UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Brenner de Souza Borges

# Desenvolvimento do *back-end* e implantação do sistema online de distribuição de disciplinas para a FACOM

**Uberlândia, Brasil**

**2023**

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Brenner de Souza Borges

# Desenvolvimento do *back-end* e implantação do sistema online de distribuição de disciplinas para a FACOM

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Bruno Augusto Nassif Travençolo

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2023

*Dedico este trabalho a todas as pessoas que me ajudaram a chegar até o momento atual. Minha família, amigos, colegas e, acima de tudo, meus estimados professores, merecem meu profundo reconhecimento e gratidão. Sem o apoio e orientação de vocês, não teria sido possível alcançar este marco em minha jornada acadêmica. Obrigado por todo o suporte e inspiração que me proporcionaram ao longo do caminho.*

# Agradecimentos

Gostaria de expressar minha sincera gratidão aos professores e mentores que me impulsionaram e proporcionaram acesso ao conhecimento. Também sou muito grato aos meus queridos amigos por sempre me apoiarem e não permitirem que eu desistisse dos meus objetivos. A contribuição e o suporte de todos vocês foram fundamentais para o meu crescimento acadêmico e pessoal.

# Resumo

Na Faculdade de Computação da Universidade Federal de Uberlândia (UFU), o Sistema Online de Distribuição de Disciplinas (SODD) é usado para selecionar os professores que ministram as disciplinas a cada semestre, seguindo regras preestabelecidas. O objetivo deste projeto é aperfeiçoar o sistema existente através de uma reescrita completa, usando tecnologias e linguagens mais recentes para aplicações web, e criando uma interface mais amigável. Neste trabalho, o foco principal foi a implementação do *back-end* de determinadas funcionalidades destinadas aos usuários administradores e aos professores, bem como a implantação do sistema no servidor. Também foi oferecido suporte para o desenvolvimento de algumas funcionalidades no *front-end*, como o disponibilização de bibliotecas e também a criação de um ambiente de desenvolvimento para o mesmo. Todas as funcionalidades implementadas já estão presentes no sistema atualmente em uso, havendo assim, apenas uma "tradução"de uma linguagem para outra.

**Palavras-chave**: desenvolvimento, implantação, sistema, online

# Lista de Figuras

# Lista de abreviaturas e siglas

| | |
|---|---|
| HTML | *HyperText Markup Language* - Linguagem de Marcação de Hipertexto |
| CSS | *Cascading Style Sheets* - Folha de Estilo em Cascatas |
| JS | *JavaScript* |
| SVN | *Subversion* |
| CVS | *Concurrent Version System* - Sistema de Versões Concorrentes |
| HTTP | *Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto. |
| HTTPS | *Hypertext Transfer Protocol Secure* - Protocolo de Transferência de Hipertexto Seguro |
| FTP | *File Transfer Protocol* - Protocolo de Transferência de Arquivos |
| SMTP | *Simple Mail Transfer Protocol* - Protocolo Simplificado de Transferência de Correio |
| ASP | *Active Server Pages* - Páginas de Servidor Ativas |
| PHP | *Hypertext Preprocessor* - Pré-Processador de Hipertexto |
| SQL | *Structured Query Language* – Linguagem de consulta estruturada |
| RFC | *Request for Comments* - Pedido para Comentários |
| IP | *Internet Protocol* - Protocolo de Rede |
| TLS | *Transport Layer Security* - Segurança da Camada de Transporte |
| API | *Application Programming Interface* - Interface de Programação de Aplicação |
| URL | *Uniform Resource Locator* - Localizador Uniforme de Recursos |

# Conteúdo

# 1 Introdução

Dentro da Faculdade de Computação (FACOM) da Universidade Federal de Uberlândia (UFU), a distribuição de disciplinas é realizada todos os semestres pela Comissão de Distribuição de Disciplinas (CDD).

A função da comissão é decidir quais disciplinas serão oferecidas no próximo semestre e quais professores ficarão responsáveis. Essa missão em si é muito difícil. Isso ocorre porque a faculdade conta com mais de 60 funcionários com diversas modalidades de trabalho (professores em período integral, professores substitutos, etc.), e é responsável por ministrar aproximadamente 600 horas-aula/semana, em mais de vinte e um cursos de graduação e dois de pós-graduação. Há também o fato de as atribuições dos temas serem determinadas pelo regulamento interno do conselho da FACOM. Essa norma consiste na distribuição dos professores pelas turmas do semestre seguinte com base em uma lista de interesses e preferências. Antes da concretização da distribuição, cada professor define as prioridades das disciplinas que gostaria de ministrar.

A comissão vai então tentar atribuir as disciplinas ao docente seguindo a ordem de prioridade definida por ele. Anteriormente, todo processo era feito de forma manual. A partir de 2015, foi desenvolvido o Sistema Online Distribuição de Disciplinas (SODD) para facilitar o processo.

A implementação de um sistema online para a realização da distribuição de disciplinas da FACOM foi essencial para automatizar processos anteriormente realizados manualmente pelo CDD. Os professores agora podem escolher com rapidez e maior autonomia as prioridades de todas as disciplinas oferecidas em um determinado semestre, de acordo com todas as regras definidas e implementadas no sistema.

Assim como qualquer sistema necessita de manutenção, o SODD não é diferente. O processo de evolução de software é inevitável ser constante, com isso, identificou a necessidade de dar continuidade na melhoria do sistema, com o intuito de reduzir as falhas e implementar novas funcionalidades.

O sistema atual utiliza o *back-end* desenvolvido em JAVA, com o servidor de aplicação *Apache Tomcat* (PROJECT, 2023). No que diz respeito ao *front-end*, ele é composto por HTML, CSS e JS, com destaque para o uso do AngularJS como principal *framework*. Vale ressaltar que a versão utilizada do AngularJS está desatualizada e não recebe mais manutenção da companhia desenvolvedora, a Google (GOOGLE, 2022). Além disso, o sistema incorpora o *framework* CSS *Bootstrap*, desenvolvido e mantido pela Twitter (OTTO, 2023).

## 1.1 Objetivos

### 1.1.1 Objetivo Geral

O objetivo deste projeto consiste em reescrever e implantar o Sistema Online de Distribuição de Disciplinas, adotando tecnologias e linguagens mais atuais, amplamente utilizadas no mercado e com uma comunidade de desenvolvedores mais robusta. Essa abordagem visa facilitar a manutenção do sistema e permitir uma implementação mais eficiente de novas funcionalidades. A motivação para esse projeto é facilitar a manutenção do SODD e permitir uma implementação mais eficiente de novas funcionalidades. Ao utilizar tecnologias em destaque e com suporte ativo, busca-se garantir uma solução escalável e adaptável às demandas contemporâneas da área de ciência da computação.

## 1.2 Objetivos Específicos

- Realizar a análise detalhada do sistema desenvolvido anteriormente, identificando suas limitações, pontos fortes e áreas de melhoria.
- Projetar e implementar as modificações e atualizações necessárias no sistema, levando em consideração as necessidades e requisitos atuais da instituição ou do ambiente em que será implantado.
- Avaliar e otimizar o desempenho do sistema, visando melhorar sua eficiência e capacidade de resposta, garantindo uma experiência de uso mais satisfatória para os usuários.
- Realizar testes abrangentes para verificar a integridade e estabilidade do sistema após as modificações realizadas, identificando e corrigindo eventuais falhas e inconsistências.
- Planejar e executar a implantação do sistema atualizado em um ambiente de produção, seguindo as melhores práticas de gerenciamento de projetos e minimizando possíveis interrupções ou impactos negativos nos processos existentes.
- Realizar a documentação completa do sistema atualizado, fornecendo um guia detalhado para futuras referências e suporte técnico.

Ao alcançar esses objetivos específicos, espera-se fornecer uma solução atualizada, estável e eficiente, contribuindo para a otimização dos processos envolvidos e atendendo às necessidades do ambiente em que o sistema será implantado.

# 2 Metodologia e Tecnologias Utilizadas

Este capítulo apresenta o ambiente e as ferramentas utilizadas no processo de criação do sistema, bem como, os procedimentos envolvidos no decorrer do projeto.

## 2.1 Metodologia

Na parte organizacional foi escolhida para a realização deste trabalho, o método Kanban utilizado em gestão de projetos, apresentando de forma explícita a progressão das atividades, tornando problemas evidentes e cultivando a cultura de melhoria contínua. Para auxiliar com o Kanban, foi usada uma plataforma de gerenciamento de processos, chamada Trello (Figura 1).

Na parte de qualidade de software aplicou-se a metodologia BDD - *Behavior Driven Development* (Desenvolvimento Guiado por Comportamento ou Desenvolvimento Orientado a Comportamento). É uma abordagem colaborativa para o desenvolvimento de software que preenche a lacuna de comunicação entre todos envolvidos no processo, pois utiliza-se uma linguagem natural na criação dos testes, podendo o código ser escrito em Português, ou em outras linguagens, das várias suportadas, o que o torna também de fácil compreensão para todos os membros.

Seguindo a linha de qualidade, agora na parte do código, foi aplicado os princípios SOLID, que são um conjunto de princípios de design de software que visam a criação de código mais robusto, flexível e sustentável. Esses princípios são aplicáveis a qualquer linguagem de programação e podem ser aplicados no desenvolvimento de qualquer tipo de aplicação. Com eles temos benefícios como melhor manutenção e maior flexibilidade no código, assim como a melhoria da legibilidade e compreensão do código (SANTOS, 2018).

Na parte de metodologia na arquitetura do software, foi utilizado o padrão Serviço/Repositório em que a ideia desse padrão é que o código de "nível mais acima" não se preocupe com como os dados são obtidos ou como os negócios são realizados (Figura 2).

Por exemplo, para entender melhor imagine que você está desenvolvendo um aplicativo de *e-commerce*. O código de nível superior desse aplicativo é responsável por mostrar os produtos aos usuários e permitir que eles façam compras. Esse código não precisa saber como os produtos são obtidos do banco de dados ou como o pagamento é processado. Essas tarefas são responsabilidade das camadas de negócios e de dados.

Sendo assim, Repositório é uma maneira de acessar e manipular dados, independentemente da sua origem (RODRIGUES, 2018) e Serviços são uma maneira de separar

as responsabilidades de uma aplicação em diferentes partes, o que torna o código mais fácil de entender, manter e reutilizar (BARBOSA, 2021). Juntamente com isso também foi utilizado a Injeção de Dependência que, ao invés de criar diretamente o *Repository*, criar uma interface, assim podemos reutilizar a camada de repositório em outro lugar do programa.

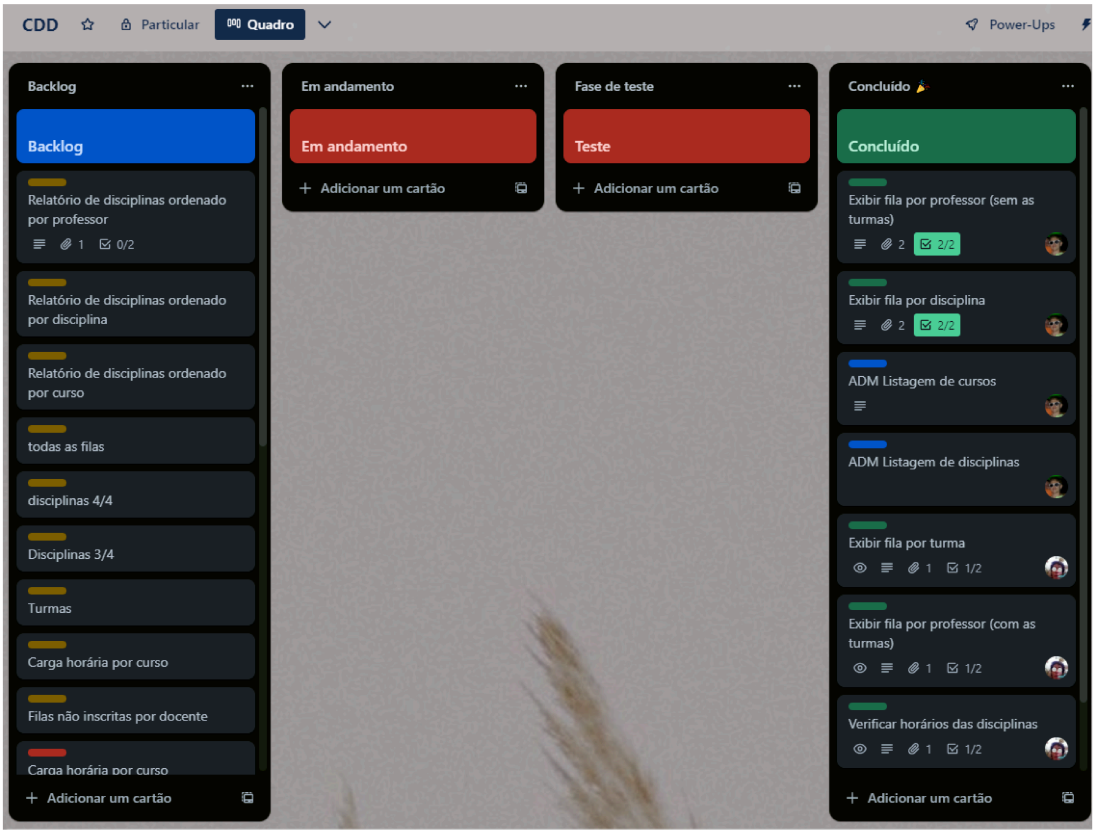


Figura 1 – Kanban utilizado para acompanhamento do projeto (Trello).

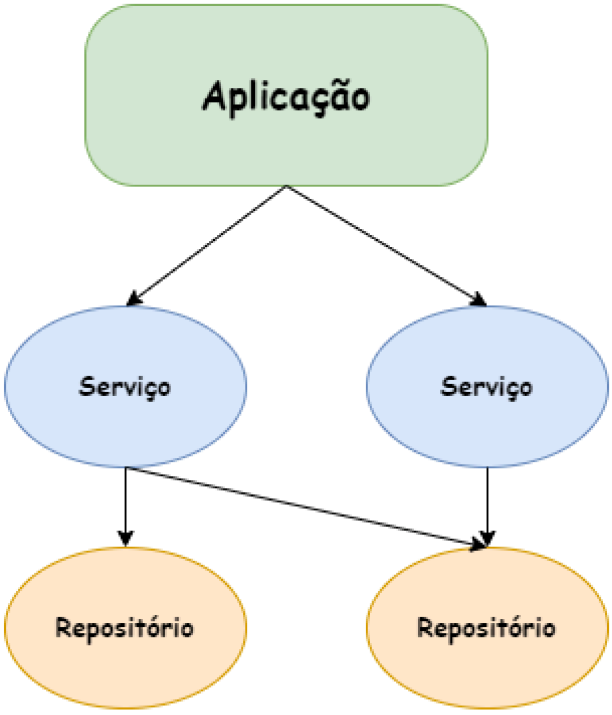


Figura 2 – Diagrama do padrão da arquitetura.

## 2.2 Tecnologias e ferramentas

Durante o processo de desenvolvimento, foram avaliadas e testadas diversas ferramentas, plataformas e tecnologias. O objetivo foi sempre aprimorar o conjunto de soluções que estavam sendo construídas para alcançar os objetivos finais. A seguir descreve-se cada uma das tecnologias e ferramentas escolhidas para o projeto.

### 2.2.1 Docker

Docker é uma plataforma aberta, criada com o objetivo de facilitar o desenvolvimento, a implantação e a execução de aplicações em ambientes isolados, que usam virtualização de nível de sistema operacional, em que o *kernel* do sistema operacional permite que múltiplos processos sejam executados isoladamente no mesmo *host*, para entregar software em pacotes chamados contêineres (INC, 2023a). O objetivo dos contêineres é criar a habilidade de executar diversos processos e aplicações separadamente para utilizar melhor a infraestrutura. Os contêineres são isolados uns dos outros e agrupam seus próprios arquivos de configuração e bibliotecas, mas eles também podem se comunicar uns com os outros por meio de canais estipulados.

### 2.2.2 Git

O Git é um sistema de controle de versão gratuito e de código aberto criado por Linus Torvalds (que também criou o sistema operacional Linux) em 2005. Diferente de outros sistemas de controle de versão centralizados, como SVN e CVS, o Git é distribuído: todo desenvolvedor tem o histórico completo de seu repositório de códigos local. Cada pasta no Git, chamada de diretório de trabalho, é como um arquivo especial que guarda toda a história das mudanças. Essa propriedade é importante pois não é necessário estar sempre conectado à internet ou depender de um servidor central para fazer nosso trabalho. Além disso, o Git não é só distribuído, o que significa que está espalhado por todos os lugares, mas também foi criado pensando em ser rápido, seguro e flexível funcionado como servidor local (CHACON; STRAUB, 2023).

### 2.2.3 IIS

O *Internet Information Services* (IIS) é como o maestro dos servidores web feito pela Microsoft, especialmente para computadores com o sistema operacional *Windows*. Ele é usado para hospedar aplicações web, como sites e *web services*, em um ambiente de servidor. O IIS inclui recursos como autenticação de usuário, gerenciamento de certificados, segurança de rede, gerenciamento de conteúdo e suporte para vários protocolos, como HTTP, HTTPS, FTP e SMTP. Com ele, é possível configurar e cuidar do servidor web de forma bem fácil pois, ele tem uma interface de gerenciamento intuitiva e inclui diversas opções de configuração avançadas. Além disso, ele oferece suporte uma variedade de linguagens de programação, incluindo ASP.NET, PHP e Node.js.

### 2.2.4 Node.js

O Node.js pode ser definido como um ambiente de execução *Javascript server-side* (FOUNDATION, 2023). Isso significa que com o Node.js é possível criar aplicativos em *JavaScript* que podem rodar sozinhos em um computador, sem precisar do navegador como de costume. Isso acontece porque o Node.js junta o ambiente que faz o *JavaScript* funcionar (no caso, o próprio Node.js) com o cérebro que interpreta e executa o *JavaScript*, que é o V8, o mesmo motor inteligente que foi criado pela Google e aplicado pelo *Chrome*.

### 2.2.5 Heroku

A Heroku é uma plataforma *cloud* que permite o *deploy* de aplicações *back-end* de forma rápida e fácil. Ela oferece uma variedade de recursos que tornam o desenvolvimento e a operação de aplicações *back-end* mais eficientes, assim se enquadrando na categoria Plataforma como Serviço (PaaS). Isso significa que a Heroku fornece um ambiente pronto para receber aplicações, incluindo infraestrutura, ferramentas e serviços. Ela oferece su-

porte para uma ampla gama de linguagens de programação, incluindo Java, *Ruby*, PHP, Node.js, *Python*, *Scala* e *Clojure*. A Heroku executa aplicações *back-end* por meio de contêineres virtuais conhecidos como *Dynos*. Os *Dynos* são instâncias de máquinas virtuais que são dimensionadas automaticamente para atender à demanda da aplicação (SALESFORCE, 2021)

### 2.2.6 PostgreSQL

PostgreSQL é um sistema de código aberto de gerenciamento de banco de dados (SGBD) objeto-relacional, em que cada elemento criado é tratado como um objeto, como bancos de dados, tabelas, visualizações, gatilhos, etc. O PostgreSQL foi projetado para ser executado em uma plataforma semelhante ao UNIX. No entanto, o PostgreSQL também foi idealizado para ser portátil, de modo que possa ser executado em muitas plataformas, como *Mac OS X*, *Solaris* e *Windows* (GROUP, 2023).

### 2.2.7 TypeScript

*TypeScript* é um superconjunto tipado da linguagem *JavaScript*, desenvolvido pela *Microsoft*. Isso significa que o *TypeScript* é compatível com o *JavaScript*, mas adiciona recursos e funcionalidades extras à linguagem base. Uma das principais vantagens do *TypeScript* é a adição de um sistema de tipos estático. Isso significa que os tipos são verificados em tempo de compilação, antes que o código seja executado. A verificação de tipos em tempo de compilação pode ajudar a detectar erros comuns, como atribuição incorreta de valores ou chamadas de função inválidas, reduzindo assim a ocorrência de *bugs* e erros durante a execução do software, de forma mais rápida e eficiente.

Além do mais, o *TypeScript* oferece recursos avançados de programação orientada a objetos, como interfaces, herança e polimorfismo, tornando o desenvolvimento mais estruturado, organizado e escalável. Os desenvolvedores podem usar esses recursos para criar código mais modular, reutilizável e de fácil manutenção. Isto é especialmente útil em grandes projetos em que a complexidade e a colaboração entre equipes são desafios importantes (TYPESCRIPT, 2023; MOZDEVNET, 2022).

### 2.2.8 TypeORM

TypeORM é um ORM (Mapeamento Objeto Relacional) baseado em *Typescript* e *Javascript*, projetado para funcionar com uma variedade de bancos de dados relacionais, como PostgreSQL, *MySQL*, *SQLite* e *SQL Server*, e fornece suporte experimental para bancos de dados *NoSQL*, como MongoDB. O TypeORM permite que os desenvolvedores se concentrem na lógica de negócios e na manipulação de objetos, em vez de se preocupar com a complexidade das consultas SQL e especifidades do banco de dados.

Uma das principais vantagens do TypeORM é a capacidade de mapear automaticamente objetos de código para tabelas de banco de dados. Por meio de anotações ou arquivos de configuração é possível definir relacionamentos entre entidades do sistema e seus campos correspondentes no banco de dados. Isso elimina a necessidade de escrever consultas SQL manualmente, permitindo que os desenvolvedores trabalhem em um nível mais alto de abstração e sejam mais produtivos. Ele também fornece uma API consistente para realizar operações CRUD (Criar, Ler, Atualizar, Excluir), simplificando a escrita de consultas e reduzindo a quantidade de código padrão. TypeORM também oferece suporte a transações, migrações de banco de dados e validação de dados, garantindo a integridade dos dados e a segurança das operações realizadas (ORM, 2023).

### 2.2.9 GitHub

O GitHub é uma plataforma de hospedagem de código-fonte que permite aos desenvolvedores e equipes colaborar, gerenciar projetos de software e controlar versões de código. Ele utiliza o sistema de controle de versão Git para rastrear alterações no código, facilitando a colaboração global, o rastreamento de problemas e a automação de tarefas de desenvolvimento, como testes e integração contínua. É amplamente utilizado na comunidade de desenvolvimento de software para hospedar, compartilhar e colaborar em projetos de código aberto e privados (INC, 2023c).

Dentro do Github existe uma variedade de serviços e ferramentas, que auxiliam durante todo o ciclo de desenvolvimento. Podemos destacar algumas como:

Repositórios: Como já citado anteriormente os repositórios são basicamente o lugar onde os desenvolvedores guardam todo o código e arquivos relacionados a um projeto. É como o quartel-general do trabalho em equipe.

Controle de Versão: O GitHub utiliza o Git, um sistema de controle de versão distribuído, para rastrear e gerenciar alterações no código-fonte. Isso permite que várias pessoas trabalhem no mesmo projeto, coordenem suas mudanças e mantenham um histórico detalhado das edições.

Colaboração: Os desenvolvedores podem trabalhar juntos usando algo chamado *"pull requests"* (solicitações de recebimento). Isso é quando alguém sugere uma mudança no código. Assim todos podem conferir e conversar sobre as mudanças e adicionar novidades ou corrigir *bugs*.

Rastreamento de Problemas: O GitHub inclui ferramentas para rastrear e gerenciar problemas, tarefas e solicitações de recursos. Isso ajuda a manter um registro organizado de questões a serem resolvidas no projeto.

Integração Contínua: O GitHub oferece integração com serviços de Integração Contínua (CI) para automatizar testes e construções de código, garantindo que as alterações

não quebrem o software existente.

Hospedagem na Nuvem: Os repositórios do GitHub são hospedados na nuvem, ou seja, em servidores pela internet. Isso quer dizer que dá para acessar tudo de qualquer lugar com internet. Bem prático para compartilhar e trabalhar juntos em projetos de software.

# 3 Referencial teórico

Este capítulo apresenta um breve histórico das entregas do projeto ao decorrer dos anos.

## 3.1 Histórico

- O desenvolvimento do SODD começou em 2015 quando surgiu a necessidade de automatizar o processo de distribuição de disciplinas, já que o mesmo era feito a partir de uma Comissão de Distribuição de Disciplinas (CDD). Foi realizado um levantamento de requisitos funcionais e não funcionais para implementar um sistema baseado na web para apoiar os professores e a FACOM na atribuição de disciplinas (LOCATELLI, 2015).
- Em 2016, foram criados um módulo de administração e um módulo de relatórios para melhorar a interface e a usabilidade, tornando o sistema responsivo. Nesse ano foi criado um repositório git controle de versões, para que ambos os envolvidos conseguissem trabalhar de forma independente e simultânea e para permitir que o orientador controlasse o andamento do trabalho (NAVES, 2016; SILVA, 2016)
- Em 2017 o trabalho continuou nos mesmos moldes dos trabalhos anteriores, fazendo o levantamento de requisitos e manutenções corretivas e evolutivas (OLIVEIRA, 2017).
- Em 2018, foi criado o módulo distribuição de disciplinas. Este módulo implementa basicamente um algoritmo de distribuição de disciplinas, ao contrário de todas as tarefas existentes que gerenciam informações relacionadas aos dados armazenados no sistema. O processo de distribuição é interativo e permite criar e gerenciar cenários, executar distribuição para cada um deles e apresentar os resultados a um comitê que decide quais cenários aceitar. Ao contrário de outros módulos construídos em Java, o módulo de implantação é escrito em C# e integrado ao sistema (OLIVEIRA, 2018).
- Ainda no ano de 2018, deu-se continuidade no processo de levantamento de novos requisitos, implementou-se novas funcionalidades e realização de correções de eventuais problemas e falhas (SANTOS, 2018).
- No ano de 2020, foi montado todo um ambiente de testes automatizados para o projeto, realizando assim uma série de validações referente ao código e a interação do mesmo com o todo. (FERNANDES, 2020).

- No ano de 2021, deu-se continuidade no processo de levantamento de novos requisitos e realização de correções de eventuais problemas e falhas (DAMASCENO, 2021).
- No ano de 2022, iniciou-se o processo de criação de uma nova versão, atualizada e com disponibilidade para a inserção de novos recursos, utilizando o que tem de recente no desenvolvimento *web*, reformulado as tecnologias utilizadas tanto no *back-end*, quanto no *front-end* (MENDES, 2022; MONTE, 2022).

Para um acompanhamento mais detalhado de todo processo de criação e evolução da ferramenta, alvitrar-se verificar as devidas seções nos trabalhos aqui citados.

## 3.2 Versão Atual

Atualmente a Faculdade de Computação da Universidade Federal de Uberlândia, FACOM-UFU, possui o sistema de distribuição de disciplina em funcionamento e em utilização a cada semestre, porém nos dias que correm, a atualização e manutenção do sistema se torna cada vez mais difícil, pois com passar do tempo, o projeto foi ganhando novas funcionalidades e a documentação juntamente com o repasse desses recursos foi minguado, ou seja, com o passar do tempo o software ficou maior e mais robusto, mas sem conhecimento guardado para possíveis manutenções.

Temos também o fato de que com a aplicação de novas funcionalidades outras mais antigas podem ter sido anuladas e não fazem mais sentido. Na atualidade, o sistema possui um *front-end* feito em uma das primeiras versões do *framework* AngularJS, que pelo fato de ser arcaico e diferente de sua versão mais recente dificulta a realização da atualização. Já no *back-end* o sistema utiliza Java na sua versão 1.8.0-151, sendo que a mais recente e com os recursos melhorados da linguagem seria a versão 8.

# 4 Desenvolvimento

## 4.1 Ambientes

O desenvolvimento de software é um processo complexo e abrangente que consiste em diversas etapas para garantir a qualidade e estabilidade da aplicação. A razão pela qual esse processo demora tanto é para garantir que todo o sistema esteja funcionando corretamente antes de ser divulgado ao público (URANO, 2023).

Neste contexto, a separação de ambientes é uma prática fundamental e amplamente reconhecida como um aspecto importante para garantir a qualidade, segurança e eficiência de uma aplicação.

A separação de ambientes refere-se à criação e ao gerenciamento de ambientes distintos para diferentes estágios do ciclo de vida de uma aplicação, como desenvolvimento, homologação e produção. Cada ambiente tem suas próprias configurações, recursos e dados específicos, adaptados à finalidade e às necessidades de cada estágio. A implementação dessa separação é realizada por meio da utilização de servidores e infraestrutura independentes, garantindo independência e isolamento dos ambientes.

Existem diversas razões pelas quais a separação de ambientes é de extrema importância em uma aplicação:

- **Segurança**: Ao separar o ambiente de produção de outros ambientes, como ambientes de desenvolvimento e homologação, minimizamos os riscos de exposição a erros e vulnerabilidades que possam comprometer a segurança do sistema. Alterações não testadas ou experimentais são isoladas dos dados e operações reais, garantindo a integridade dos dados e a estabilidade do ambiente de produção.

- **Testes e Validação**: A separação de ambientes permite que testes e validações sejam realizados em um ambiente específico sem afetar a funcionalidade e a disponibilidade do ambiente de produção. Isso permite que as equipes de desenvolvimento e teste executem cenários e casos de teste de maneira controlada, identificando e corrigindo problemas antes que afetem os usuários finais.

- **Desenvolvimento Iterativo**: O ambiente separado permite adotar uma abordagem de desenvolvimento flexível e iterativa. As equipes podem trabalhar em novos recursos e correções em um ambiente de desenvolvimento isolado, sem interromper a experiência do usuário de produção. Essa abordagem permite que você implemente

atualizações gradualmente, permitindo que cada versão seja testada e aprovada antes de ser disponibilizada aos usuários.

- **Escalabilidade e Performance**: O isolamento do ambiente permite que cada ambiente dimensione os seus recursos de forma independente. Isso significa que você pode adaptar o processamento, o armazenamento e a capacidade de rede de cada ambiente às suas necessidades específicas. Dessa forma, você pode garantir que seu aplicativo seja escalonável e possa lidar com uma variedade de requisitos de carga de trabalho.

- **Manutenção e Diagnóstico**: Ambientes separados facilitam a manutenção, as atualizações e a solução de problemas. Podemos aplicar correções ou melhorias ao seu ambiente específico sem afetar diretamente as suas operações de produção. A separação também permite que *logs* e registros de cada ambiente sejam separados e analisados com mais precisão, permitindo melhor detecção e diagnóstico de problemas.

No projeto descrito neste trabalho foi adotada uma estratégia ágil e simplificada onde apenas três ambientes foram construídos: desenvolvimento, homologação e produção. Isso porque quanto mais ambientes tivermos, mais tempo o processo pode demorar e maior o risco de falha. Na sequência serão detalhados cada um deles.

### 4.1.1 Ambiente de Desenvolvimento

Imagine que você está prestes a construir uma casa. Antes de começar a erguer as paredes e pintar as paredes, é necessário ter um local adequado para planejar e montar tudo, certo? Quando criamos um novo projeto ou entramos em uma nova equipe, precisamos alterar o código do projeto para adicionar as funcionalidades que combinamos com a equipe. Isso sempre é feito na nossa própria máquina.

Assim, em nossas máquinas, o projeto é codificado e atualizado sem preocupação com possíveis erros, e com liberdade para experimentar possibilidades criativas. Mas para fazer isso, esse ambiente deve estar separado do resto do time de desenvolvimento e das outras maquinas que fazem parte da aplicação nas demais etapas.

Esses ambientes separados são conhecidos como ambientes de desenvolvimento e são isolados e não integrados especificamente com ferramentas externas. Como os desenvolvedores utilizam seus próprios computadores para trabalhar nos projetos, este é o ambiente onde eles podem interagir mais e concentrar seu tempo sem correr o risco de impactar o sistema em produção. Você pode criar e testar novos recursos, solucionar problemas e garantir que tudo esteja funcionando conforme esperado, como relatado em (URANO, 2023).

O ambiente de desenvolvimento é como uma oficina para os desenvolvedores de software. É um espaço dedicado onde eles podem criar, experimentar e transformar suas ideias em realidade. É um local controlado e isolado, projetado para facilitar a tarefa de escrever, testar e depurar código.

Um bom ambiente de desenvolvimento fornecerá as ferramentas certas para os desenvolvedores. Essas ferramentas simplificam o trabalho e aumentam a produtividade do desenvolvedor.

Nesse projeto utilizamos o Docker, para auxílio na virtualização e criamos o ambiente a partir dele. Basicamente temos um arquivo para o Docker *Compose* realizar a orquestração dos nossos contêineres, chamado docker-compose.yml e outro chamado Dockerfile em que tem-se a arquitetura em si do contêiner. A seguir é mostrado um exemplo de um dos arquivos utilizados no projeto, no caso, o docker-compose.yml, em que o mesmo possui dois serviços, um para o *back-end* e outro para o banco de dados. Esses dois serviços são criados dentro da mesma rede, gerida pelo Docker. Também é possível executar os contêineres dos serviços de maneira independente entre si ou de forma conjunta, de acordo com a necessidade. Analisando o serviço *back-end*, pode-se verificar as seguintes configurações: *container_name: back-end*: Define o nome do contêiner como *"back-end"*.

*build*: Especifica como construir a imagem do contêiner. O contexto é o diretório atual ("."), e o Dockerfile a ser usado é o "Dockerfile".

*restart: always*: Garante que o contêiner seja reiniciado automaticamente em caso de falha ou reinicialização do sistema.

*environment*: Define variáveis de ambiente para o contêiner. Neste caso, há uma variável chamada DATABASE_URL que contém informações de conexão ao banco de dados PostgreSQL.

*volumes*: Mapeia volumes entre o *host* e o contêiner. Isso é útil para persistir dados ou para evitar sobrescrever certos diretórios no contêiner. No exemplo, o diretório /app/node_modules é mapeado, e o diretório ./src do *host* é montado no /app/src do contêiner.

*ports*: Mapeia a porta do *host* para a porta do contêiner. O parâmetros PORT:-3333 é usada, o que significa que se a variável PORT estiver definida, ela será usada; caso contrário, será usada a porta padrão 3333. Isso permite uma maior flexibilidade ao executar o contêiner em diferentes ambientes.

Para o serviço do banco de dados temos basicamente as mesmas configurações, com exceção de que ao invés de termos um Dockerfile para ele, utilizamos um registrado no servidor Docker, descrito em *image: postgres:14.4-alpine*

```
1 version: "3.9"
```

```
services:
  db:
    container_name: postgres_db
    image: postgres:14.4-alpine
    restart: always
    ports:
      - "${TYPEORM_PORT:-5432}:5432"
    environment:
      - POSTGRES_USER=${POSTGRES_USER:-postgres}
      - POSTGRES_DB=${POSTGRES_DB:-sodd}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD:-1234}
      - PGDATA=/var/lib/postgresql/data
    volumes:
      - ./postgresql/data:
            /var/lib/postgresql/data
      - ./docker_postgres_init.sql:
          /docker-entrypoint-initdb.d/postgres_init.sql
  backend:
    container_name: backend
    build:
      context: .
      dockerfile: Dockerfile
    restart: always
    environment:
      - DATABASE_URL=postgres://
            postgres:1234@host.docker.internal:5432/sodd
    volumes:
      - /app/node_modules
      - ./src:/app/src
    ports:
      - "${PORT:-3333}:3333"
```

A seguir temos o Dockerfile utilizado no projeto, para termos exemplo de como é sua estrutura.

```
FROM node:18.6.0-alpine3.15
RUN mkdir -p /app/src && chown -R node:node /app/src
WORKDIR /app
COPY package*.json ./
COPY .env ./
```

```
6  COPY tsconfig.json ./
7  RUN yarn install
8  COPY --chown=node:node . .
9  EXPOSE 3333
10 CMD ["yarn", "dev"]
```

Verificando as instruções desse arquivo DockerFile, temos que:

*FROM node:18.6.0-alpine3.15*: Define a imagem base para a construção, usando a imagem oficial do Node.js na versão 18.6.0, baseada no Alpine Linux 3.15. Isso fornece uma imagem leve e eficiente para executar aplicações Node.js.

*RUN mkdir -p /app/src && chown -R node:node /app/src*: Cria um diretório /app/src dentro do contêiner e define o proprietário como o usuário *"node"*. Isso é feito para garantir que o processo Node.js, que será executado mais tarde, tenha permissão para acessar e escrever neste diretório.

*WORKDIR /app*: Define o diretório de trabalho dentro do contêiner como /app. Isso indica onde os comandos subsequentes serão executados.

*COPY package*.json ./*: Copia os arquivos *package.json* e *package-lock.json* do diretório de construção do *host* para o diretório de trabalho do contêiner (/app). Isso é feito antes de instalar as dependências para aproveitar o cache do Docker, evitando reinstalar as dependências se o arquivo package.json não foi alterado.

*COPY .env ./*: Copia o arquivo .env do diretório de construção para o diretório de trabalho do contêiner. O arquivo .env geralmente contém variáveis de ambiente para configurar a aplicação.

*COPY tsconfig.json ./*: Copia o arquivo de configuração *TypeScript* tsconfig.json para o diretório de trabalho do contêiner.

*RUN yarn install*: Executa o comando *yarn install* para instalar as dependências do Node.js. Este comando é executado após copiar os arquivos package.json para garantir que as dependências estejam atualizadas.

*COPY –chown=node:node . .*: Copia todos os arquivos restantes do diretório de construção para o diretório de trabalho do contêiner, definindo o proprietário como o usuário "node". Isso inclui os arquivos do código-fonte da aplicação.

*EXPOSE 3333*: Informa ao Docker que a aplicação dentro do contêiner estará ouvindo na porta 3333. Isso não publica automaticamente a porta no *host*, mas é uma documentação para outros desenvolvedores sobre qual porta expor.

*CMD ["yarn", "dev"]*: Define o comando padrão que será executado quando o contêiner for iniciado. Neste caso, o comando é *yarn dev*, que geralmente é usado para

iniciar o servidor de desenvolvimento do Node.js.

Para o *front-end* foi utilizado os mesmos recursos e conceito, com a única diferença é que seu arquivo docker-compose.yml possui apenas um serviço, sendo este para o *front-end* em si da aplicação.

### 4.1.2 Ambiente de Homologação

Você já pensou em lançar um produto sem testá-lo? Isso seria arriscado, certo? Da mesma forma, durante o desenvolvimento de software, é importante ter um ambiente específico para realizar os testes finais antes de liberar a aplicação aos usuários. Este ambiente é denominado "ambiente de homologação".

Portanto, o ambiente de homologação (ou *Staging*), é também um ambiente de testes que se diferencia do estágio anterior por proporcionar uma verificação mais profunda do produto final esperado do software, conforme mencionado anteriormente, mas com menor exposição ao usuário final. Esta é a fase final de testes antes do lançamento oficial da aplicação, como dito em (URANO, 2023).

Neste ambiente, os desenvolvedores e testadores têm a oportunidade de verificar se todos os recursos funcionam corretamente, se quaisquer *bugs* ou comportamentos inesperados que possam ocorrer durante o uso real do aplicativo foram corrigidos e se o desempenho atende às expectativas. Você também pode executar testes de integração, testes de carga, testes de segurança e outros tipos de testes relacionados para verificar a qualidade e o desempenho da sua aplicação.

Outro benefício de um ambiente de homologação é a capacidade de envolver os usuários finais ou *stakeholders* importantes no processo de teste. Isso permite coletar *feedback* valioso e identificar possíveis melhorias ou ajustes antes do lançamento oficial. O envolvimento do usuário final também ajuda a validar se a aplicação atende às suas necessidades e expectativas.

Podemos comparar o ambiente de homologação a um ensaio geral de uma peça de teatro. Antes de subir ao palco e encantar o público, é necessário realizar ensaios para ajustar detalhes, garantir que todos os atores estejam preparados e que as cenas sejam executadas de forma harmoniosa. Da mesma forma, no ambiente de homologação, é possível simular situações reais, testar cenários complexos e garantir que a aplicação esteja pronta para ser lançada.

Para esse projeto esse projeto foi utilizado a plataforma Heroku como ambiente de homologação (Figura 3), em que foi utilizado os seus serviços em nuvem para montar o *back-end*, o *front-end* e também o banco de dados PostgreSQL com disponibilidade de acesso de qualquer lugar similar à versão final, mas com recursos limitados e não escaláveis, mas com acesso restringido. Para se criar o que o Heroku chama de *Dyno*, que

é basicamente um *pod* de aplicação, ou melhor, é a infraestrutura em nuvem que nossa aplicação utilizará, é necessário criar um arquivo na raiz do projeto chamado Procfile , em que a sintaxe do mesmo lembra a sintaxe YAML, como mostrado abaixo.

```
1 web: npm run start:prod
```

Com o arquivo criado, basta entrar na página inicial da plataforma e ir em *'New'* depois em *'Create new app'* e atribui-se um nome para o seu app e seleciona a região onde o mesmo será hospedado. Em seguida seleciona-se a opção *'Deploy'* na barra superior, vai em *'Deployment method'* e seleciona o *'GitHub'* e após se autenticar no mesmo, clica em *'Deploy Branch'* e observa-se os logs para validar se sua aplicação irá subir corretamente. Todo esse processo é detalhado na documentação oficial disponível em (HEROKU, 2022).

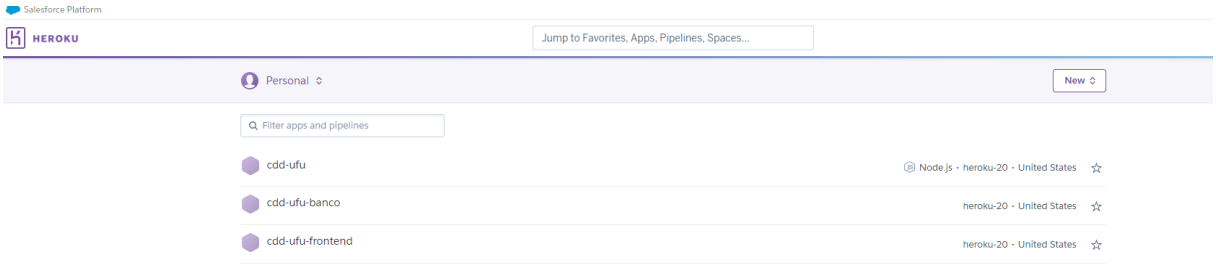


Figura 3 – Página inicial da Heroku.

Um ponto importante sobre a Heroku é que no começo do projeto, a plataforma possuía um plano para criação de recursos de forma gratuita, porém, atualmente o mesmo plano não existe mais, entretanto temos algumas outras oportunidade presentemente, a primeira delas é a utilização do programa de eco *Dynos*, que são basicamente o mesmo ambiente do antigo programa de gratuidade, porém para se inscrever nele é preciso registrar um cartão de crédito, como segurança para a plataforma. A outra opção, sendo está a em prática, seria utilizar-se de créditos dentro da plataforma, a partir do programa de recursos para alunos e professores do Github (INC, 2023b) que disponibiliza trezentos e doze dólares para serem utilizado na plataforma, o que daria uma utilização por pelo menos dois anos, pois o *Dyno Basic* custa hoje sete dólares por mês.

### 4.1.3 Ambiente de Produção

Imagine que você está prestes a apresentar um espetáculo para uma plateia ansiosa. Você ensaiou, testou todos os detalhes e agora é o momento de brilhar no palco. No desenvolvimento de software, o ambiente de produção é como esse palco, onde a aplicação é lançada oficialmente para os usuários finais.

O ambiente de produção é o ambiente final em que a aplicação é implantada e executada para os usuários utilizarem. É o palco principal, onde tudo acontece em tempo

real. Nesse ambiente, a aplicação está disponível para os usuários acessarem, interagirem e utilizarem as funcionalidades oferecidas.

Esta etapa é considerada por muitos o ambiente mais importante pois é onde os testes finais ocorrem com os utilizadores da aplicação.

Uma das principais características de um ambiente de produção é a estabilidade. Ao contrário dos ambientes de desenvolvimento ou homologação, onde são realizados ensaios e testes, o ambiente de produção é projetado para uso da aplicação no mundo real, por isso é importante que o ambiente seja robusto, estável e capaz de lidar com a carga de usuários e transações.

No ambiente de produção, é necessário considerar aspectos como a escalabilidade, desempenho, segurança e disponibilidade. A aplicação deve ser capaz de lidar com um número crescente de usuários e demandas sem comprometer sua performance.

No projeto desenvolvido, o ambiente de produção ficou em disponibilidade em um servidor *on-premise*, ou seja, uma máquina física local, situada na própria universidade, com o sistema operacional *Windows Server* 2022, com recursos de oito gigabytes de memória e um processador com quatro núcleos. Além disso, se faz necessário a ativação de certos recursos do sistema operacional, mais especificamente o *Internet Information Services* e ter instalado o Node.js, mas trataremos disso nos próximos tópicos. A Figura 4 ilustra os ambientes apresentados nessa seção.

Para executar a aplicação em produção, basta montar e exportar o código da mesma com o auxílio do gerenciador de pacotes para o Node.js, NPM, e mapeia-lo no IIS.

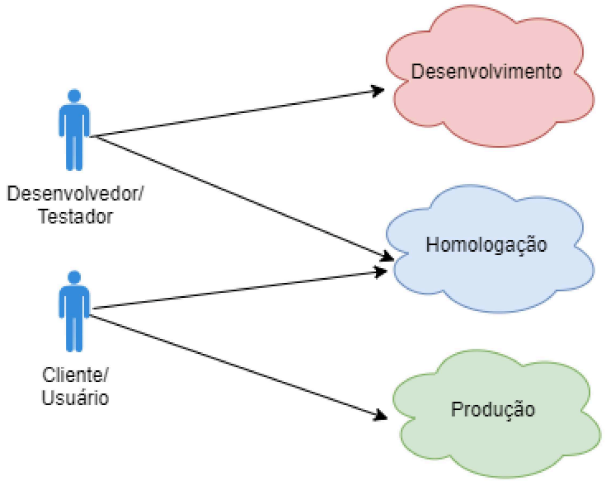


Figura 4 – Diagrama dos ambientes.

## 4.2 Segurança

Quando se trata de hospedar um servidor *on-premise* em uma rede fechada, ou seja, uma rede local restrita e de acesso controlado, o *Windows Server* 2022 pode oferecer uma série de recursos e medidas de segurança que ajudam a proteger os dados e os recursos da empresa.

O *Windows Server* 2022 é um sistema operacional projetado especificamente para ambientes de servidor, oferecendo uma série de funcionalidades que promovem a segurança dos dados e a confiabilidade da infraestrutura.

### 4.2.1 JWT

Segundo (AUTH0, 2022) um *JSON Web Token* (JWT) é um padrão aberto (RFC 7519) que define um formato compacto e autocontido para representar informações de forma segura. Ele consiste em três partes principais: um cabeçalho (*header*), uma carga útil (*payload*) e uma assinatura (*signature*). O cabeçalho especifica o tipo de *token* e o algoritmo de assinatura utilizado. A carga útil contém as informações que são transportadas pelo *token*, como dados do usuário ou reivindicações adicionais. A assinatura é gerada a partir do cabeçalho, da carga útil e de uma chave secreta, garantindo a integridade dos dados.

Todas os *endpoints* criados para a API foram feitos seguindo boas práticas e padrões de desenvolvimento web em *JavaScript*, que por sua vez garante um nível de segurança, porem para melhorar ainda mais essa segurança, foi implementado também a utilização de JWT em cada uma das rotas desses *endpoints*.

A principal vantagem do JWT é a sua capacidade de autenticação *stateless* (sem estado). Isso significa que o servidor não precisa armazenar informações sobre o estado de autenticação do cliente. Em vez disso, as informações necessárias para autenticar e autorizar o usuário serão incluídas no próprio *token*. Isso simplifica a implantação e a escalabilidade do sistema porque não há necessidade de armazenar sessões ou consultar um banco de dados para validar cada solicitação.

Os JWTs também fornecem uma maneira segura de compartilhar informações entre diferentes partes do sistema. Os *tokens* são assinados digitalmente usando uma chave secreta ou um par de chaves assimétricas, de forma que apenas partes confiáveis possam ler e verificar as informações contidas no *token*. Impede a manipulação, adulteração ou falsificação de dados durante a comunicação entre sistemas.

O uso de JWT em APIs de um sistema *REST* oferece várias vantagens significativas em termos de segurança:

*Autenticação*: O JWT permite a autenticação segura do usuário. Quando um usuá-

rio faz login no sistema, ele recebe um *token* JWT contendo informações de identificação e autenticação. Esse *token* é enviado em todas as solicitações subsequentes e permite que o servidor verifique a identidade do usuário sem a necessidade de armazenar informações de sessão no servidor.

*Segurança de Dados*: As informações contidas no JWT são assinadas digitalmente e não podem ser adulteradas ou modificadas durante a transmissão. Você também pode criptografar *tokens* para garantir a confidencialidade dos dados, ocultando dados confidenciais de carga útil.

*Autorização*: O JWT pode conter informações sobre direitos e privilégios do usuário. Isso permite que o servidor verifique se o usuário está autorizado a acessar determinados recursos ou executar determinadas ações.

*Redução do Tempo de Processamento*: Como o JWT contém informações relevantes no próprio *token*, o servidor não precisa consultar o banco de dados em cada solicitação para verificar a identidade e as permissões do usuário. Isso reduz o tempo de processamento e torna a API mais eficiente.

*Escalabilidade*: Usar JWT em sua API melhora a escalabilidade do sistema. Como os *tokens* são autocontidos e autênticos, o servidor não precisa armazenar informações de sessão na memória ou em um banco de dados, facilitando o escalonamento horizontal.

*Cross-Origin Resource Sharing (CORS)*: O JWT é especialmente útil para permitir acesso a APIs em diferentes domínios (*Cross-Origin*) devido ao seu formato de *token* e capacidade de transportar informações autenticadas.

### 4.2.2 IIS

Conforme apontado anteriormente, para este projeto optamos por hospedar nossa aplicação em IIS e um dos pontos levados em consideração na escolha foram a segurança. É possível usá-lo para hospedar com segurança seus sites e serviços graças ao isolamento automático de aplicação, que permite que cada aplicação hospedado no servidor seja executado em seu próprio ambiente isolado, evitando que uma aplicação prejudique a segurança ou o desempenho de outros programas.

Como dito em (MORRIS, 2022), há também uma configuração de *sandbox* padrão que restringe o acesso e as permissões do aplicativo, portanto, essa abordagem de *sandboxing* garante a segurança do aplicativo e do servidor, restringindo o acesso a recursos críticos do sistema. Além disso, o espaço do servidor é reduzido, resultando em uma superfície de ataque menor. Menos componentes expostos reduzem vulnerabilidades e potencial de ataque.

Também inclui cache dinâmico e compactação aprimorada para aumentar a velo-

cidade do site. Além disso, a plataforma modular permite que os desenvolvedores criem módulos personalizados para expandir sua funcionalidade. O IIS é diferente de outros servidores web pois está totalmente incorporado ao sistema operacional (SO) *Windows*. Isso significa que o IIS consegue aproveitar todos os recursos de segurança integrados ao *Windows*, como *Active Directory* e *Group Policy*.

Outro ponto, seguindo a linha de recursos de segurança incorporado ao *Windows* é que o IIS vem com recursos integrados de autenticação, autorização e controle de acesso para fortalecer a segurança de seus aplicativos da web. É possível gerar contas separadas para administradores de sistemas e aplicativos para níveis de acesso granulares.

Recursos de segurança adicionais incluem filtragem de solicitações para tráfego de lista de permitidos/bloqueados, bloqueio dinâmico de IP, criptografia SSL e TLS, registro de solicitações, compactação de site e controles de segurança dedicados para FTP. Mais informações podem ser encontradas na documentação oficial disponibilizada pela Microsoft em (COMMUNITY, 2023).

# 5 API implementadas

Foi realizado um mapeamento de todas as funcionalidades disponíveis atualmente no sistema e montados um quadro no Trello dividindo-as em quatro grupos de acordo com as relações. O módulos são os seguintes:

1. Módulo Professor, onde nesse módulo o usuário consegue entrar em uma fila para concorrência de disciplina além de conseguir ver quais os demais professores estão concorrendo na mesma, também é possível ver as demais disciplinas os concorrentes estão, além da sua grade de horários entre outras coisas, dentro desse escopo.

2. Módulo ADM, nesse módulo o usuário tem acesso máximo ao sistema, conseguindo acessar todas as funcionalidade, mas em exclusividade o painel administrativo onde consegue ativar ou desativar funcionalidades do sistema, além de conseguir editar turmas, disciplinas e filas, entre outras funcionalidades

3. Módulo Relatório, aqui nesse módulo o usuário consegue gerar diversos relatórios sobre as filas, disciplinas, turmas e cursos.

4. Módulo Levantamento, este módulo o usuário tem acesso a dados mais estáticos relacionados ao negócio, como a disciplina mais disputada, a taxa de troca de professores nas disciplinas, entre outras.

Ao primeiro momento foi implementado oito APIs sendo seis do Módulo Professor e totalizando o mesmo, e duas do Módulo ADM. As APIs são as seguintes:

- ***Relação fila, professores e turma***.

  Para os professores no aplicativo desenvolvido, é fundamental saber quem está cadastrado em uma disciplina e listar quais as demais disciplinas os demais professores estão. Esse *endpoint* é responsável por isso, usando como parâmetro o número geral da turma, ao ser chamado via verbo *GET*, é retornado a lista dos professores cadastrados juntamente com uma lista com as demais disciplinas que o mesmo tem registro, caso tenha sucesso.

  *Exemplo de chamada:*

```
1 GET /fila_new/turma/<nro_turma> HTTP/1.1
2 Host: sodd-backend-dbcb044a61ba.herokuapp.com
3 Authorization: Bearer <token>
```

  *Exemplo de retorno:*

```
[
    {
        "id_turma": 1904,
        "id_fila": 22770,
        "prioridade": 2,
        "created_at": "2022-05-29T16:51:42.513Z",
        "fila": {
            "id": 22770,
            "siape": "411740",
            "codigo_disc": "FACOM31302",
            "pos": 1,
            "prioridade": 2,
            "qte_ministrada": 1,
            "qte_maximo": 4,
            "ano": 2021,
            "semestre": 1,
            "status": -1,
            "periodo_preferencial": false,
            "created_at": "2022-05-29T16:50:38.803Z",
            "professor": {
                "siape": "123456␣␣",
                "nome": "Jose␣da␣Silva",
                "data_ingresso": "1988-12-05T00:00:00.000Z",
                "data_nasc": "1988-03-16T00:00:00.000Z",
                "afastado": false,
                "regime": "de",
                "carga_atual": 12,
                "locacao": "udi",
                "cnome": "Jose",
                "data_saida": null,
                "data_exoneracao": null,
                "data_aposentadoria": null,
                "status": "Ativo",
                "created_at": "2022-05-28T12:26:02.845Z"
            },
            "disciplina": {
                "codigo": "FACOM31302",
                "nome": "Programacao␣de␣Computadores",
                "ch_teorica": 2,
```

```
                "ch_pratica": 2,
                "ch_total": 4,
                "curso": "FIS.LICENC",
                "temfila": true,
                "periodo": 3,
                "cod_antigo": "GFC005",
                "created_at": "2022-05-28T12:50:52.961Z"
            }
        }
    }
]
```

- ***Relação fila, professores e semestre***.

  Assim como é importante a relação fila, professores e turma para um usuário de nível administrativo dentro da aplicação desenvolvida a relação fila, professores e semestre também segue no mesmo aspecto, uma vez que a disciplina e o curso estão intimamente ligados. Este *endpoint* é o responsável por isso, usando como parâmetro o siape e o número geral do semestre, quando chamado via verbo *GET*, é retornada a lista de cursos cadastrados com seu código, e se está ativo ou não, caso tenha sucesso.

  *Exemplo de chamada:*

```
GET /fila_new/professor/<siape>/semestre/<nro_semest>
    HTTP/1.1
Host: sodd-backend-dbcb044a61ba.herokuapp.com
Authorization: Bearer <token>
```

  *Exemplo de retorno:*

```
{
        "id_turma": 1886,
        "id_fila": 22516,
        "prioridade": 1,
        "created_at": "2022-05-29T16:51:42.513Z",
        "fila": {
            "id": 22516,
            "siape": "123456",
            "codigo_disc": "GSI529",
            "pos": 3,
            "prioridade": 1,
            "qte_ministrada": 0,
```

```
13              "qte_maximo": 4,
14              "ano": 2021,
15              "semestre": 1,
16              "status": -1,
17              "periodo_preferencial": false,
18              "created_at": "2022-05-29T16:50:38.803Z",
19              "professor": {
20                  "siape": "123456 ",
21                  "nome": "Jose da Silva",
22                  "data_ingresso": "2014-01-14T00:00:00.000Z",
23                  "data_nasc": "1984-09-15T00:00:00.000Z",
24                  "afastado": false,
25                  "regime": "de",
26                  "carga_atual": 12,
27                  "locacao": "mc",
28                  "cnome": "Jose",
29                  "data_saida": null,
30                  "data_exoneracao": null,
31                  "data_aposentadoria": null,
32                  "status": "Ativo",
33                  "created_at": "2022-05-28T12:26:02.845Z"
34              },
35              "disciplina": {
36                  "codigo": "GSI529",
37                  "nome": "MC -Sistemas Distribuidos",
38                  "ch_teorica": 60,
39                  "ch_pratica": 0,
40                  "ch_total": 60,
41                  "curso": "MC-BSI     ",
42                  "temfila": true,
43                  "periodo": -1,
44                  "cod_antigo": null,
45                  "created_at": "2022-05-28T12:50:52.961Z",
46                  "curso_disciplinas": {
47                      "codigo": "MC-BSI     ",
48                      "nome": "Sistemas de Informacao",
49                      "unidade": "FACOM",
50                      "campus": "mc",
51                      "permitir_choque_periodo": true,
```

```
                    "permitir_choque_horario": true,
                    "created_at": "2022-05-28T12:47:43.837Z"
                }
            }
        }
    ]
```

- ***Exibir fila por disciplina.***

  Um dos usos que o cliente da aplicação desenvolvida consegue utilizar é poder visualizar a fila de professores para ministrar certa disciplina, além das demais filas que aquele professor também está registrado, apenas digitando o nome da disciplina na caixa de pesquisa. Este *endpoint* é encarregado disso, usando como parâmetro o código da disciplina e o semestre. Quando chamado com o verbo *GET*, retorna a lista de docentes registrados nela, juntamente com o nome, a posição de cada docente na fila, a prioridade do mesmo e a quantidade mínima e máxima de vezes ministrada, além da lista das demais informação detalhada do docente, caso tenha sucesso.

```
GET /fila/disciplina/<cod_disc>/semestre/<nro_semestre>
        HTTP/1.1
Host: sodd-backend-dbcb044a61ba.herokuapp.com
Authorization: Bearer <token>
```

*Exemplo de retorno:*

```
[
    {
        "id": 22770,
        "siape": "123456",
        "codigo_disc": "FACOM31302",
        "pos": 1,
        "prioridade": 2,
        "qte_ministrada": 1,
        "qte_maximo": 4,
        "ano": 2021,
        "semestre": 1,
        "status": -1,
        "periodo_preferencial": false,
        "created_at": "2022-05-29T16:50:38.803Z",
        "professor": {
            "siape": "123456  ",
            "nome": "Jose da Silva",
```

```
            "data_ingresso": "1988-12-05T00:00:00.000Z",
            "data_nasc": "1988-03-16T00:00:00.000Z",
            "afastado": false,
            "regime": "de",
            "carga_atual": 12,
            "locacao": "udi",
            "cnome": "Jose",
            "data_saida": null,
            "data_exoneracao": null,
            "data_aposentadoria": null,
            "status": "Ativo",
            "created_at": "2022-05-28T12:26:02.845Z"
        }
]
```

- ***Exibir fila por professor (sem as turmas).***

  O usuário final, pode verificar de maneira simples em quais filas para disciplinas ele está concorrendo e além dos outros professores da mesma disciplina apenas digitando o nome dele na caixa de pesquisa. Com esse *endpoint* conseguimos isso, tendo como parâmetro o siape do docente e o número geral do semestre, e ao ser chamado via verbo *GET*, é retornado as informações da fila, a lista com as informações dos docentes cadastrados nela juntamente com as informações da disciplina, caso tenha sucesso.

  *Exemplo de chamada:*

```
GET /fila/professor/<siape>/semestre/<nro_semest> HTTP/1.1
Host: sodd-backend-dbcb044a61ba.herokuapp.com
Authorization: Bearer <token>
```

  *Exemplo de retorno:*

```
[
    {
        "id": 21835,
        "siape": "123456",
        "codigo_disc": "FACOM49010(V)",
        "pos": 1,
        "prioridade": 2,
        "qte_ministrada": 2,
        "qte_maximo": 6,
        "ano": 2021,
```

```
        "semestre": 1,
        "status": -1,
        "periodo_preferencial": true,
        "created_at": "2022-05-29T16:50:38.803Z",
        "professor": {
            "siape": "2297590 ",
            "nome": "Jose da Silva",
            "data_ingresso": "2009-11-24T00:00:00.000Z",
            "data_nasc": "1988-04-12T00:00:00.000Z",
            "afastado": false,
            "regime": "de",
            "carga_atual": 8,
            "locacao": "udi",
            "cnome": "Jose",
            "data_saida": null,
            "data_exoneracao": null,
            "data_aposentadoria": null,
            "status": "Ativo",
            "created_at": "2022-05-28T12:26:02.845Z"
        },
        "disciplina": {
            "codigo": "FACOM49010(V)",
            "nome": "Programacao de Computadores",
            "ch_teorica": 4,
            "ch_pratica": 0,
            "ch_total": 4,
            "curso": "ENG.MECATR",
            "temfila": true,
            "periodo": 1,
            "cod_antigo": null,
            "created_at": "2022-05-28T12:50:52.961Z",
            "curso_disciplinas": {
                "codigo": "ENG.MECATR",
                "nome": "Engenharia Mecatronica",
                "unidade": "FEMEC",
                "campus": "udi",
                "permitir_choque_periodo": false,
                "permitir_choque_horario": false,
                "created_at": "2022-05-28T12:47:43.837Z"
```

```
50                }
51            }
52        }
53    ]
```

- ***Exibir restrições***.

  Os usuários, que são docentes, e que estão atualmente ministrando alguma disciplina já possuem horários reservados aos quais não podem se dispor para alguma outra atividade. Pela aplicação podemos verificar quais são esses períodos na grade de horários desse docente. A partir desse *endpoint* conseguimos isso, tendo como parâmetro o siape do docente e ao ser chamado via verbo *GET*, é retornado os dias e suas posições na grade de horários, caso for sucesso.

  *Exemplo de chamada:*

```
1 GET /restricoes/professor/<siape> HTTP/1.1
2 Host: sodd-backend-dbcb044a61ba.herokuapp.com
3 Authorization: Bearer <token>
```

  *Exemplo de retorno:*

```
1  [
2      {
3          "siape": "123456",
4          "dia": "5",
5          "letra": "h"
6      },
7      {
8          "siape": "123456",
9          "dia": "5",
10         "letra": "j"
11     }
12 ]
```

- ***Verificar horários das disciplinas***.

  Nem sempre o usuário final conseguirá ministrar duas ou mais disciplinas a qual ele deseja, um dos fatores para que isso ocorra é o conflito de horário destas disciplinas. Uma maneira fácil de validar se é possível ministrar duas ou mais disciplinas é por meio do grade de horários disponível no sistema. Este *endpoint* é incumbido disso, tendo como parâmetros o número geral do semestre e um ou mais siapes dos docentes, e ao ser chamado via verbo *GET*, é retornado o nome da disciplina e sua posição na grade de horários, caso for sucesso.

*Exemplo de chamada:*

```
GET /ministra/professor/semestre/<nro_semestre>?
        siapes[]=123456 HTTP/1.1
Host: sodd-backend-dbcb044a61ba.herokuapp.com
Authorization: Bearer <token>
```

*Exemplo de retorno:*

```
[
    {
        "siape": "1234546 ",
        "id_turma": 1775,
        "created_at": "2022-05-28T13:07:51.533Z",
        "turma": {
            "id": 1775,
            "codigo_disc": "GBC036",
            "turma": "C",
            "ch": 4,
            "ano": 2021,
            "semestre": 1,
            "created_at": "2022-05-28T13:04:54.350Z"
        },
        "ofertas": [
            {
                "id": 6891,
                "dia": "4",
                "letra": "a",
                "id_turma": 1775,
                "created_at": "2022-05-29T16:31:43.065Z"
            },
            {
                "id": 6892,
                "dia": "4",
                "letra": "b",
                "id_turma": 1775,
                "created_at": "2022-05-29T16:31:43.065Z"
            },
            {
                "id": 6893,
                "dia": "6",
```

```
                "letra": "a",
                "id_turma": 1775,
                "created_at": "2022-05-29T16:31:43.065Z"
            },
            {
                "id": 6894,
                "dia": "6",
                "letra": "b",
                "id_turma": 1775,
                "created_at": "2022-05-29T16:31:43.065Z"
            }
        ]
    }
]
```

- ***Exibir fila por professor (com as turmas).***

  O docente que no caso da aplicação desenvolvida é o usuário final, pode verificar de maneira simples em quais filas para disciplinas ele está concorrendo e além da turma dessa mesma disciplina apenas digitando o nome dele na caixa de pesquisa. Esse *endpoint* é responsável por isso, tendo como parâmetro o siape do docente, o ano e o semestre, e ao ser chamado via verbo *GET*, é retornado uma lista com o nome da disciplina, a turma dessa disciplina, o código da mesma, a posição posição do docente na fila para ela, a prioridade do mesmo e a quantidade miníma e máxima de vezes ministrada, caso for sucesso.

  *Exemplo de chamada:*

```
GET /fila/professor/<siape>/ano/<ano>/semestre/<semest>
    HTTP/1.1
Host: sodd-backend-dbcb044a61ba.herokuapp.com
Authorization: Bearer <token>
```

  *Exemplo de retorno:*

```
[
    {
        "posicao": 7,
        "prioridade": 16,
        "qte_ministrada": 0,
        "qte_maximo": 4,
        "turma": "I␣",
        "codigo_disciplina": "FACOM39702",
```

```
        "nome_disciplina": "Inteligencia Artificial"
    },
    {
        "posicao": 13,
        "prioridade": 15,
        "qte_ministrada": 0,
        "qte_maximo": 6,
        "turma": "S ",
        "codigo_disciplina": "GSI005",
        "nome_disciplina": "Logica para Computacao"
    }
]
```

- ***Exibir fila por turma***.

  Uma das funcionalidades do SODD é o usuário docente conseguir ver de forma rápida e simples a disposição da fila para uma determinada disciplina e turma, do período corrente e com isso o mesmo pode realizar seu planejamento de solicitações de forma mais inteligente. Utilizando esse *endpoint* podemos obter isso, tendo como parâmetros o código da fila, ao ser chamado via verbo *GET*, é retornado uma lista com os nomes dos docentes na fila da disciplina, juntamente com a posição na mesma, a prioridade e a quantidade mínima e máxima de vezes ministrada para tal, caso for sucesso.

  *Exemplo de chamada:*

```
GET /fila/turma/<nro_fila> HTTP/1.1
Host: sodd-backend-dbcb044a61ba.herokuapp.com
Authorization: Bearer <token>
```

  *Exemplo de retorno:*

```
[
    {
        "posicao": 23,
        "prioridade": 16,
        "qte_ministrada": 0,
        "qte_maximo": 4,
        "nome_professor": "Rafael Pasquini"
    },
    {
        "posicao": 34,
        "prioridade": 30,
```

```
12          "qte_ministrada": 0,
13          "qte_maximo": 4,
14          "nome_professor": "Shigueo␣Nomura"
15      }
16  ]
```

# 6 Hospedagem/Implantação

Para a hospedagem foi escolhido o sistema IIS (*Internet Information Services*) da Microsoft.

### 6.0.1 Instalação

Chamar de "instalação" não é tão preciso. Afinal, o IIS é um recurso do *Windows*. O que você realmente precisa fazer é habilitar o recurso. No *Windows* 10, você pode pressionar a tecla *Windows*[5] e digitar *"turn win"*. Você deve ver "Ativar ou desativar recursos do *Windows*" [6]. Selecione isso para abrir a caixa de diálogo "Recursos do *Windows*" [7]. A partir daqui, é possível ativar todos os tipos de recursos adicionais. Basta clicar na caixa da opção "Serviços de Informações da Internet" para obter um bom ponto de partida com as mínimas configurações pré-definidas. Quando a instalação estiver concluída, é possível acessar a GUI [8] do IIS digitando "IIS" na pesquisa do *Windows* ou executando "inetmgr" em um terminal de comando.

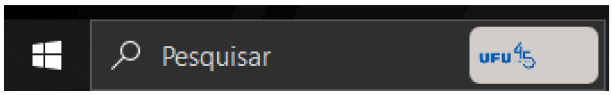


Figura 5 – Barra de busca do *Windows*

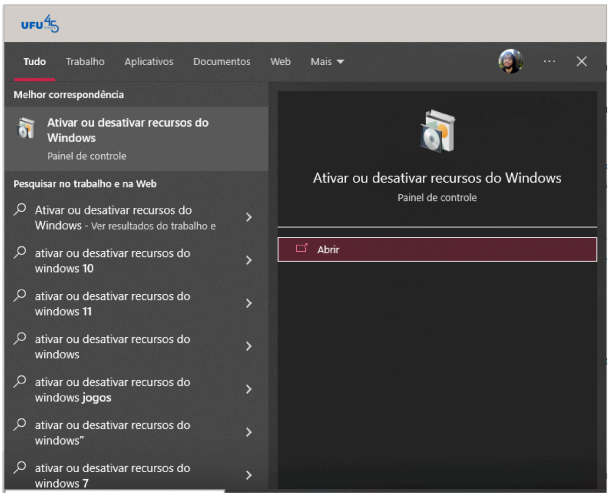


Figura 6 – Resultado da busca

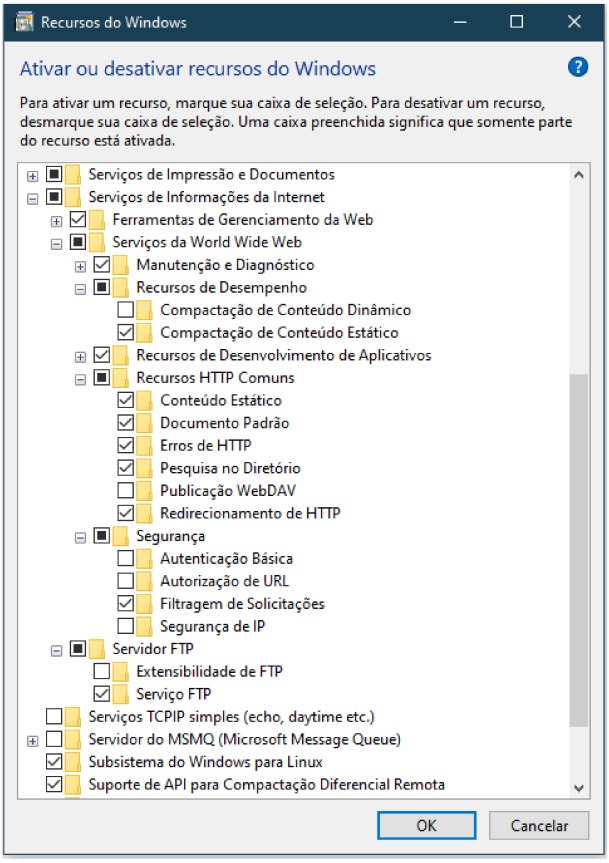


Figura 7 – Local de ativação do IIS

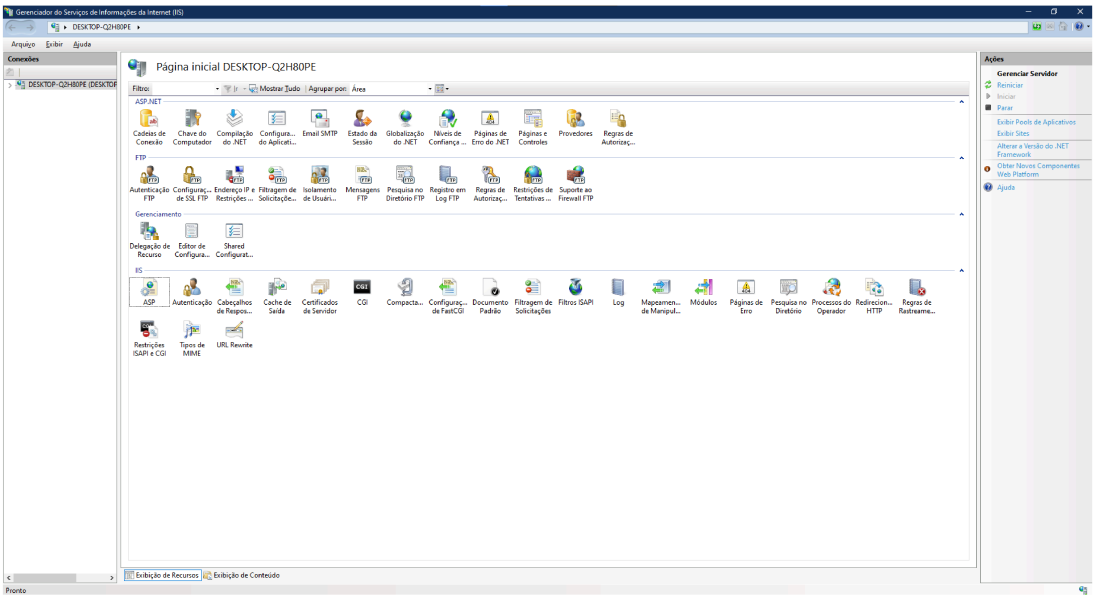


Figura 8 – Interface do IIS

No *Windows Server*, existe o painel do gerenciador do servidor (Figura 9). A partir dele tem-se acesso ao painel de recurso 'Gerenciar', 'Adicionar Funções e Recursos' (Figura 10). Na janela que se abre, selecionamos a opção 'Instalação baseada em função

ou recurso' e clicamos em próximo (Figura 11). A seguir selecionamos o servidor que instalaremos os recursos e clicamos em próximo (Figura 12). Na página sequente selecionamos as funções que queremos, no caso basta clicar na caixa '*Web Server* (IIS)' e depois em próximo (Figura 13). Agora escolhemos os recursos relacionados as função anteriormente selecionadas que gostaríamos de adicionar, sendo que os essenciais já estão selecionado, e clicamos em próximo (Figura 14). Agora estamos na página de resumo do que foi selecionado, e após conferir os mesmos, clicamos por fim, em instalar.

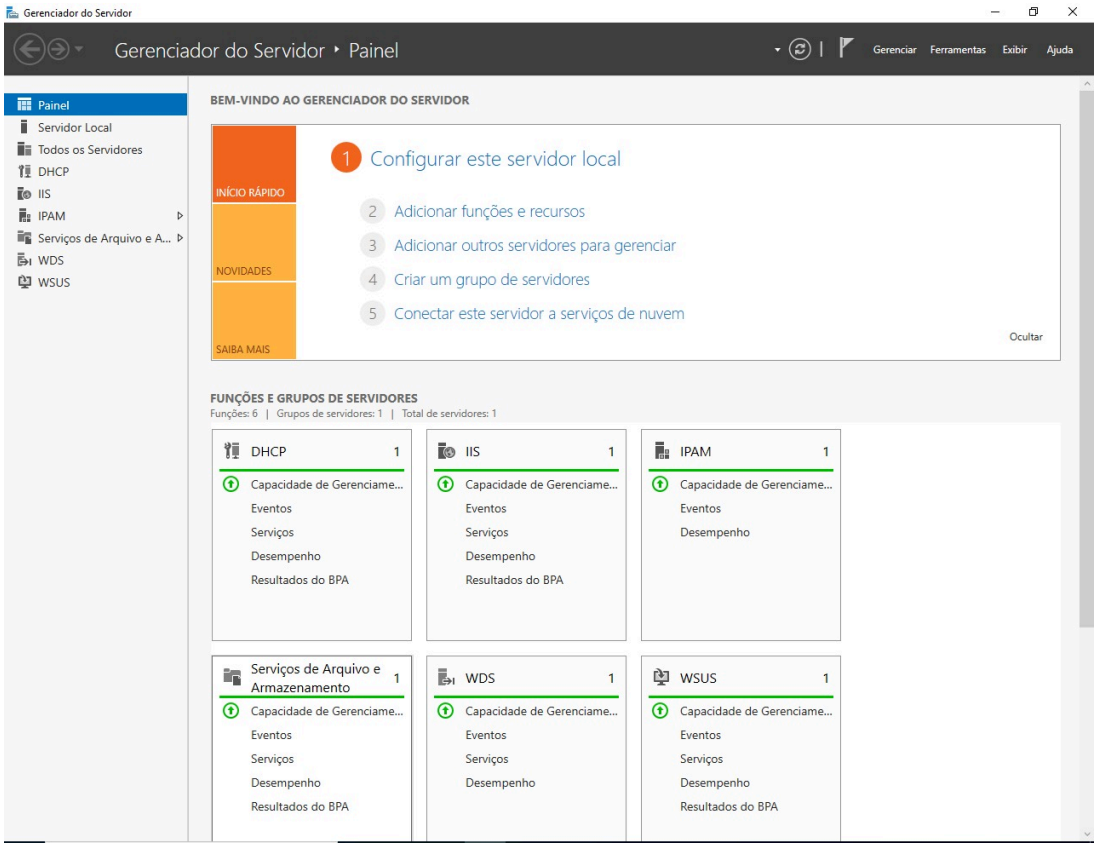


Figura 9 – Painel do gerenciador do servidor.

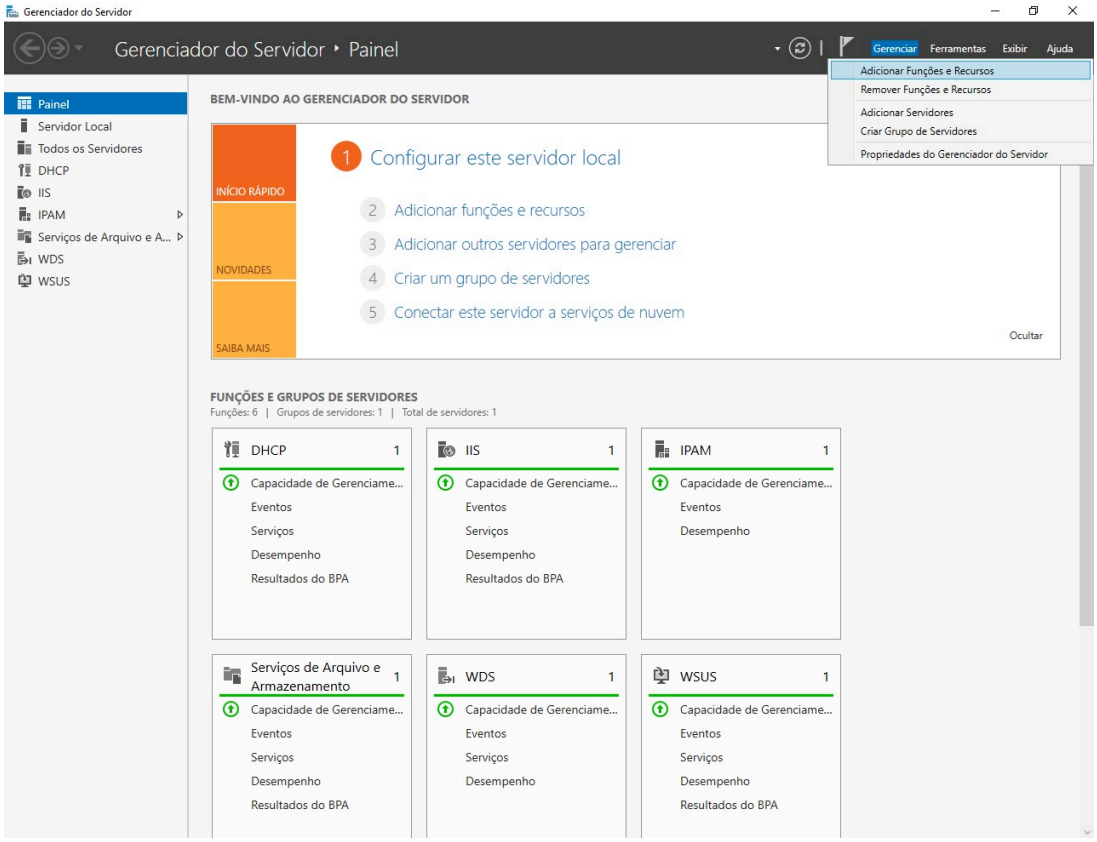


Figura 10 – Botão Gerenciar.

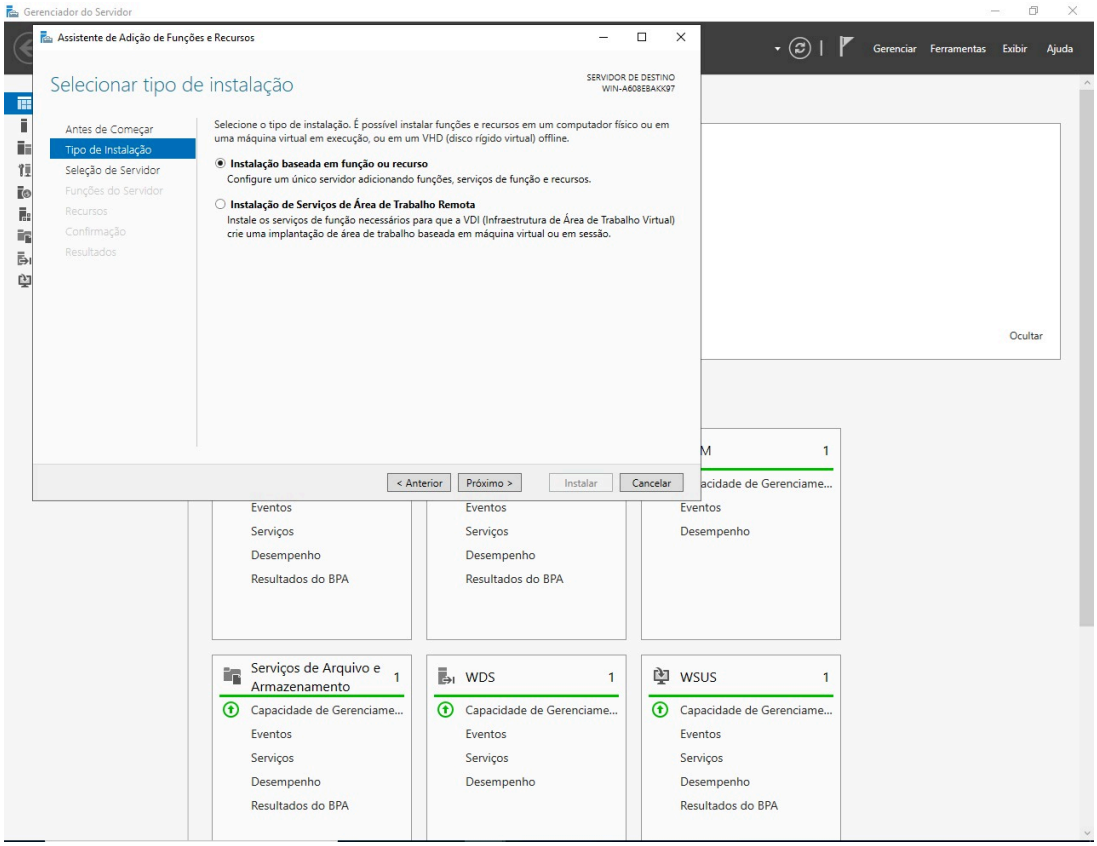


Figura 11 – Tipo de instalação.

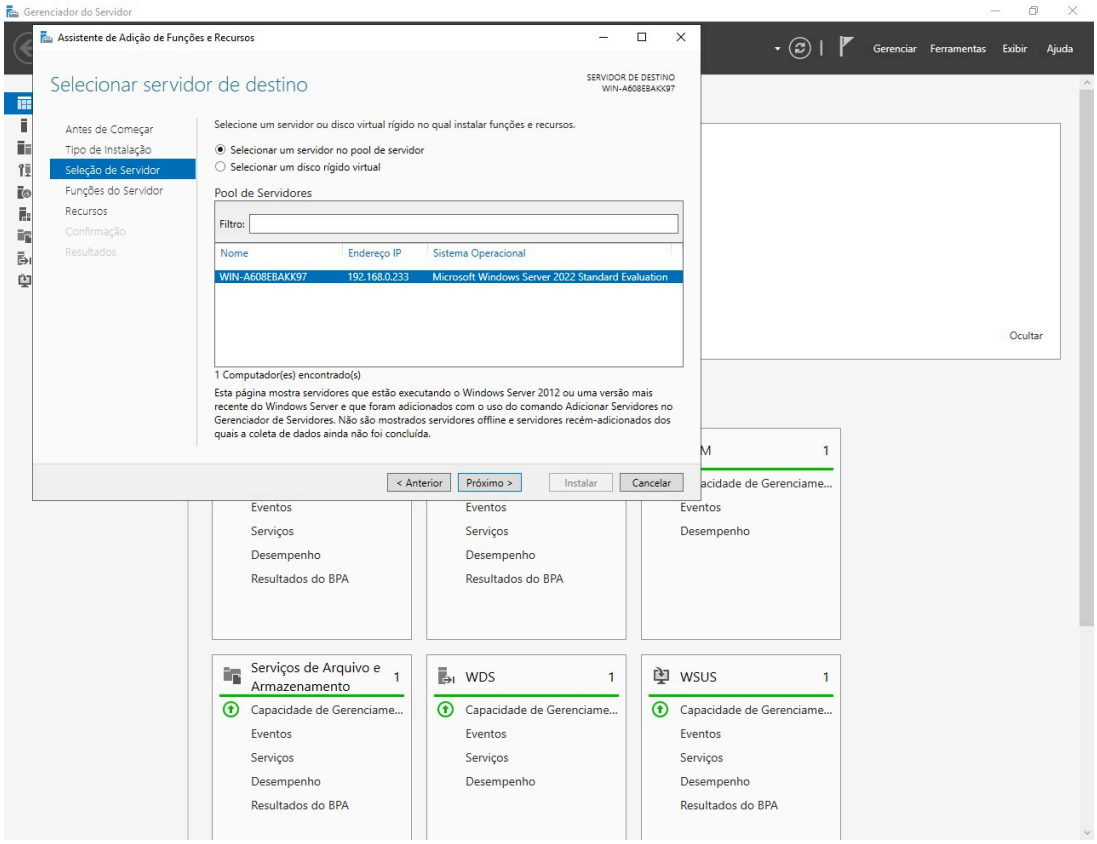


Figura 12 – Seleção do servidor.

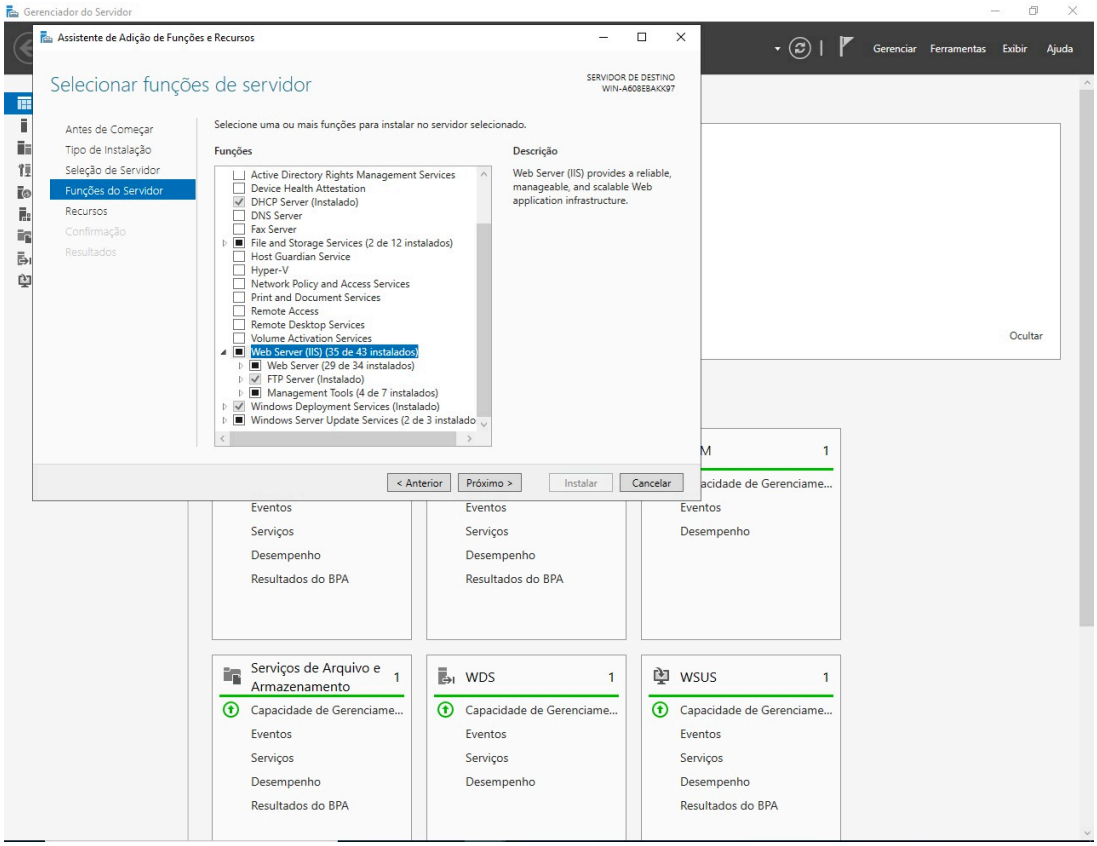


Figura 13 – Seleção de função.

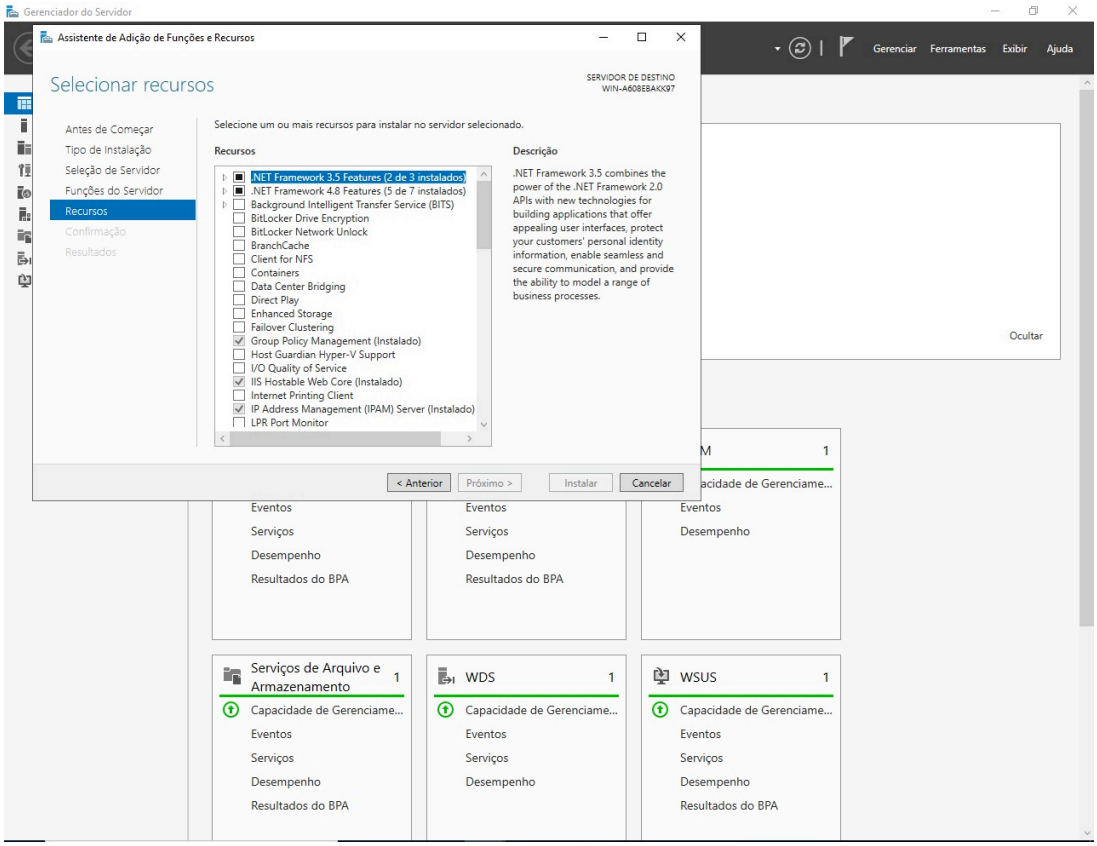


Figura 14 – Seleção de recursos.

Como a aplicação desenvolvida neste projeto tem como núcleo a linguagem *JavaScript*, são necessárias algumas outras ferramentas para aplicar subir a aplicação no IIS. São dois módulos novos, o URL *Rewrite* e o IISNode, além é claro do próprio interpretador para o *JavaScript*, o *Node.js*. Todas essas ferramentas extras ao IIS são *open-source*.

Primeiramente o *Node.js* pode ser adquirido em seu site oficial, para instalar no *Windows* basta executar o arquivo .msi após o download. Agora o URL *Rewrite*, que também pode ser baixado diretamente do seu site oficial, e sua instalação também é simples, basta executar o .msi após o download. Já o IISNode que é um módulo personalizado criado pela comunidade e mantido atualmente pela Microsoft, que pode ser adquirido via repositório no GitHub, onde também encontra-se uma documentação sobre o mesmo. Sua instalação também é simplificada, onde basta executar o arquivo do *Windows Installer*, após baixá-lo.

Além disso, é necessário atribuir permissão ao usuário responsável pela aplicação no IIS definido em 'Pool de aplicativos', criar *threads* Node.js, além de acessar e ler o repositório do projeto. Essas tarefas podem ser feitas no terminal de comando *PowerShell* por meio dos seguintes comandos:

```
1 icacls.exe C:\Program Files\nodejs\ /grant "IIS_IUSRS:(OI)(CI)F"
2 icacls.exe C:\Users\Administrator\ /grant "IIS_IUSRS:(OI)(CI)F"
```

Embora você possa executar código *JavaScript* diretamente por meio do Node.js no ambiente de execução (*JavaScript Runtime*), há várias vantagens em usar o IIS junto com o módulo IISNode para hospedar aplicativos na dada linguagem de programação em comparação com um processo node.exe auto-hospedado, como dito por Scott Hanselman (criador do módulo) em (HANSELMAN, 2011), tais como:

- *Gerenciamento de processos*: O módulo IISNode cuida do gerenciamento da vida útil dos processos node.exe, simplificando a melhoria da confiabilidade geral. Não é preciso implementar infraestrutura para iniciar, parar e monitorar os processos.
- *Escalabilidade em servidores multi-core*: Como o node.exe é um processo de encadeamento único, ele escala apenas para um núcleo da CPU. O módulo IISNode permite a criação de vários processos node.exe por aplicativo e balanceia a carga do tráfego HTTP entre eles, permitindo a utilização total da capacidade da CPU de um servidor sem exigir código de infraestrutura adicional de um desenvolvedor de aplicativos.
- *Atualização automática*: O módulo IISNode garante que, sempre que o aplicativo Node.js for atualizado (ou seja, o arquivo de *script* for alterado), os processos node.exe sejam reciclados. As solicitações em andamento podem terminar normalmente a execução usando a versão antiga do aplicativo, enquanto todas as novas solicitações são despachadas para a nova versão do aplicativo.
- *Acesso aos logs por HTTP*: O módulo IISNode fornece acesso à saída do processo node.exe (por exemplo, gerado por chamadas console.log) via HTTP. Esse recurso é fundamental para ajudá-lo a depurar aplicativos Node.js implantados em servidores remotos.
- *Lado a lado com outros tipos de conteúdo*: O módulo IISNode integra-se com o IIS de uma forma que permite que um único site contenha uma variedade de tipos de conteúdo. Por exemplo, um único site pode conter um aplicativo Node.js, arquivos HTML e JavaScript estáticos, aplicativos PHP e aplicativos ASP.NET. Isso permite escolher as melhores ferramentas para o trabalho em questão, bem como a migração progressiva de aplicativos existentes.
- *Mudanças mínimas no código do aplicativo NodeJS*: O módulo IISNode permite a hospedagem de aplicativos HTTP Node.js existentes com alterações mínimas. Normalmente, tudo o que é necessário é alterar o endereço listado do servidor HTTP para um fornecido pelo módulo IISNode por meio da variável de ambiente *process.env.PORT*.

- *Experiência de gestão integrada*: O módulo IISNode é totalmente integrado ao sistema de configuração do IIS e usa as mesmas ferramentas e mecanismos que outros componentes do IIS para configuração e manutenção.

Além dos benefícios específicos do módulo IISNode, hospedar aplicativos Node.js no IIS permite que o desenvolvedor se beneficie de diversos recursos do IIS, entre eles:

- Compartilhamento de porta (hospedando vários aplicativos HTTP na porta 80)
- Segurança (HTTPS, autenticação e autorização)
- Reescrita de URL
- Compressão
- Cache

### 6.0.2 Criação de um site

Anteriormente foi descrito o processo de instalação do IIS e das ferramentas necessárias para a aplicação do projeto desenvolvido, agora será descrito a parte de criação do site, utilizando o que foi realizado no processo anterior. Primeiramente deve-se definir o arquivo de configuração do site para o IIS ao qual ele verificará no primeiro momento em que um site é executado, esse arquivo é chamado de *web.config*. A seguir tem-se o modelo utilizado para o projeto, lembrando que são duas aplicações hospedas no IIS, uma para o *front-end* e outra para o *back-end*. Esse arquivo utiliza a formatação estilo XML e deve ficar na pasta raiz do projeto.

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3  <configuration>
4      <appSettings>
5      </appSettings>
6      <system.webServer>
7          <iisnode node_env="%node_env%"
8              nodeProcessCountPerApplication="1"
9              maxConcurrentRequestsPerProcess="1024"
10             maxNamedPipeConnectionRetry="200"
11             namedPipeConnectionRetryDelay="500"
12             maxNamedPipeConnectionPoolSize="512"
13             maxNamedPipePooledConnectionAge="30000"
14             asyncCompletionThreadCount="0"
15             initialRequestBufferSize="4096"
16             maxRequestBufferSize="65536"
17             uncFileChangesPollingInterval="50000"
```

```
18              gracefulShutdownTimeout="600000"
19              loggingEnabled="true"
20              logDirectory="iisnode"
21              debuggingEnabled="true"
22              debugHeaderEnabled="false"
23              debuggerPortRange="5058-6058"
24              debuggerPathSegment="debug"
25              maxLogFileSizeInKB="128"
26              maxTotalLogFileSizeInKB="1024"
27              maxLogFiles="20"
28              devErrorsEnabled="true"
29              flushResponse="false"
30              enableXFF="false"
31              promoteServerVars=""
32              configOverrides="iisnode.yml"
33              watchedFiles="web.config;*.js"
34              nodeProcessCommandLine="C:\Program␣Files\nodejs\node.exe"/>
35          <handlers>
36              <add name="iisnode" path="src/shared/infra/http/server.js"
37                  verb="*" modules="iisnode"/>
38          </handlers>
39
40          <rewrite>
41              <rules>
42                  <rule name="NodeInspector" patternSyntax="ECMAScript"
43                        stopProcessing="true">
44                      <match url="^server.js\/debug[\/]?"/>
45                  </rule>
46                  <rule name="StaticContent" patternSyntax="Wildcard">
47                      <action type="Rewrite"
48                              url="public/{R:0}"
49                              logRewrittenUrl="true"/>
50                      <conditions>
51                          <add input="{REQUEST_FILENAME}"
52                               matchType="IsFile"
53                               negate="true"/>
54                      </conditions>
55                      <match url="*.*"/>
56                  </rule>
57                  <rule name="DynamicContent">
58                      <conditions>
59                          <add input="{REQUEST_FILENAME}"
60                               matchType="IsFile"
61                               negate="True"/>
62                      </conditions>
63                      <action type="Rewrite"
64                              url="src/shared/infra/http/server.js"/>
```

```
65                  </rule>
66                  <rule name="SocketIO" patternSyntax="ECMAScript">
67                      <match url="socket.io.+"/>
68                      <action type="Rewrite"
69                              url="src/shared/infra/http/server.js"/>
70                  </rule>
71              </rules>
72          </rewrite>
73          <directoryBrowse enabled="false"/>
74      </system.webServer>
75  </configuration>
```

Com o arquivo *web.config* criado e com as configurações necessárias para o site devidamente preenchidas, os passos para criar um novo site no IIS são simples, segue os mesmos:

1. Inicie o Gerenciador do IIS (Figura 15).
2. No painel Conexões, clique com o botão direito do mouse no nó Sites na exibição em árvore e clique em Adicionar site.
3. Na caixa de diálogo Adicionar site, digite um nome amigável para o site na caixa Nome do site (Figura 16).
4. Se desejar selecionar um pool de aplicativos diferente daquele listado na caixa Pool de aplicativos. Na caixa de diálogo Selecionar pool de aplicativos, selecione um pool de aplicativos na lista Pool de aplicativos e clique em OK .
5. Na caixa Caminho físico, digite o caminho físico da pasta do site da Web ou clique no botão Procurar (...) para navegar no sistema de arquivos e localizar a pasta.
6. Se o caminho físico inserido na etapa 5 for para um compartilhamento remoto, clique em Conectar como para especificar as credenciais que têm permissão para acessar o caminho. Se você não usar credenciais específicas, selecione a opção Usuário do aplicativo (autenticação de passagem) na caixa de diálogo Conectar como.
7. Selecione o protocolo para o site da Web na lista Tipo.
8. O valor padrão na caixa de endereço IP é *All Unassigned*. Se você precisar especificar um endereço IP estático para o site da Web, digite o endereço IP na caixa Endereço IP.
9. Digite um número de porta na caixa de texto Porta.
10. Opcionalmente, digite um nome de cabeçalho de *host* para o site na caixa Cabeçalho de *host*.

11. Se você não precisar fazer nenhuma alteração no site e quiser que o site esteja disponível imediatamente, marque a caixa de seleção Iniciar site imediatamente.

12. Clique em OK.

E pronto, um novo site está apto a ser executado e gerenciado via IIS.

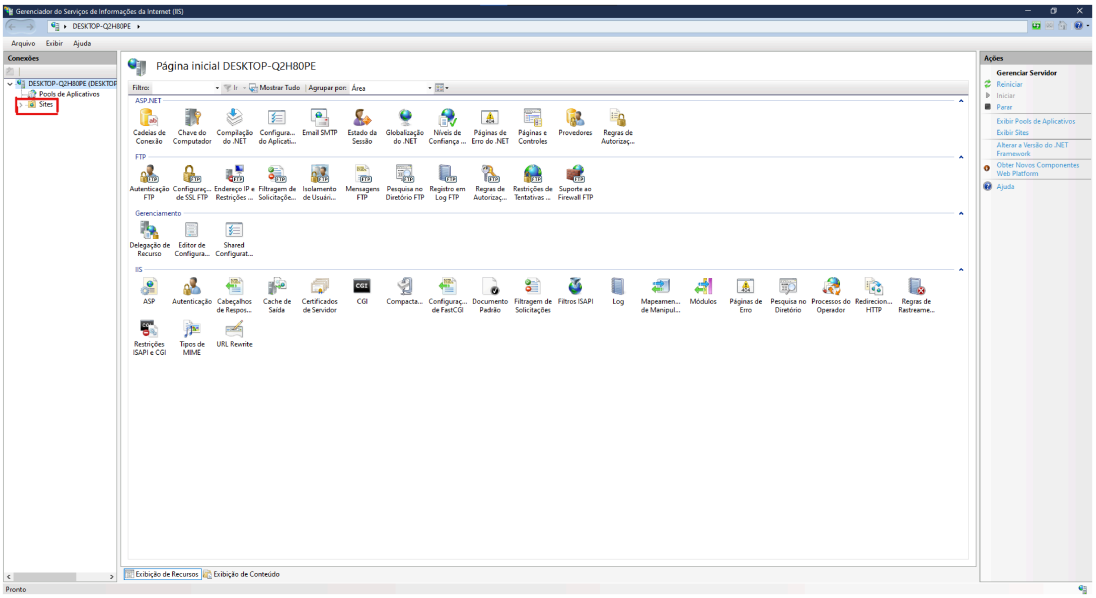


Figura 15 – Primeiro passo para criar um site.

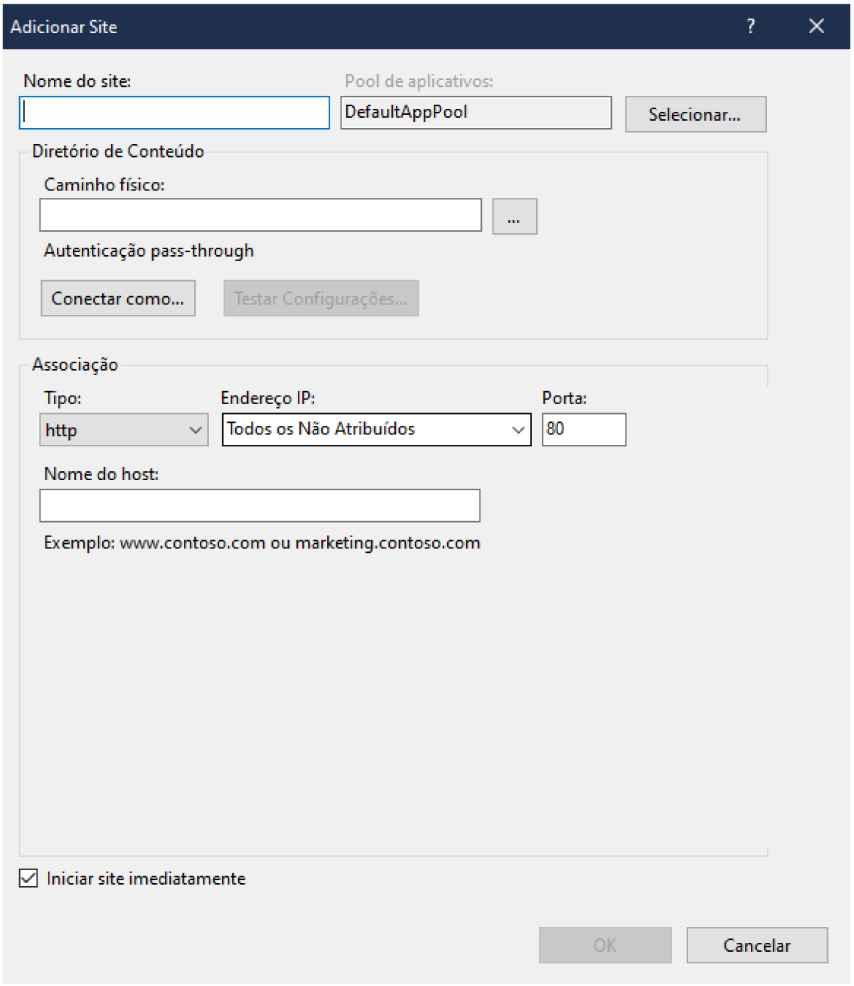


Figura 16 – Segundo passo para criar um site.

## 6.1 *Deployment*

Em termos simples, *deployment* (ou implantação) é o processo de levar o software do ambiente de desenvolvimento para um ambiente onde os usuários reais podem usá-lo. Como já dito anteriormente, quando um site rodando sob o IIS com o módulo IISNode, enquanto ativo, sempre que o código da aplicação for alterado o site será atualizado automaticamente, a partir de instanciações do node.exe. A partir disso, o processo para aplicação de alterações nesse projeto se dá apenas modificando o código necessário na instância produtiva, ou seja, alterando o que for preciso no diretório mapeado na criação do site no IIS, sem demais passos.

### 6.1.1 CI/CD

No mundo do desenvolvimento de software, a eficiência e a qualidade são fundamentais. Imagine se houvesse uma abordagem que unisse um fluxo de trabalho organizado e a automação do processo de desenvolvimento, garantindo que o código seja verificado

e implantado com confiabilidade. Bem, é aqui que entra o *GitFlow* e a Integração Contínua com GitHub *Actions*, uma combinação poderosa para otimizar o desenvolvimento de aplicações.

Para este trabalho foi desenvolvido um fluxo de integração contínua e entrega contínua baseado no Github *Actions* pela comodidade, uma vez que o projeto está disponibilizado e versionado no próprio GitHub, além da facilidade e oportunidades de personalização (Figura 17).

A integração contínua envolve a automação dos testes e da construção do software sempre que novos códigos são adicionados ao repositório. Isso ajuda a identificar problemas cedo, antes que eles se tornem mais difíceis e caros de corrigir. A entrega contínua automatiza o processo de empacotamento e implantação do software. Isso significa que, uma vez que o código passe pelos testes de CI, ele está pronto para ser implantado em um ambiente de produção a qualquer momento, com o mínimo de esforço humano.

Ainda dentro do CI/CD, foi sugerido um modelo de ramificação para controle de versão que define um fluxo de trabalho bem definido para os desenvolvedores no projeto. Ele organiza as ramificações em categorias claras, no caso do projeto, *"develop"* , *"homolog"* e *"main"*, permitindo uma colaboração mais harmoniosa. Além disso, temos o fluxo de integração, foram criados três serviços: *'lint'*, *'test'* e *'deploy'*. O primeiro verifica a formatação e a sintaxe do código de acordo com as particularidades da linguagem escolhida, o segundo realiza a execução automatizada dos testes criados junto com o criação de cada serviço feito, no fluxo de desenvolvimento gerando uma cobertura de código (*coverage*) e por fim é realizado o processo de *deploy* que basicamente leva as alterações realizadas para o ambiente estipulado, assim garantindo qualidade e estabilidade no código. Esses passos são aplicados nas três ramificações principais.

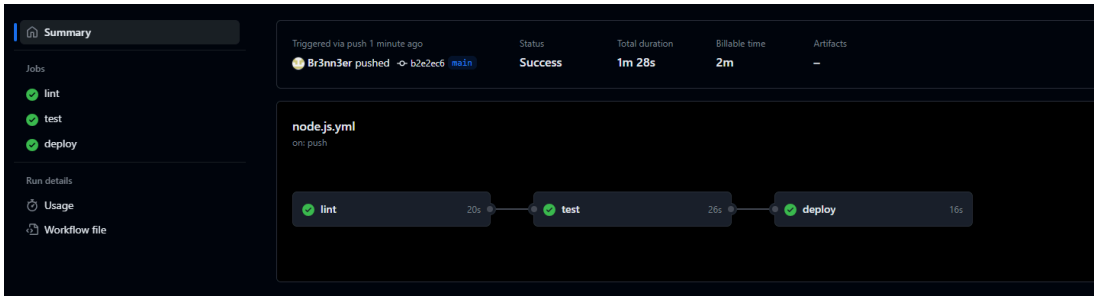


Figura 17 – CI/CD no GitHub.

Também foi convencionado e posteriormente documentado na *Wiki* do projeto regras de nomenclaturas e boas práticas para *branch* quanto para *commits* e a partir disso, também foi acrescentado no *Gitflow* os processo acordados nesse documento, a fim de gerar um histórico compreensível de todas as implementações dentro da aplicação, ou

seja, funciona como uma linha do tempo das modificações, de fácil busca e compreensão a todos os envolvidos no projeto.

O modelo descrito aqui não se propõe a ser uma bala-de-prata nem à prova-de-balas. A ideia principal desse modelo é favorecer a sinergia ao participantes, atuais ou futuros, desse projeto, aderente aos princípios de agilidade da organização, visando melhorar a eficiência de entrega, diminuir o operacional e fomentar a comunicação. Sendo assim, esse modelo pode sim ser replicado ou estendido por outras pessoas que se sentirem confortáveis.

Como dito anteriormente, para aplicar uma alteração em produção basta atualizar os arquivos mapeados para o site no IIS, e para que isso ocorra de forma automática, foi usada uma funcionalidade do GitHub chamada de *Self-hosted Runners*. Em essência, *self-hosted runners* são máquinas virtuais ou servidores físicos que você configura e controla para executar tarefas em seus repositórios do GitHub. Imagine esses *runners* como assistentes virtuais que executam tarefas específicas para você, seguindo suas instruções.

Com os *self-hosted runners*, em vez de depender exclusivamente dos *runners* fornecidos pelo GitHub, que são compartilhados entre vários usuários, tem-se o controle total sobre a capacidade e os recursos dos seus *runners*. Desse modo, com os *runners* instalados no mesmo servidor do IIS, podemos programá-los para realizar a atualização dos arquivos do site via serviço e/ou tarefa, integrado ao fluxo do CI/CD, de maneira automatizada, como podemos ver em (INC, 2023d).

# 7 Conclusão

O objetivo deste trabalho foi implantar uma ferramenta de gerenciamento de distribuição de disciplinas construída principalmente na linguagem *TypeScript* utilizando o IIS. A ferramenta foi desenvolvida utilizando os padrões de arquitetura, o que permitiu uma estrutura clara do projeto.

O GitHub foi utilizado para gerenciar o código-fonte da ferramenta. O fluxo de trabalho de desenvolvimento foi definido de acordo com as melhores práticas de desenvolvimento ágil. O CI/CD foi implementado para automatizar o processo de compilação, teste e implantação da ferramenta.

A ferramenta foi testada com sucesso em diferentes ambientes. Os resultados dos testes mostraram que a ferramenta atende aos requisitos definidos no escopo do trabalho. A ferramenta foi testada com sucesso e atende aos requisitos especificados no projeto. Ela é uma solução robusta e escalável, que pode ser utilizada por instituições de ensino superior para gerenciar a distribuição de disciplinas entre os docentes.

O uso de *TypeScript* no IIS foi uma escolha acertada, pois permitiu o desenvolvimento de uma ferramenta segura e eficiente. *TypeScript* é uma linguagem moderna e robusta, que oferece suporte a recursos avançados, como tipagem forte e inferência de tipos. O uso dos padrões de arquitetura também foi uma escolha acertada, pois permitiu o desenvolvimento de uma ferramenta fácil de manter e evoluir.

A ferramenta implementada neste trabalho pode ser utilizada como um exemplo de como implementar uma ferramenta de gerenciamento de disciplinas utilizando *TypeScript* no IIS.

Como recomendações para trabalhos, o projeto pode ser expandido em várias direções. Por exemplo, a ferramenta pode ser adaptada para atender às necessidades de outras instituições de ensino. Também pode ser desenvolvida uma versão mobile da ferramenta. Podemos também propor a implementação de funcionalidades adicionais, tais quais já foram mapeadas e incluídas no Kanban. Temos também uma melhoria na hospedagem, passando todos os ambientes não exclusivos ao desenvolvimento, para uma solução em nuvem. Além disso, podemos colocar a implementação de testes automatizados mais variados e abrangentes que garantiriam a qualidade da aplicação e facilitaria a identificação de erros.

# Bibliografia

AUTH0, o. **JSON web tokens introduction**. 2022. Disponível em: <https://jwt.io/introduction>. Citado na página 28.

BARBOSA, T. **Movendo a lógica de sua aplicação para services e repositories**. DEV Community, 2021. Disponível em: <https://dev.to/tadeubdev/movendo-a-logica-de-sua-aplicacao-para-services-e-repositories-4lee>. Citado na página 12.

CHACON, S.; STRAUB, B. **Pro Git: Everything you need to know about git**. Second. [S.l.]: Apress, 2023. Citado na página 14.

COMMUNITY, M. **Documentação do IIS**. 2023. Disponível em: <https://learn.microsoft.com/pt-br/iis/>. Citado na página 30.

DAMASCENO, M. I. R. Manutenção do sistema online para distribuição de disciplina. Universidade Federal de Uberlândia, 2021. Citado na página 19.

FERNANDES, G. S. Automação de testes com selenium webdriver aplicada no sistema de distribuição de disciplinas. Universidade Federal de Uberlândia, 2020. Citado na página 18.

FOUNDATION, O. **Introduction to node.js**. 2023. Disponível em: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>. Citado na página 14.

GOOGLE, A. **Superheroic JavaScript MVW framework**. 2022. Disponível em: <https://angularjs.org/>. Citado na página 9.

GROUP, T. P. G. D. **About**. 2023. Disponível em: <https://www.postgresql.org/about/>. Citado na página 15.

HANSELMAN, S. **Installing and running node.js applications within IIS on windows - are you mad?** 2011. Disponível em: <https://www.hanselman.com/blog/installing-and-running-nodejs-applications-within-iis-on-windows-are-you-mad>. Citado na página 51.

HEROKU, S. **Categories**. 2022. Disponível em: <https://devcenter.heroku.com/categories/deployment>. Citado na página 26.

INC, D. **Docker Overview**. 2023. Disponível em: <https://docs.docker.com/get-started/overview/>. Citado na página 13.

INC, G. **GitHub Student Developer Pack**. 2023. Disponível em: <https://education.github.com/pack>. Citado na página 26.

_____. **Olá, Mundo**. 2023. Disponível em: <https://docs.github.com/pt/get-started/quickstart/hello-world>. Citado na página 16.

_____. **Sobre executores auto-hospedados**. 2023. Disponível em: <https://docs.github.com/pt/actions/hosting-your-own-runners/managing-self-hosted-runners/about-self-hosted-runners>. Citado na página 58.

LOCATELLI, J. A. Sistema online para distribuição de disciplina. Universidade Federal de Uberlândia, 2015. Citado na página 18.

MENDES, M. d. S. Proposta de nova implementação do sistema online de distribuição de disciplinas. Universidade Federal de Uberlândia, 2022. Citado na página 19.

MONTE, I. G. Uma nova proposta de frontend para o sistema online de distribuição de disciplinas da facom. Universidade Federal de Uberlândia, 2022. Citado na página 19.

MORRIS, W. **What is Microsoft IIS web server software?** 2022. Disponível em: <https://www.elegantthemes.com/blog/wordpress/microsoft-iis>. Citado na página 29.

MOZDEVNET, M. c. **Guia JavaScript - JavaScript: MDN**. 2022. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>. Citado na página 15.

NAVES, P. E. d. P. Novas funcionalidades para o sistema online para distribuição de disciplina entre docentes. Universidade Federal de Uberlândia, 2016. Citado na página 18.

OLIVEIRA, A. S. d. Novas funcionalidades e manutenção no sistema online de distribuição de disciplinas. Universidade Federal de Uberlândia, 2017. Citado na página 18.

OLIVEIRA, I. A. G. d. Implementação de um sistema para distribuição de disciplinas. Universidade Federal de Uberlândia, 2018. Citado na página 18.

ORM, T. **Amazing orm for typescript and JavaScript (ES7, ES6, es5). supports mysql, PostgreSQL, mariadb, sqlite, MS SQL Server, Oracle, WebSQL databases. works in nodejs, browser, Ionic, Cordova and electron platforms.** 2023. Disponível em: <https://typeorm.io/>. Citado na página 16.

OTTO, M. **About**. 2023. Disponível em: <https://getbootstrap.com/docs/3.4/about/>. Citado na página 9.

PROJECT, A. T. **Apache Tomcat**. 2023. Disponível em: <https://tomcat.apache.org/>. Citado na página 9.

RODRIGUES, M. **Você Entende Repository Pattern? Você Está Certo Disso?** LaravelTips, 2018. Disponível em: <https://medium.com/laraveltips/voc%C3%AA-entende-repository-pattern-voc%C3%AA-est%C3%A1-certo-disso-d739ecaf544e>. Citado na página 11.

SALESFORCE, H. **Heroku dynos**. 2021. Disponível em: <https://www.heroku.com/dynos>. Citado na página 15.

SANTOS, A. V. d. Sistema online de distribuição de disciplinas: Manutenção e implementação de novos requisitos. Universidade Federal de Uberlândia, 2018. Citado 2 vezes nas páginas 11 e 18.

SILVA, S. S. d. Implementação de novos requisitos para o sistema online para distribuição de disciplinas. Universidade Federal de Uberlândia, 2016. Citado na página 18.

TYPESCRIPT, M. **Handbook - the typescript handbook**. 2023. Disponível em: <https://www.typescriptlang.org/docs/handbook/intro.html>. Citado na página 15.

URANO, L. **O Que São Ambientes?** 2023. Disponível em: <https://www.alura.com.br/artigos/o-que-sao-ambientes>. Citado 3 vezes nas páginas 20, 21 e 25.