meu papel como profissional no dinâmico cenário do desenvolvimento de software.

# Organização do Portfólio

Nesta seção apresentarei um conjunto de soluções elaboradas durante a realização deste curso e que considero importante para expressar minha formação. As soluções são apresentadas considerando a linha do tempo de realização das disciplinas e optei por utilizar meu espaço no GitHub disponibilizado em <https://github.com/mthrodrigues301/posdsf-pucrs> como repositório principal para o armazenamento dos conjuntos de códigos desenvolvidos durante o curso.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente com confiança média

Figura 1 – Repositório GitHub de armazenamento de código

## Fundamentos de computação e algoritmos

O primeiro conjunto de soluções que apresento fazem menção a disciplina de “fundamentos de computação e algoritmos”. Os códigos desenvolvidos estão disponíveis para acesso em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/FUND_COMP_E_ALG>. O grande objetivo daquele repositório foi o de investigação da linguagem de programação Javascript, sendo assim foram elaboradas algumas soluções e escolhi as seguintes para serem comentadas.

A primeira solução está disponível para acesso em <https://github.com/mthrodrigues301/posdsf-pucrs/blob/main/FUND_COMP_E_ALG/JAVASCRIPT/01_Hello_World.js>, tinha por objetivo a validação do ambiente de desenvolvimento e um primeiro contato com a linguagem.

A segunda, disponível para acesso em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/FUND_COMP_E_ALG/MINI_BROWSER>, tinha por objetivo explorar recursos um pouco mais avançados da linguagem, como por exemplo, o entendimento a aplicação dos conceitos de fila, pilha, set e map.

## Arquitetura *Client-Side*

Para a disciplina "Arquitetura Client-Side", o conjunto de soluções desenvolvidas está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/ARQ_CLIENT_SIDE>. O grande objetivo desta disciplina foi fornecer conhecimentos desde o básico, como fundamentos web, HTML, CSS, JavaScript e TypeScript, até o avançado, como a integração com serviços web e APIs, essenciais para o funcionamento da maioria das soluções cliente-servidor. Ao final do curso, com base em todo o conhecimento adquirido, desenvolvemos uma aplicação do tipo SPA (Single Page Application). Durante o aprendizado de conceitos, exploramos a possibilidade de desenvolvimento web, mobile e híbrido. Os professores optaram por ensinar React, uma biblioteca desenvolvida pelo time do Facebook (hoje Meta) em 2011 e hoje amplamente consolidada, que ainda está muito em alta no mercado do desenvolvimento front-end e supre bem as necessidades que se propõe a resolver.

As primeiras soluções que gostaria de mencionar estão disponíveis em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/ARQ_CLIENT_SIDE/compilers> e referem-se às ferramentas utilizadas para a construção e desenvolvimento de um projeto com React. O CRA (Create React App) é o configurador que surgiu junto à biblioteca e atende muito bem às necessidades, mas ao longo do tempo surgiram dores associadas a ele, o que levou à criação do Vite, que se propõe a ser mais rápido, flexível e fácil de configurar em comparação ao CRA.

A segunda solução que merece destaque está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/ARQ_CLIENT_SIDE/demo_react_component> e aborda um dos principais pontos da utilização do React: a forma simplificada de aplicar o conceito de componentes. Com o React, conseguimos separar nossa aplicação para que cada pedaço de código carregue unicamente suas responsabilidades e possa ser reutilizado em outras partes do sistema, sem duplicidade de código. Também aprendemos ferramentas do React, como eventos, formulários, useEffect, useReducer, useState e o conceito de construção de uma aplicação com arquitetura limpa (Clean Architecture).

Por fim, exploramos a construção de uma aplicação utilizando o framework Next.js, disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/ARQ_CLIENT_SIDE/next-fcc-familyguy>, que é um grande facilitador. O Next.js oferece diversas ferramentas, como o *server* *rendering*, que permite ao servidor renderizar as telas previamente para o cliente.

## Arquitetura *Server-Side*

Sobre a disciplinas “Arquitetura *Server-Side*”, onde o código desenvolvido está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/ARQ_SERVER_SIDE>. Considero esta disciplina de extrema importância e também tenho mais afinidade com ela, pois faz parte do meu dia a dia de trabalho. Durante o curso, fomos apresentados a um conteúdo de alta qualidade. Iniciamos com o entendimento da arquitetura MVC (Model-View-Controller) utilizando NodeJS e a biblioteca ExpressJS. Além disso, exploramos programação assíncrona, reativa e o funcionamento do NodeJS, além de conceitos fundamentais como HTTP (Hypertext Transfer Protocol). Esses tópicos são essenciais para compreender a construção de uma API (Application Programming Interface), sua arquitetura, comunicação, entre outras etapas.

Também foram abordados os conceitos de SSR (Server Side Rendering). Por exemplo, antigas soluções de renderização do lado do servidor possibilitavam a criação de páginas web (com PHP, por exemplo), onde o servidor compilava o HTML e os dados, enviando tudo para o cliente exibir.

Com o objetivo de ter mais contato com aplicações de estruturas reais, foi apresentado o framework NestJS, comumente utilizado para a construção de backend junto com NodeJS. Com base no NestJS, exploramos os conceitos de *controllers*, *providers*, *modules*, *middlewares*, *pipes*, testes e como podemos contruilos, utilizando formas como o *mock* que simulam o comportamento de objetos reais de entrada e saída de forma.

Por fim e não menos importante, tivemos contato com a construção de documentação para uma API, utilizando o Swagger, uma biblioteca facilitadora para gerador de documentações e em complemento a utilização de *API Model Description*, mapeando assim os modelos disponíveis pela aplicação.

## Micro Serviços

Referente a disciplinas “Micro Serviços”, onde o código desenvolvido está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/MICRO_SERVICOS>. Teve objetivo de construir uma base solida sobre os conhecimentos do que são micro serviços, os tipos, como funcionam, a diferença entre as arquiteturas de micro serviços e o tão conhecidos monolíticos e como são aplicados no dia a dia dos softwares. Quando o assunto é micro serviços não podemos nos esquecer também, como os mesmos devem ser arquitetados em relação a conteinerização com Docker, pipelines e as nuvens, assuntos que também foram abordados durante as matérias. Trabalhamos então, com um micro serviço na prática que está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/MICRO_SERVICOS/exemplo-microsservicos> e também com a implementação do RabbitMQ um servidor de mensageria, implementado para suportar mensagens em um protocolo chamado AMQP (*Advanced Message Queuing Protocol*) entre aplicações, disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/MICRO_SERVICOS/produtor-consumidor>.

## Banco de dados

As soluções referentes às disciplinas "Banco de Dados Relacional" e "Banco de Dados NoSQL" estão disponíveis em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/BANCO_DE_DADOS>. Durante a disciplina de "Banco de Dados Relacional", onde todo conteúdo desenvolvido está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/BANCO_DE_DADOS/BD_RELACIONAL>, foram realizadas diversas atividades, tais como o desenho do modelo lógico utilizando Diagrama de Entidade-Relacionamento (DER) e diagrama de classes, todos construídos por meio da ferramenta Draw.io, acessível em <https://app.diagrams.net>, e o desenho do modelo de relacionamento com a ferramenta brModelo, disponível para download em <http://www.sis4.com/brModelo>. Além disso, foram elaborados scripts para executar a criação de tabelas, buscas, inserções, atualizações e remoções de dados. Destaca-se ainda a utilização da linguagem Python para exemplificar o desenvolvimento prático utilizando o ORM (*Object Relational Mapper*), em conjunto com o SQLAlchemy, uma biblioteca que facilita a criação de modelos de dados em bancos de dados. Foram utilizados, como exemplos, o banco de dados Oracle, acessado por meio da ferramenta Live SQL disponível em <https://livesql.oracle.com>, e o SQLite, juntamente com o SGBD SQLiteStudio, disponível para download em <https://sqlitestudio.pl>.

Na disciplina de "Banco de Dados NoSQL", novamente foram abordados temas como a construção de modelos conceituais de entidade e relacionamento, orientado a objetos e lógico-relacional, além da normalização de dados. Aprendemos também sobre os modelos de transações de SGBDs (Sistema de Gerenciamento de Banco de Dados), como ACID (*Atomic Consistent Isolated Durable*), que garante o correto funcionamento do sistema e previne a corrupção ou perda de dados, e BASE (*Basically Available Soft state Eventually consistent*), que prioriza a disponibilidade do serviço sobre a consistência dos dados. Durante o curso, foram apresentados diversos bancos de dados, como Redis (do tipo *Key Value*), MongoDB (do tipo *Document-Based*), Neo4J (do tipo *Graph*) e CassandraDB (considerado do tipo híbrido, por aceitar mais de um tipo, mas apresentado como *Column-Based*). Para ilustrar os conceitos, o professor optou por desenvolver scripts de exemplo para todos os bancos mencionados, e os códigos estão disponíveis em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/BANCO_DE_DADOS/BD_NOSQL>.

## Segurança de software

Para a disciplina "Segurança de Software", o código desenvolvido está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/SEGURANCA_SOFTWARE>. O objetivo desta disciplina foi compreender, tanto na teoria quanto na prática, o funcionamento da segurança de softwares e como as aplicações, de modo geral, se protegem contra os ataques aos quais estão suscetíveis ao estarem conectadas à internet. Ao longo do curso, também foi introduzido o OWASP (*Open Web Application Security Project*), uma comunidade que oferece gratuitamente artigos, metodologias, documentação, ferramentas e tecnologias relacionadas à segurança de aplicações web.

A solução prática desenvolvida, concentrou-se no estudo do funcionamento das criptografias simétricas de bloco AES (*Advanced Encryption Standard*), em seus modos CTR (*Counter*) e CBC (*Cypher Block Chaining*). Essa é uma das criptografias mais amplamente utilizadas em todo o mundo e tem como objetivo geral garantir que somente os usuários de origem e destino tenham acesso aos dados trafegados.

## DevOps

As soluções referentes às disciplinas 'DevOps Básico' e 'DevOps Avançado' estão disponíveis em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/DEVOPS>. No 'DevOps Básico', cujo código desenvolvido está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/DEVOPS/DEVOPS_BASICO>, o professor convidado, Fabricio Veronez, nos proporcionou um amplo ensino sobre o processo de DevOps e apresentou os *The Twelve-Factor Apps*, que constituem um conjunto de doze fatores e princípios que, quando seguidos, tornam possível a criação de aplicações como serviço com grande eficiência. Além disso, foram apresentadas ferramentas de mercado, como o Heroku, que oferece serviços de nuvem com suporte para várias linguagens de programação. Para as atividades práticas, optou-se por utilizar uma aplicação já desenvolvida em NodeJs para aplicar na prática a construção de todo o processo de CI/CD, utilizando containers com Docker e GitHub Actions.

Quanto à disciplina 'DevOps Avançado', as soluções desenvolvidas estão disponíveis em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/DEVOPS/DEVOPS_AVANCADO>. Esta disciplina teve seu foco voltado para a implementação de soluções na plataforma AWS da Amazon, responsável por disponibilizar serviços de nuvem, mais especificamente o *CloudFormation*, disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/DEVOPS/DEVOPS_AVANCADO/cloudformation-demo>. Construímos também pipelines de CI/CD para uma aplicação de teste já desenvolvida em NodeJs, utilizando novamente containers Docker, disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/DEVOPS/DEVOPS_AVANCADO/nodepipeline-devops>. Por fim, abordamos os conceitos básicos e como começar a utilizar a ferramenta de IaC (*Infrastructure as Code*), o Terraform, que permite gerenciar, provisionar e atualizar recursos de infraestrutura em nuvens públicas, privadas ou híbridas, cujo código desenvolvido está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/DEVOPS/DEVOPS_AVANCADO/terraform-demo>.

## Qualidade e Testes de Software

Por último, mas não menos importante, gostaria de destacar o conteúdo abordado na disciplina "Qualidade e Teste de Software", cujos códigos desenvolvidos podem ser acessados em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/QA_TESTE_SOFTWARE>. Nesta disciplina, exploramos diversas estratégias para a implementação de testes, incluindo metodologias consagradas como TDD (Desenvolvimento Orientado a Testes) e BDD (Desenvolvimento Orientado a Comportamento). Além disso, discutimos a integração de testes abrangendo desde os de ponta a ponta (*end-to-end*) até os de desempenho e testes unitários nos fluxos de desenvolvimento. Esses testes podem ser executados de forma reativa localmente, por meio de pipelines ou automaticamente a cada *commit*.

Uma das soluções que gostaria de destacar está disponível em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/QA_TESTE_SOFTWARE/JEST_DEMO>. Nessa solução, exploramos desde os conceitos básicos até os avançados do framework Jest, que é uma das principais ferramentas para a realização de testes unitários utilizando a linguagem de programação JavaScript.

Outra solução relevante, encontrada em <https://github.com/mthrodrigues301/posdsf-pucrs/tree/main/QA_TESTE_SOFTWARE/VVSUPER>, envolve o uso da linguagem de programação Java em conjunto com o framework JUnit 5. Nesse contexto, foram explorados casos de implementação de testes automatizados em aplicações desse tipo.

# Recursos explorados e suas aplicações

O curso de pós-graduação em Desenvolvimento *Full Stack* explorou de forma bastante ampla vários conceitos e recursos que podem ser utilizados na construção de uma solução completa em projetos web, desde a construção da arquitetura até a experiência do usuário. De todos os assuntos abordados, me interessei de forma mais significativa com as partes de arquitetura *server* e *client side*, banco de dados, segurança de *software*, qualidade de *software* e DevOps.

A arquitetura *server-side*, tem o objetivo de estruturar o lado em que está mais distante do usuário final e é o responsável na maior parte das vezes por armazenar e estruturar as regras de negócios das aplicações, executar integrações com banco de dados entre outros serviços. O *server-side* processa as solicitações vindas do *client* (como navegadores da web, aplicativos moveis ou outros dispositivos), executa a lógica de negócios necessária e retorna a resposta ao *client*.

Nesta disciplina do curso, nos foi apresentado diversos padrões e ferramentas necessárias para se trabalhar com esse tipo de aplicação. Por exemplo, construímos uma API (*Application Programming Interface*) REST (*Representational State Transfer*) utilizando o NodeJS, responsável por interpretar e executar o JavaScript e o *framework* Express, que fornece recursos mínimos para construção de servidores web, utilizando por exemplo a arquitetura MVC, explorando também os conceitos de assincronismos e reatividade.

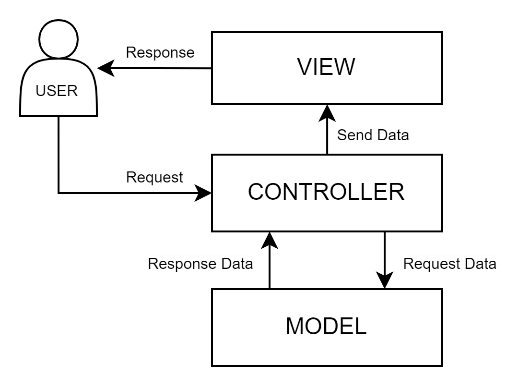


Figura 2 – Diagrama arquitetura MVC

A arquitetura *client-side* tem o objetivo de estruturar o lado em que está mais próximo ao cliente, podemos dizer que é construída uma interface de interação entre o humano computador, então responsável por integrar e dar a visibilidade de forma simplificada as regras de negócios muita das vezes vindas do *server-side*.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Figura 3 - Arquitetura de comunicação client-server

Ao iniciar o desenvolvimento de uma aplicação web, devemos pensar em qual arquitetura podem ser utilizadas, durante a disciplina, tivemos contato com o CSR (*Client-Side Rendering*), mais comumente utilizado e onde o cliente solicita os dados ao servidor, espera o retorno e somente assim disponibiliza a página construída com as informações solicitadas pelo usuário e também temos o SSR (*Server-Side Rendering*), onde a busca dos dados e a construção da página fica 100% responsável pelo servidor e o browser somente exibe a “tela” que o servidor o forneceu.

Diagrama, Texto

Descrição gerada automaticamente

Figura 4 - Arquitetura CSR e SSR

Após a escolha da arquitetura, devemos estudar quais as linguagens e bibliotecas que melhor fornecem o necessário para construção dela. Durante a disciplina tivemos com HTML, CSS, JavaScript, TypeScript e algumas bibliotecas como React, Angular, Vue.JS algumas das ferramentas fundamentais para a construção do lado do cliente no desenvolvimento WEB.

Ao pensar no desenvolvimento *client-side*, não podemos nos esquecer que vai muito além da construção de telas para site na web, devemos lembrar que na verdade o desenvolvimento *client-side* também aborda a construção de aplicações denominadas *desktop* e aplicativos *mobile* de forma nativa. Pela necessidade de construir diversas aplicações e muita das vezes com tempo escasso, surgem as abordagens hibridas ou PWA (*Progressive Web App*), que conseguem executar aplicações compatíveis com a *web*, *desktop* e outros tipo de *devices*, onde podemos citar o React Native, Flutter, Xamarin e Ionic, como os principais frameworks que atendem a abordagem hibrida hoje no mercado.

Como um dos pilares de uma aplicação o banco de dados é o responsável por armazenar os dados utilizados nos sistemas e ele pode ser estruturado de algumas formas, sendo dos tipos relacional e não relacional. Desta forma nos foi apresentada duas disciplinas, banco relacional onde tivemos contato com a construção de MER (Modelo de Entidade Relacionamento), que é utilizado para entender os conceitos e palavras-chave implantados no sistema de banco de dados.

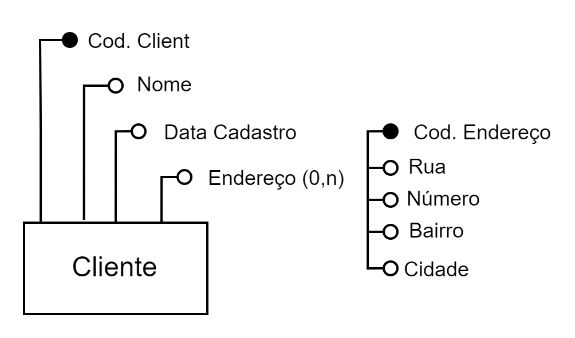


Figura 5 - Modelo Entidade-Relacionamento

As entidades e atributos são os fatores mais importantes desse modelo, que serão identificados de acordo com os requisitos do sistema. Nos foi apresentado também o modelo lógico que é o avanço para o desenvolvimento da estrutura do banco de dados em si e que identificamos tipos para colunas, identificadores, chaves primarias, compostas e estrangeiras para os relacionamentos entre si. A ferramenta que foi utilizada para a identificação e documentação do banco de dados foi o, Draw.io, que disponibiliza todos os recursos necessários para esta etapa de documentação, mas é possível encontrar outras ferramentas do tipo sendo utilizadas no mercado.

Para o banco de dados de fato, nos foram apresentados alguns sistemas de gerenciamento de baco de dados (SGBD), como o MySQL, Oracle SQL, Microsoft SQL Server, Azure SQL DB, PostgreSQL, SQLite e IBMDB2.

Já o banco de dados não relacional, também conhecidos como de NoSQL, tem a divisão em quatro tipos, *Key Value* (Chave valor), *Document-Based* (Baseado em documentos), *Column-Based* (Baseado em Colunas) e *Graph* (Gráfos).

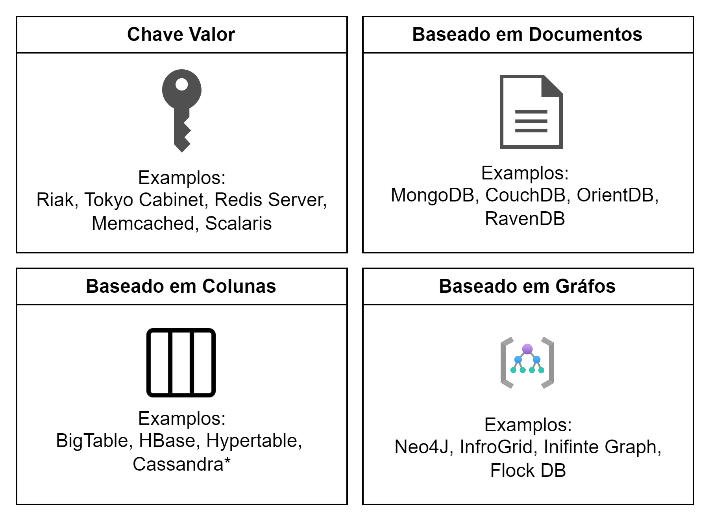


Figura 6 – Tipo de banco de dados NoSQL

Ao pensar na implementação de um banco de dados não relacional NoSQL, costumamos pensar no teorema CAP, *Consistency* (Consistencia), *Availability* (Disponibilidade), *Partition tolerancy* (Tolerância de partição ou particionamento), proposto por Eric Brewer em 1998, que consiste em quando ocorre uma falha de partição (P), deve-se decidir se a operação será cancelada e, portanto, diminuirá a disponibilidade (A), mas garantirá a consistência (C), ou prosseguirá com a operação e, assim, fornecerá disponibilidade (A), mas arriscará a inconsistência (~C), assim, se houver um particionamento de dados, deve-se escolher entre consistência (C) ou disponibilidade (A).

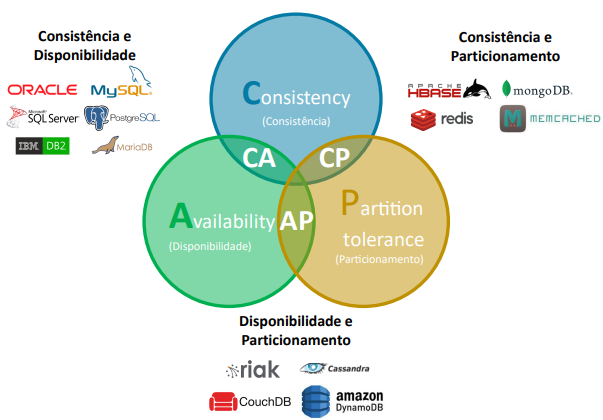


Figura 7 – SGBD’s e suas alocações no teorema de CAP

Visando citar alguns dos pontos principais sobre bancos NoSQL, podemos começar falando sobre a flexibilidade em relação a estrutura de dados, pois pode ser utilizado de forma estratégica por sistemas que necessitam de armazenar dados com estruturas sobe demanda (variáveis) e em relação a sua capacidade de armazenamento, que no geral é bem maior que banco de dados relacionais. Outro ponto positivo em relação aos bancos de dados NoSQL é que a maior parte dos bancos tem facilitado a integração com as *Clouds*, facilitando a construção da arquitetura *cloud-native* como um todo. Os bancos de dados que nos foram apresentados durante a matéria foram, MongoDB (*Document-Based*), Redis (*Key Value*), Neo4j (*Graph*) e CassandraDB (*Column-Based*).

A qualidade no desenvolvimento de software é um pilar que deve ser levado com excelência onde se garante a entrega de um sistema coeso e sem bugs para os usuários final, onde segundo o IEEE é “o grau de conformidade de um sistema, componente ou processo com seus respectivos requisitos”. Podendo ser dividida em alguns quadrantes como, testes manuais e/ou automatizados, testes unitários, testes integração, testes de interface com o usuário, acessibilidade, performance, localização e A/B.

Nos testes unitários, podemos citar duas principais estratégias o TDD (*Test Driven Development*), onde cria-se os testes unitários antes de começar a codificação de fato. Nessa estratégia os testes devem falhar e após a implementação da solução desejada, todos os testes devem ser aprovados com êxito. Já a abordagem BDD (*Behavior Driven Development*), entende-se a demanda da unidade para que sejam criados os devidos testes, ou seja, temos um determinado conjunto esperado de entradas e saídas conhecidos, se escreve os testes necessários para atender aos requisitos.

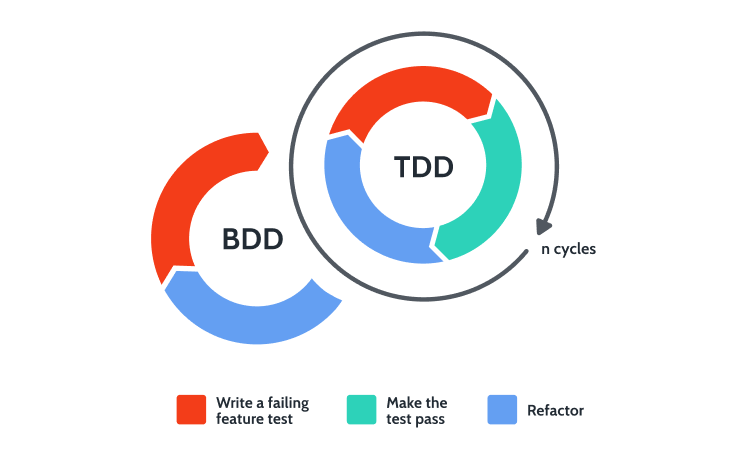


Figura 8 – Processo das metodologias BDD e TDD

No processo de qualidade, o levantamento de métricas é um dos processos essencial, onde são possíveis encontrar determinados pontos de melhorias do ciclo de vida da aplicação, ou seja, quanto mais métricas existirem no processo de qualidade e verificação do projeto, mais completo e complexo serão as análises providas por ela. E para a isso, são utilizadas abordagens como as KPIs (*Key Performance Indicator*), que podem medir pontos como, satisfação do cliente, processo interno de qualidade, satisfação dos funcionários, os índices de performance, entre outros.

DevOps é uma forma de organização de desenvolvimento de projetos que vai além de uma solução web. Ela é importante por vários motivos, mas a meu ver, o mais importante é a contribuição para evitar trabalhos manuais e tediosos que podem levar a ocorrência de erros, como por exemplo, a entrega de versões nos ambientes de desenvolvimento, qualidade e produção, que contam com diversas etapas repetitivas. Tivemos duas disciplinas de DevOps quais sejam (a) DevOps básico e (b) DevOps avançado. Nelas conhecemos algumas ferramentas tais como GitHub Actions, GitLab CI/CD e Jenkins que são ferramentas concorrentes, mas ambas servem para implementação do processo de CI/CD (*Continuos Integration/Continuos Delivery*) e que são essenciais em um projeto que preza pela qualidade, pois preenche as lacunas entre as atividades e equipes de desenvolvimento e operação, reforçando a automação na compilação, teste e implantação de aplicativos.

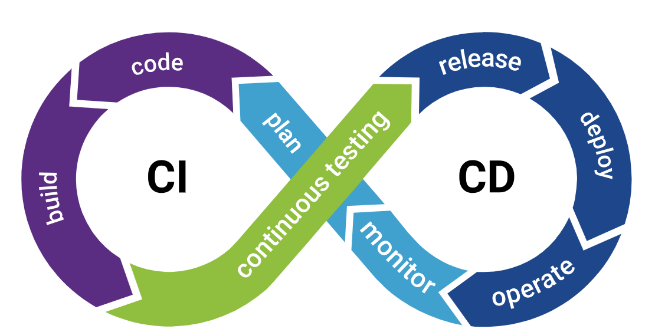


Figura 9 - Processo de CI/CD

Citando outras ferramentas apresentadas durante as aulas, o Docker que é fundamental para a aplicação do processo de conteinerização, onde podemos dividir nossos sistemas em “pacotes”, que podem ser testados e entregue para os respectivos ambientes com o auxílio da implementação do processo de CI/CD, utilizando uma ferramenta como Jenkins, por exemplo. Já o Terraform que permite a automação da IaC (*Infrastructure as Code*), facilitando a criação, configuração e gerenciamento de recursos de infraestrutura em nuvens de forma declarativa e repetível.

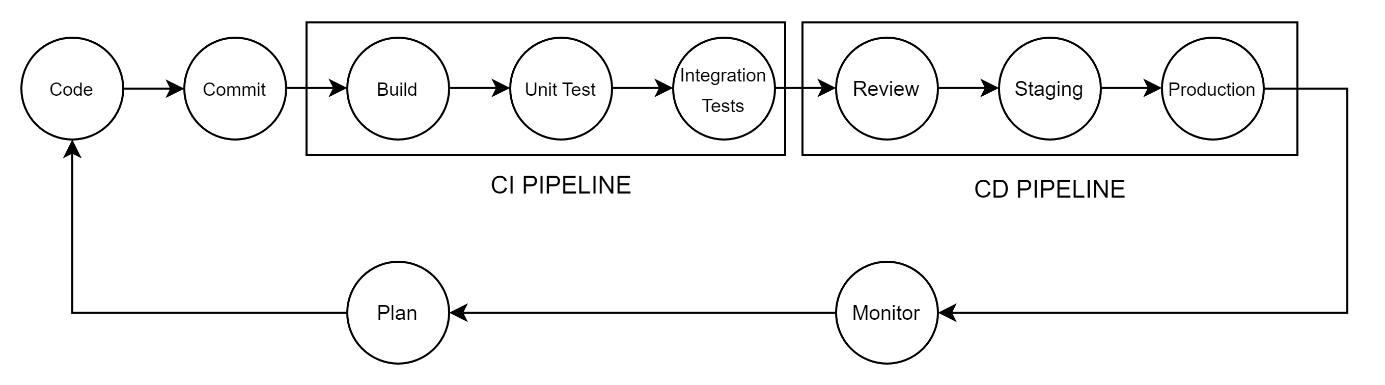


Figura 10 - Processo CI/CD

Durante o período das aulas tive a oportunidade de atuar como desenvolvedor Full Stack e líder técnico em um projeto de grande porte para uma indústria, onde então precisei atuar e aplicar 100% do conhecimento que estava sendo apresentado durante o curso, como a escolha das arquiteturas de client/server, escolha do tipo de banco de dados, relacional ou não relacional, modelagem de dados, DevOps, UI/UX, entre outros pontos que nos foi apresentado durante as aulas.

Optamos então por construir uma aplicação utilizando a linguagem de programação C# utilizando o *framework* .NET 6 (o mais atual da época), a linguagem de programação TypeScript juntamente com a biblioteca React e banco de dados Oracle, sendo essas escolhas parte do requisito do cliente. Onde então pontos como a arquitetura de micros serviços, Docker para conteinerização, mensageria (RabbitMQ) para reprocessamento de processos, Jenkins para execução de pipelines, Jest e XUnit para execução de testes e sendo interessante citar também, a construção de um ORM (*Object Relational Mapping*) próprio, por conta da abordagem dinâmica das tabelas que seriam utilizadas no processo, juntamente a um banco de dados não relacional (MongoDB) para o armazenamento estrutural das tabelas.

Uma imagem contendo Linha do tempo

Descrição gerada automaticamente

Figura 11 - Diagrama do Sistema

# Lições aprendidas durante a formação

Ao longo do curso de desenvolvimento Full Stack, uma diversidade de desafios e aprendizados moldou minha jornada formativa. Ao enfrentar diversas situações, percebi que a habilidade técnica, embora fundamental, não é suficiente para lidar com todos os obstáculos que podemos enfrentar em nosso dia a dia como desenvolvedor Full Stack. Nesse contexto, consigo enxergar de forma clara algumas lições e acontecimentos que se destacaram, contribuindo significativamente para meu crescimento profissional.

Inicialmente, identifiquei áreas do conhecimento em que já possuía certo domínio, permitindo-me aprofundar-me ainda mais nos assuntos fundamentos de computação e algoritmos, arquitetura cliente e server side, banco de dados relacional e NoSQL e a área de DevOps. Essa imersão me proporcionou um avanço técnico substancial, fortalecendo minha base de competências. No entanto, também me deparei com tópicos em que tinha pouca ou nenhuma experiência prévia, o que resultou em desafios adicionais. Lidar com essas lacunas de conhecimento exigiu esforço extra e determinação para superar as dificuldades apresentadas pela qualidade e testes de software, segurança de software, DevOps avançado, micro serviços e arquitetura em nuvem.

Essas situações foram oportunidades cruciais para desenvolver minha resiliência e capacidade de superação. Através da busca por soluções criativas e da persistência diante dos obstáculos, pude não apenas adquirir novos conhecimentos, mas também fortalecer minha confiança e habilidades interpessoais. Além disso, aprendi a importância da organização pessoal e da gestão eficaz do tempo para enfrentar as demandas complexas do curso.

Em retrospectiva, essas lições não apenas ampliaram meu repertório técnico, mas também me prepararam para os desafios do mundo profissional, onde a adaptabilidade e a capacidade de resolver conflitos são essenciais. Assim, reconheço que cada desafio enfrentado durante a formação Full Stack foi uma oportunidade valiosa de crescimento, consolidando meu compromisso em buscar constantemente o aprimoramento profissional e pessoal.

# Competências e expectativas

## Competências Adquiridas

Durante o curso de Desenvolvimento Full Stack, adquiri uma variedade de competências que têm uma aplicação direta em minha atuação profissional como Analista de Sistemas Pleno. Aprofundei-me em conceitos e práticas que abrangem tanto o desenvolvimento de client quanto de server side, permitindo-me compreender o ciclo completo de desenvolvimento de software. Entre as competências adquiridas, destaco a habilidade de desenvolver aplicações web, de ponta a ponta, utilizando diversas tecnologias responsáveis pela criação do frontend, backend e os processos de CI/CD. Além disso, aprendi a trabalhar com bancos de dados relacionais e não relacionais, bem como a utilizar frameworks e bibliotecas para agilizar o desenvolvimento de projetos.

A integração entre teoria e prática foi um aspecto fundamental do curso, permitindo-me aplicar os conhecimentos adquiridos em situações reais do dia a dia profissional. As aulas forneceram insights valiosos que me auxiliaram na resolução de problemas complexos de forma mais eficiente e eficaz.

Ao enfrentar áreas de conhecimento que demandavam um esforço adicional, desenvolvi a capacidade de perseverança e dedicação, essenciais para superar desafios técnicos. Aprendi a buscar recursos e apoio quando necessário, construindo uma mentalidade de aprendizado contínuo e adaptação às novas tecnologias e metodologias.

## Expectativas em relação ao curso

Minhas expectativas em relação ao curso de Desenvolvimento Full Stack foram amplamente superadas. Inicialmente, almejava adquirir conhecimentos técnicos sólidos que me permitissem expandir minhas habilidades profissionais. No entanto, o curso proporcionou muito mais do que isso.

Além do aspecto técnico, o curso possibilitou uma profunda reflexão sobre minha carreira e meus objetivos profissionais. Proporcionou-me a oportunidade de explorar novas áreas de conhecimento e expandir minha visão sobre o mercado de tecnologia.

Esperava enfrentar desafios e momentos de frustração, e essas expectativas se concretizaram ao longo do curso. No entanto, esses momentos foram fundamentais para meu crescimento pessoal e profissional, fortalecendo minha determinação e capacidade de superação.

## Próximos passos

Olhando para o futuro, pretendo aplicar as competências adquiridas no curso em projetos desafiadores e inovadores no mercado de tecnologia. Buscarei oportunidades que me permitam continuar aprendendo e crescendo, expandindo meu conjunto de habilidades e conhecimentos.

Almejo uma carreira que me permita fazer a diferença no mundo da tecnologia, contribuindo para o desenvolvimento de soluções que impactem positivamente a vida das pessoas. Estou aberto a explorar diferentes áreas dentro do campo da tecnologia, considerando também a possibilidade de assumir cargos de liderança técnica ou de pessoas no futuro.

Em resumo, estou comprometido em continuar minha jornada de desenvolvimento profissional, buscando sempre desafios que me permitam crescer e evoluir como um profissional de tecnologia.

# Referências

AWS. Qual a diferença entre um banco de dados ACID e um BASE?. Disponível em: <<https://aws.amazon.com/pt/compare/the-difference-between-acid-and-base-database/#:~:text=Os%20bancos%20de%20dados%20ACID,disponibilidade%20em%20vez%20da%20consist%C3%AAncia.>>. Acessado em: 18 fev. 2024.

Bonel, Claudio. Metodologia E Engenharia De Requisitos Para Projetos De Business Intelligence. Clube de Autores, 2021.

Broadcom. What can RabbitMQ do for you?. Disponível em: <<https://www.rabbitmq.com/docs/features>>. Acessado em: 25 fev. 2024.

COPELAND, Rick. Essential sqlalchemy. 2. ed. O'Reilly Media, Inc., 2016.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de banco de dados. 7 ed. São Paulo: Pearson Education do Brasil, 2018.

Express. O que é o Express?. Fundação Node.js. Disponível em: <<https://expressjs.com/pt-br/>>. Acesso em: 14 fev. 2024.

Kroth, Vinicius. Banco de dados NoSQL. Pontifícia Universidade Católica Rio Grande do Sul, Rio Grande do Sul, 2023. Disponível em: Apresentação de Apoio, Curso de Pós-graduação Desenvolvimento Full Stack. Acesso em: 17 fev. 2024.

Moreno, Edson e Flores, Marcia. Especialização em Desenvolvimento Full. PUCRS, Rio Grande do Sul, 2024. Disponível em: <<https://github.com/empucrs/DesenvolvimentoFullStack>>. Acesso em: 14 fev. 2024.

Red Hat. O que é uma API REST?. Red Hat, 2024. Atualizado 17 de ago. de 2023. Disponível em: <<https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>>. Acesso em: 14 fev. 2024.

Red Hat. O que é CI/CD?. Red Hat, 2024. Atualizado 25 de jan. de 2023. Disponivel em: <<https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd>>. Acessado em: 14 fev. 2024.

SQLite. Datatypes in SQLite. Disponível em: <<https://www.sqlite.org/datatype3.html>>. Acesso: 18 fev. 2024.

Venturus – Construindo novos futuros. Venturus Centro de Inovação Tecnológica, 2024. Disponível em: <<https://www.venturus.org.br/>>. Acesso em: 28 fev. 2024.

Wiggins, Adam. The Twelve-Factor App. Disponível em: <<https://12factor.net/pt_br/>>. Acessado em: 21 fev. 2024.

Higor. Introdução ao Padrão MVC. DevMedia. 2013. Disponível em: <https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308>. Acessado em: 05 mar. 2024.

Lira, Kevin & Rogatis, Jean. O poder dos Micro-Frontends: Introdução a SSR, CSR e IR. AWS. Disponível em: <https://aws.amazon.com/pt/blogs/aws-brasil/o-poder-dos-micro-frontends-introducao-a-ssr-csr-e-ir/>. Acessado em: 05 mar. 2024.

Lenildo. Qualidade de Software - Engenharia de Software 29. DevMedia. Disponivel em: <https://www.devmedia.com.br/qualidade-de-software-engenharia-de-software-29/18209>. Acessado em: 05 mar. 2024.

BECK, Ricardo. Qualidade e testes de software [Aula 01 - Pós-graduação]. Pontifica Universidade Católica RS, Rio Grande do Sul, 2024. 56 páginas. Disponível em: <https://assets-pucrs-priv.pucrs.br/courses/aulas/9555/2vMRQw7b7BEeKhyEvYf8ricardo-beck-aula-01.pdf?Expires=1709596379&Signature=iWK0K3ONvBdZaCPiyapxcxMj1lPk2zl5DvmA8CoEZQT0mRmKrflN7AxDqsxR02j7dFP2VR5GEg23NsCig9nHx5Tbt7DMXTl98xZ2tbuSzqggeuEqnMc49EtYu~I7ooVBem3MszLjQQeWX86JS8Xog8e~WCJI58aKdNgFPDgY5pOcYUU64L2u5iPMU7W~mJQRWnTC8db4~nRFh-Rd96rjt5uWzSIMeC9jDWqeaJbyMNLnqWdmNy-34jMGu-X4AwTGRN3q8rquMJUg4phiueXSsvakzgkIUFbb7Eu~J42hmWjreMc3dA4Wadn4xqx7HxIuycXbLS~iuBqX4AkEWAQeqg__&Key-Pair-Id=KRXNHALY2MG34>. Acesso em: 04 mar. 2024.

Katalon. Disponível em: <https://katalon.com/resources-center/blog/ci-cd-introduction>. Acessado em: 10 mar. 2024.