UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rafael Faria Macedo Gonçalves

Desenvolvimento e Integração de API REST para a Comunicação entre o Pannotator e o Medpipe

Uberlândia, Brasil

2023

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rafael Faria Macedo Gonçalves

Desenvolvimento e Integração de API REST para a Comunicação entre o Pannotator e o Medpipe

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 Sistemas de Informação.

Orientador: Dr. Anderson Rodrigues dos Santos

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2023

Rafael Faria Macedo Gonçalves

Desenvolvimento e Integração de API REST para a Comunicação entre o Pannotator e o Medpipe

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 Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 09 de dezembro de 2023:

Dr. Anderson Rodrigues dos Santos
Orientador

Dr. Alexsandro Santos Soares
Professor

Luiz Fernando Afra Brito
Professor

Uberlândia, Brasil
2023

Dedico este trabalho à memória do meu amado pai, à minha mãe que me deram estrutura para seguir meus estudos e à minha esposa, cujo apoio inabalável e amor incondicional foram fundamentais para que eu pudesse concluir esta importante etapa da minha jornada acadêmica.

Agradecimentos

Agradeço a todas as pessoas que contribuíram para a conclusão deste trabalho. Primeiramente, agradeço ao meu orientador pelo seu apoio, orientação e valiosas contribuições ao longo deste processo. À minha família e amigos, manifesto meu agradecimento pelo constante incentivo e compreensão. Também gostaria de estender meus agradecimentos aos membros da banca examinadora, professores, colegas e todas as fontes de conhecimento que enriqueceram este trabalho com suas contribuições.

"A maneira de começar é parar de falar e começar a fazer."
Walt Disney

Resumo

O presente Trabalho de Conclusão de Curso (TCC) representa um esforço para integrar duas ferramentas na área de bioinformática. A integração bem sucedida do sistema Medpipe ao Pannotator, viabilizada pelo microsserviço ms-medpipe, oferece uma abordagem para a análise genômica. O microsserviço ms-medpipe, desenvolvido em Kotlin com o framework Spring Boot, desempenha um papel central na automação do processamento Medpipe. A utilização de *endpoints* REST, como a execução do Medpipe de forma assíncrona, busca de status e das predições geradas, contribui para a modularidade e escalabilidade do sistema. A disponibilização da documentação dos *endpoints*, exemplos de solicitações e logs detalhados facilitam a compreensão e utilização do ms-medpipe. Os resultados obtidos destacam a importância das informações fornecidas pelo Medpipe, enriquecendo a anotação genômica com mais detalhes, como a densidade de epítopos maduros (MED) e a classificação de proteínas de acordo com suas localizações subcelulares. Com os resultados obtidos com a integração utilizando o ms-mdepipe o Pannotator ficou equivalente a programas que tentam prover mais do que apenas anotação de função acrescentando dados sobre anotação sobre potencial imunológico, estrutura e localização celular.

Palavras-chave: Bioinformática, Genômica, Medpipe, Pannotator, Microsserviços.

Lista de ilustrações

Lista de códigos

Lista de abreviaturas e siglas

BLAST	Basic Local Alignment Search Tool
API	Application Programming Interface
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
TCC	Trabalho de Conclusão de Curso
NGS	Next-Generation Sequencing
RASTtk	RAST tool kit
PSAT	Protein Sequence Annotation Tool
DDBJ	DNA Data Bank of Japan
PVCs	Vacina baseados em proteínas
VIOLIN	Integrative Vaccine Investigation and Online Information Network
URL	Uniform Resource Locator
DNA	Deoxyribonucleic Acid

Sumário

1 Introdução

A bioinformática é uma área de pesquisa que integra a ciência da computação, a estatística, a matemática e a biologia. Ela tem sido fundamental para o avanço da pesquisa genômica, fornecendo ferramentas e metodologias para o processamento e análise de dados biológicos (GONÇALVES, 2023). Nesse contexto, o alinhamento de sequências emergiu como um pilar fundamental, proporcionando uma visão profunda das relações evolutivas e funcionais entre os genomas.

> Alinhamentos de sequências biológicas são ferramentas que, além de serem usadas para análise de regiões conservadas e de regiões que sofreram mutações em sequências homólogas, também servem como ponto de partida para outras aplicações em Biologia Computacional, como o estudo de estruturas secundárias de proteínas e a construção de árvores filogenéticas (BRITO; SOARES, 2003).

O alinhamento de sequências, ao revelar padrões de conservação e variação, não apenas desvenda a intrincada tapeçaria da evolução genômica, mas também serve como ponto de partida crucial para investigações mais aprofundadas. Uma aplicação vital desse processo reside na anotação genômica, onde a identificação de elementos funcionais, como genes e regiões regulatórias, constitui a essência da decodificação do código genético.

Segundo (RODRIGUES, 2023) "o processo de anotação é essencial para o desenvolvimento de metodologias cuja base parte da análise do material genético, tais como pan-genômica e taxogenômica". A anotação genômica é um processo complexo que requer a identificação e catalogação de elementos funcionais em um genoma e a grande quantidade de dados gerados por técnicas de sequenciamento torna essa tarefa ainda mais complicada. Nesse contexto, o Pannotator é uma ferramenta projetada para gerar uma anotação automática a partir de um genoma que foi manualmente curado. Essa ferramenta foi desenvolvida com o intuito de minimizar a carga de trabalho exigida na elaboração de relatórios e na correção de diversas anotações durante a execução de um projeto de pan-genoma (SANTOS et al., 2013a).

Contudo, a busca por aprimoramentos e predições adicionais conduz à introdução do *Mature Epitope Density* (Medpipe). O Medpipe é um pipeline de bioinformática projetado para prever a densidade de epítopos por porções maduras de proteínas (SANTOS et al., 2013b).

Tanto Medpipe quanto Pannotator são ferramentas independentes. No Pannotator proteínas somente são utilizadas para a realização do *Basic Local Alignment Search Tool* (BLAST). BLAST é um algoritmo que pode alinhar e comparar rapidamente uma

sequência de DNA, do inglês *deoxyribonucleic acid*, permitindo consulta a banco de dados de sequências (LOBO, 2008).

Os dados sobre o potencial e localização subcelular (Citoplasma, Parede Celular, Exposta em Superfície e Secretada) de proteínas gerados pelo Medpipe podem ser úteis a um usuário do Pannotator que esteja anotando o genoma de uma bactéria. Seria interessante se os usuários do Pannotator pudessem contar com uma integração com o Medpipe para solicitar o seu serviço de anotação e incorporá-lo na anotação de um genoma pelo Pannotator. Tal integração reduziria o trabalho de um pesquisador de executar o Pannotator e Medpipe em separado e depois ainda ter que anexar o resultado do Medpipe ao Pannotator de forma manual, via importação de tabelas. Além disso, o serviço oferecido pelo Medpipe é único, não existindo outra ferramenta que ofereça o mesmo serviço na internet. Considerando que o Medpipe é constituído de software não livre, não é uma alternativa liberar o código fonte. É desejável que exista um serviço de comunicação entre o Medpipe e qualquer outro software de análises de sequências biológicas. Acreditamos que a implementação de um microsserviço na internet que execute e retorne os resultados do Medpipe atenderá às expectativas de prover dados adicionais para o Pannotator e qualquer outro software.

> O termo "Arquitetura de Microsserviços" surgiu nos últimos anos para descrever uma maneira particular de projetar aplicativos de software como conjuntos de serviços implementados de forma independente. Embora não exista uma definição precisa desse estilo de arquitetura, existem certas características comuns em torno da organização em torno da capacidade de negócios, implantação automatizada, inteligência nos terminais e controle descentralizado de linguagens e dados. (FOWLER, 2014)

Uma das características do uso da Arquitetura de Microsserviços é poder projetar um conjunto de serviços independentes, cada um responsável por uma parte específica de uma funcionalidade, construídos em torno de recursos de negócios, o que significa que eles são projetados para atender às necessidades de um determinado negócio. Esses serviços implantados de forma independente podem ser atualizados ou desativados sem afetar os outros serviços. A gestão é descentralizada, o que significa que cada um é responsável por sua própria gestão, além de poder ser escritos em diferentes linguagens de programação e utilizar diferentes tecnologias de armazenamento de dados, o que dá às organizações mais flexibilidade (FOWLER, 2014).

Conforme (HOSSAIN et al., 2023) os microsserviços oferecem benefícios como melhor escalabilidade, implantação mais rápida, flexibilidade, resiliência, equipes de desenvolvimento menores, modularidade, confiabilidade e reutilização. Mas há desvantagens e apresentam desafios em seu uso, como especificado por Velepucha e Flores (2023), por exemplo a alta curva de aprendizado dessa nova arquitetura, falta de experiência da equipe de desenvolvimento, sobrecarga de rede, aumento da complexidade, duplicação de dados.

Um microsserviço escalável e de alto desempenho não se resume apenas à capacidade de lidar com um grande volume de tarefas ou solicitações simultâneas. Sua essência reside na eficiência com que executa tais operações, sendo capaz de processar tarefas de forma rápida e eficiente, utilizando recursos de forma otimizada, manter um alto nível de desempenho mesmo sob carga elevada, sem comprometer a qualidade do serviço e estar preparado para o crescimento futuro da demanda, adaptando-se e escalando de forma transparente (FOWLER, 2017).

A flexibilidade de realizar modificações em um único serviço e implantá-lo de forma independente otimizam o desenvolvimento e a entrega de software, permitindo a rápida implementação de novas funcionalidades e correções de *bugs*. Essa autonomia também facilita o isolamento de falhas, caso ocorram, restringindo o problema ao serviço em questão e simplificando a sua resolução. Além disso, a arquitetura de microsserviços possibilita a reversão de alterações de maneira rápida e simples, minimizando os riscos e impactos negativos (NEWMAN, 2021).

Muitas empresas utilizam de microsserviços em suas soluções, um exemplo é a Netflix que desenvolveu uma plataforma de microsserviços centrada em fluxos de trabalho de mídia, visando aumentar a flexibilidade e velocidade de desenvolvimento (Netflix Technology Blog, 2023).

Outro exemplo é a Amazon, que diante de problemas de escalabilidade e produtividade devido a atualizações e projetos frequentes enfrentou a necessidade de refatorar seu sistema e adotou a abordagem de microsserviços. A divisão dos aplicativos monolíticos em pequenos serviços independentes permitiu à empresa responder melhor aos requisitos de expansão e realizar alterações rápidas e específicas (HILLPOT, 2023).

Com base na arquitetura de microsserviços, projetamos o microsserviço ms-medpipe que faz uso do pipeline do Medpipe. Essa abordagem permite a implantação independente e atualizações ágeis, garantindo rápida disponibilização de novas funcionalidades aos usuários. Apesar do trabalho mostrar a integração entre Pannotator e Medpipe através do ms-medpipe é essencial ressaltar que o microsserviço desenvolvido não é específico para uso do Pannotator, podendo ser utilizado por qualquer software ou pipeline de análises de sequências biológicas através da disponibilização dos *endpoints* na internet.

1.0.1 Objetivo

O objetivo principal deste trabalho foi criar um microsserviço com o nome de ms-medpipe, que faz a execução do Medpipe através de Interface de Programação de Aplicação, do inglês *Application Programming Interface* (API) que segue o estilo de arquitetura de Transferência de Estado Representacional, do inglês *Representational State Transfer* (REST), além de integrar essas APIs ao Pannotator, visando fornecer uma plataforma

acessível para pesquisadores e profissionais que trabalham com montagem de genomas e análise de sequências genômicas. Especificamente, os objetivos incluem:

- Descrever o desenvolvimento do microsserviço do ms-medpipe;
- Integrar as APIS desenvolvidas no microsserviço ao Pannotator, considerando as necessidades da ferramenta;

2 Revisão Bibliográfica

2.1 Arquitetura de Microsserviços

Já foi dito que o uso de microsserviços são construídos em torno de recursos de negócios e podem ser implantados de forma independente (FOWLER, 2014).

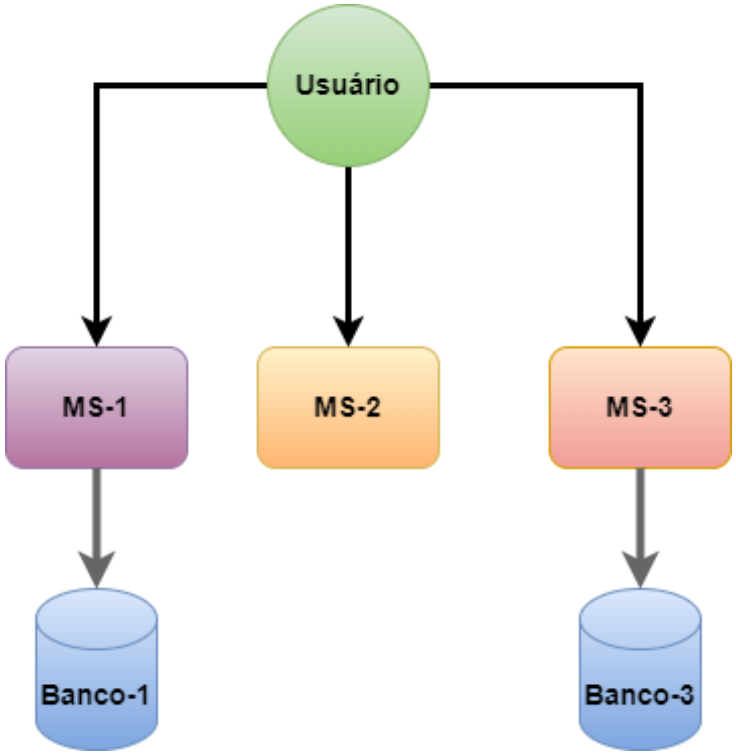

Figura 1 – Exemplo de Arquitetura de microsserviços.

Conforme Figura 1 cada microsserviço pode ser responsável por um negócio específico e funcionar de forma independente sem afetar os demais. Nesse exemplo MS-1 poderia ser um serviço de cadastro de colaboradores de uma empresa, MS-2 um serviço que realiza cálculos de impostos para a empresa e o MS-3 um serviço para cadastro de fornecedores da empresa. Todos os três microsserviços são independentes, isso facilita qualquer alteração em algum deles, caso a empresa precise incluir uma nova regra de cálculo de imposto somente o MS-2 seria afetado sem qualquer efeito colateral ao MS-1 e MS-3.

Isso foi apenas um exemplo de como microsserviços podem ser utilizados, e no caso do ms-medpipe que foi construído para este trabalho seguimos a mesma lógica. Caso seja necessário uma execução ou busca de resultados de outra pipeline de bioinformática que não seja o Medpipe deve-se construir outro microsserviço. O ms-medpipe foi construído apenas para execução e busca de resultados do Medpipe, disponibilizando

endpoints para internet, qualquer funcionalidade nova que não faça parte do Medpipe não deve ser incluído no ms-medpipe, a partir disso deve-se pensar na construção de um novo microsserviço.

2.2 Trabalhos Correlatos

Nesta seção são abordados trabalhos com objetivos semelhantes à proposta do Pannotator e Medpipe.

2.2.1 Revisão da literatura referente ao Pannotator

Nos últimos dez anos, várias ferramentas de anotação automática de genomas foram disponibilizadas como software de código aberto ou em páginas de acesso público na internet. Abordaremos, de maneira concisa nos parágrafos subsequentes, as principais características dessas ferramentas.

A capacidade multiplex e alto rendimento dos instrumentos de sequenciamento de DNA tornaram a sequência completa do genoma bacteriano uma rotina. O Prokka é uma ferramenta de linha de comando implementada em Perl que permite a anotação completa de um genoma bacteriano preliminar em cerca de 10 minutos em um computador *desktop*, produzindo arquivos de saída compatíveis com padrões para análises adicionais ou visualização em navegadores genômicos, superando sistemas baseados na web e e-mail que são inadequados para dados sensíveis ou integração em pipelines computacionais (SEEMANN, 2014).

BG7 é uma ferramenta de código aberto baseada em um paradigma de anotação de genes centrado em proteínas, projetada especificamente para genomas bacterianos sequenciados com tecnologias de sequenciamento de próxima geração (NGS), considerando as peculiaridades dos genomas bacterianos (ausência de íntrons e escassez de sequências não codificantes de proteínas) e das tecnologias NGS, sendo capaz de lidar com erros de sequência e anotar genomas altamente fragmentados ou sequências mistas de vários genomas (como as obtidas por amostras de metagenômica), além de ter sido projetada para escalabilidade, utilizando uma infraestrutura de computação em nuvem baseada no *Amazon Web Services* (AWS) (TOBES et al., 2015).

O *RAST tool kit* (RASTtk), é uma versão modular do mecanismo de anotação RAST que permite aos pesquisadores construir pipelines de anotação personalizados, escolher software para identificar e anotar características genômicas, adicionar recursos personalizados a um trabalho de anotação, acomodar a submissão em lote de genomas e personalizar protocolos de anotação para submissões em lote, marcando a primeira grande reestruturação de software do RAST desde sua criação em 2008 (BRETTIN et al., 2015).

O *Protein Sequence Annotation Tool* (PSAT), é uma meta-plataforma baseada na web para análises integradas e de alto rendimento de sequências genômicas, demonstrando sua utilidade ao anotar os produtos gênicos de peptídeos preditos de *Herbaspirillum sp. strain* RV1423, importar os resultados para o EC2KEGG e utilizar as comparações funcionais resultantes para identificar uma via catabólica putativa, destacando o potencial metabólico em um genoma com anotação limitada (LEUNG et al., 2016).

Genix é uma plataforma de anotação de genoma bacteriano baseada na web, que se destaca por fornecer resultados mais próximos à anotação de referência, com uma menor quantidade de proteínas falsas positivas e proteínas funcionais não anotadas, sendo capaz de aprimorar a precisão dos passos de anotação de genomas bacterianos e fornecer resultados de alta qualidade (KREMER et al., 2016).

A Sma3s é uma ferramenta computacional precisa para a anotação de proteínas de forma automática, com funcionalidades úteis para a ciência fundamental e aplicada, fornecendo categorias funcionais e requerendo baixos recursos computacionais, permitindo a anotação completa de proteomas e transcriptomas em cerca de 24 horas em um computador pessoal (CASIMIRO-SORIGUER; MUÑOZ-MÉRIDA; PÉREZ-PULIDO, 2017).

O proGenomes é um banco de dados abrangente que contém informações genômicas de alta qualidade de uma ampla variedade de microrganismos. Ele fornece acesso a genomas bacterianos, arqueais e eucarióticos, juntamente com metagenomas e plasmídeos. O proGenomes é uma valiosa ferramenta para estudos de genômica comparativa, evolução microbiana e pesquisa de novas espécies (MENDE et al., 2017).

O DFAST, um pipeline de anotação de genomas para procariotos, que também auxilia no envio de dados para o banco de dados de sequências públicas, com destaque para sua capacidade de anotar um genoma bacteriano de tamanho típico em até 5 minutos e sua integração com o *DNA Data Bank of Japan* (DDBJ) (TANIZAWA et al., 2019).

EuGene é uma ferramenta de busca de genes que pode ser usada em genomas procariontes e eucariontes. Ele utiliza informações estatísticas, similaridades com genes e proteínas conhecidos e dados estruturados em formato GFF3 para prever as unidades de transcrição principais no genoma e realizar anotações funcionais. Essa ferramenta é capaz de lidar com genomas complexos, com regiões repetidas e elementos transponíveis, e pode ser configurada como ab initio, baseada em similaridade ou híbrida, dependendo das fontes de informação utilizadas (SALLET; GOUZY; SCHIEX, 2019).

O DescribePROT é um banco de dados que contém 13 descritores preditos em nível de aminoácidos para estrutura e função de proteínas, abrangendo 83 proteomas completos de organismos modelo e incluindo 7,8 bilhões de predições para quase 600 milhões de aminoácidos em 1,4 milhão de proteínas, com a possibilidade de busca por sequência de aminoácidos e *UniProt accession number* (ZHAO et al., 2021).

O µProteInS é um pipeline de proteogenômica implementado em Python 3.8, que combina genômica, transcriptômica e proteômica para identificar microproteínas em bactérias, superando as limitações das abordagens tradicionais e permitindo a identificação de ORFs pequenos (smORFs) com sobreposição de genes, transcritos sem líder e sequências não conservadas. O µProteInS é distribuído como um software de código aberto (SOUZA et al., 2022).

2.2.2 Revisão da literatura referente ao Medpipe

Esta é uma revisão da literatura sobre software e servidores web que se dispõem a encontrar proteínas candidatas a alvos vacinais ou diagnósticos contra bactérias patogênicas, nos últimos dez anos. Surpreendentemente, encontrou-se poucos trabalhos sobre o tema.

O Jenner-Predict é um servidor web que utiliza uma abordagem baseada em conhecimento de patogênese bacteriana para prever candidatos a vacinas baseadas em proteínas (PVCs) a partir de proteomas de patógenos bacterianos. O servidor considera domínios funcionais de diferentes classes de proteínas envolvidas em interações hospedeiro-patógeno e patogênese, incluindo adesinas, virulência, invasinas, porinas, flagelina, toxinas e outros. Além disso, o Jenner-Predict avalia a imunogenicidade potencial dos PVCs, comparando-os com epítopos conhecidos e considerando a ausência de autoimunidade e conservação em diferentes cepas. O servidor demonstrou alta precisão na previsão de PVCs conhecidos e superou métodos existentes como NERVE, Vaxign e VaxiJen (JAISWAL et al., 2013).

Outro site, o VacTarBac é um servidor web que utiliza uma abordagem de imunoinformática para identificar candidatos a vacina baseados em epítopos contra 14 espécies bacterianas patogênicas. O servidor utiliza uma análise abrangente de proteínas-alvo, incluindo fatores de virulência e genes essenciais, para prever epítopos com potencial para estimular diferentes componentes do sistema imunológico. Além disso, o VacTarBac removeu epítopos auto-reconhecidos para evitar respostas imunes indesejadas. A análise revelou 21 proteínas de 5 espécies bacterianas como alvos de vacinas promissoras. O servidor também identifica epítopos de células B, células T e ligantes MHC-II, além de adjuvantes, resultando em um total de 252 epítopos únicos. O VacTarBac apresenta ferramentas de visualização para auxiliar os usuários na identificação dos melhores candidatos à vacina em uma sequência antigênica (NAGPAL; USMANI; RAGHAVA, 2018).

O *Integrative Vaccine Investigation and Online Information Network* (VIOLIN) é um banco de dados e sistema de análise de pesquisa de vacinas que realiza curadoria, armazenagem, análise e integração de diversos dados de pesquisa relacionados a vacinas. Desde sua primeira publicação em 2008, o VIOLIN passou por atualizações significativas. Atualmente, ele inclui mais de 3240 vacinas para 192 doenças infecciosas e oito doenças não infecciosas. Dentro do VIOLIN, existem mais de 10 programas relativamente indepen-

dentes, como o Protegen, que armazena proteínas antigênicas comprovadamente válidas para o desenvolvimento de vacinas, e o VirmugenDB, que anota genes de fatores de virulência que podem ser mutados para gerar vacinas atenuadas com sucesso. O VIOLIN também inclui o Vaxign, o primeiro programa de previsão de candidatos a vacinas baseado em vacinologia reversa, e outros componentes de vacinas, como adjuvantes (Vaxjo) e plasmídeos de vacina de DNA (DNAVaxDB). Além disso, o VIOLIN possui bancos de dados de vacinas humanas licenciadas (Huvax) e vacinas veterinárias (Vevax). A Ontologia de Vacinas é aplicada para padronizar e integrar os diferentes dados no VIOLIN. O VIOLIN também hospeda a Ontologia de Eventos Adversos de Vacinas (OVAE), que representa logicamente os eventos adversos associados às vacinas humanas licenciadas (HE et al., 2014).

Uma alternativa não baseada em web é o TiD, um aplicativo autônomo que identifica alvos potenciais para o desenvolvimento de medicamentos. Ele utiliza a premissa de que uma proteína deve ser essencial para a sobrevivência do patógeno e não homóloga ao hospedeiro para se qualificar como alvo. O TiD remove proteínas parálogas, seleciona as essenciais e exclui aquelas homólogas a organismos hospedeiros. Os alvos são classificados como conhecidos, novos ou virulentos. Os usuários podem realizar análises de vias metabólicas, interações proteicas e outras funcionalidades por meio de servidores web integrados. Alvos identificados pelo TiD para Listeria monocytogenes, Bacillus anthracis e Pseudomonas aeruginosa mostraram sobreposição com estudos anteriores. O TiD é uma ferramenta útil para o desenvolvimento racional de medicamentos, sendo capaz de analisar alvos em um proteoma bacteriano em cerca de duas horas (GUPTA et al., 2017).

2.2.3 Integração de anotação e patogenicidade

Além de prover a anotação funcional, algumas ferramentas também se dispõe a prover dados adicionais de anotação relativos ao tipo de exportação proteica, resistência a antibióticos, estrutura e capacidade imunológica de proteínas. Por exemplo, o MacSyFinder é uma ferramenta de bioinformática que permite a busca de sistemas genéticos específicos em genomas completos. Ele utiliza uma abordagem baseada em padrões e perfis de domínio para identificar e anotar sistemas de secreção, sistemas de resistência a antibióticos e outros sistemas genéticos em diferentes organismos. O MacSyFinder é distribuído como um software de código aberto (ABBY; ROCHA, 2017). Já no capítulo "*Antigen Discovery in Bacterial Panproteomes*" é descrita uma metodologia in silico que integra abordagens pangenômicas, imunoinformáticas, estruturais e evolutivas para a triagem de potenciais antígenos em uma determinada espécie bacteriana, visando o desenvolvimento de vacinas amplamente protetoras e evitando a imunidade específica a alelos, além de permitir o desenvolvimento de ensaios diagnósticos (YERO; CONCHILLO-SOLé; DAURA, 2021).

O intuito de nosso trabalho segue estes exemplos quando integra a ferramenta Pannotator com o Medpipe para prover, junto à anotação funcional, dados que ajudem a encontrar proteínas de um genoma com potencial para produção de vacinas e testes de diagnósticos contra patógenos. Entretanto, nossa proposta foi adiante quando implementou um microsserviço de anotação de patogenicidade proteica que pode ser usado por qualquer outro software de anotação genômica. Nosso software, em vez de um concorrente com os softwares de anotação funcional, se propõe a ser um parceiro das propostas mais avançadas de anotação funcional atualmente existentes quando oferece um microsserviço que pode ser consumido de forma democrática.

3 Desenvolvimento

Neste capítulo será detalhado como o microsserviço ms-medpipe foi concebido, projetado e construído. Será abordada as tecnologias utilizadas e a implementação dos *endpoints* cruciais.

3.1 Linguagem de programação

A linguagem de programação escolhida para o desenvolvimento do microsserviço foi Kotlin. Criada pela JetBrains, Kotlin é uma linguagem de programação moderna e concisa que é segura, expressiva e interoperável com Java (SAUDATE, 2021). Kotlin minimiza a necessidade de código *boilerplate* em comparação com Java. Isso resulta em código mais claro e de fácil manutenção.

3.2 Tecnologias e *Frameworks*

Nesta subseção serão descritas as principais tecnologias e *frameworks* utilizados no projeto do ms-medpipe.

3.2.1 *Spring Boot*

Spring Boot é um *framework* de desenvolvimento de aplicações Java que é baseado no princípio de "apenas o necessário". Ele fornece uma série de recursos e ferramentas necessárias para criar aplicações Java, mas não fornece recursos que não são essenciais. Isso torna *Spring Boot* uma plataforma leve e fácil de usar (Spring, 2023). Uma das características marcantes do *Spring Boot* é a sua abordagem de "opinião sobre configuração". Isso significa que ele fornece configurações padrão sensatas para muitos aspectos de uma aplicação, permitindo que os desenvolvedores se concentrem mais na lógica de negócios do que nas configurações complexas. Além disso, o *Spring Boot* oferece um sistema integrado de construção e gerenciamento de dependências, o que simplifica a gestão das bibliotecas necessárias para o projeto.

3.2.2 Maven

Maven é uma ferramenta de automação de compilação e gerenciamento de projetos baseada em um modelo de gerenciamento de projetos chamado de modelo de artefato. O modelo de artefato é uma estrutura que define os artefatos que um projeto pode ter, como código fonte, bibliotecas, arquivos de configuração e outros arquivos. Maven usa o

modelo de artefato para automatizar os processos de compilação, teste, empacotamento e implantação de projetos (Apache Software Foundation, 2023).

3.2.3 Banco de Dados H2

O banco de dados H2 foi escolhido para armazenar informações relacionadas aos processos de integração. H2 é um banco de dados relacional escrito em Java. Ele pode ser executado em modo cliente-servidor ou em modo embutido. No modo cliente-servidor, o banco de dados é executado em um servidor separado da aplicação Java. No modo embutido, o banco de dados é executado no mesmo processo da aplicação Java (H2 Database Engine, 2023).

3.3 Visão Geral do Microsserviço

O código fonte do microsserviço se encontra no GitHub[1] e o projeto recebeu o nome de ms-medpipe fazendo referência a ferramenta Medpipe que será executada pelo microsserviço. O ms-medpipe possui três *endpoints* essenciais para a utilização do Medpipe.

O *endpoint* de execução do *script* Medpipe é utilizado para preparação do diretório temporário de arquivos gerados pelo Medpipe e execução de forma assíncrona do *script*. O mesmo faz uso do H2 para registrar o status do processo e o diretório gerado para posterior consulta. O *endpoint* de busca do status do processo é utilizado para informação sobre o status do processamento do Medpipe. Essa funcionalidade será utilizada para que se possa saber se o processamento do *script* do Medpipe foi finalizado e se houve erro na execução. O *endpoint* de busca das predições do Medpipe é utilizado para extração dos resultados gerados pelo Medpipe e retornar para o usuário.

3.4 *Endpoints* HTTP

Nesta seção serão descritos os *endpoints* REST criados e suas funcionalidades.

3.4.1 Execução do Medpipe

O *endpoint* de execução do Medpipe é acionado por uma solicitação HTTP utilizando o verbo *POST* na rota /v1/medpipe/run e é um componente essencial do ms-medpipe, responsável por processar solicitações para a execução do *script* Medpipe. Desempenha um papel central na automação do processamento Medpipe, permitindo que os usuários enviem arquivos e parâmetros relevantes para a execução do *script* Medpipe.

[1] https://github.com/rafaelfmg/ms-medpipe

Tipo de requisição: *POST*

Rota: /v1/medpipe/run

Parâmetros de Entrada:

- ***file***: Este parâmetro representa o arquivo que o cliente deseja processar com o *script* Medpipe. Ele é do tipo *MultipartFile*, que é comumente usado para lidar com o envio de arquivos em aplicativos da web.
- ***cellWall***: Medida de espessura da parede celular em aminoácidos.
- ***organismGroup***: Grupo de organismos, atualmente os valores aceitos são 0 ou 1.
- ***epitopeLength***: Comprimento de epítopo.
- ***e-mail***: Email para envio dos resultados gerados pelo Medpipe.
- ***membraneCitoplasm***: Parâmetro que deve receber o valor 1 se considera membrana e citoplasma ou zero se não considera.

Etapas da rota

- **Criação de Diretório**: Primeiro, utiliza a função *buildDirectory* para construir o caminho completo para um diretório temporário onde os resultados do processamento serão armazenados. Esse diretório é gerado na pasta "temp" com o nome "MsMedpipe" concatenado com o timestamp evitando que ocorra duplicação de nomes.
- **Salvamento do Arquivo**: Em seguida, utiliza-se a função *saveFile* para salvar o arquivo recebido na solicitação HTTP no diretório especificado. Isso é fundamental, pois o *script* Medpipe requer acesso a esse arquivo para o processamento.
- **Controle de Processo**: Além disso, é criado um objeto *MedpipeControl* para rastrear o status do processo. Esse objeto é salvo no banco de dados e serve como uma ferramenta de monitoramento do progresso do processamento do Medpipe.
- **Execução do *script* Medpipe**: A função então constrói um comando *shell* que incorpora todos os parâmetros fornecidos, incluindo o caminho do arquivo, detalhes sobre a parede celular, grupo de organismos, etc. Esse comando é executado por meio da função *Runtime.getRuntime().exec()*, que executa o *script* Medpipe em um processo assíncrono.

- **Atualização de Status**: O status do processo é atualizado com base no resultado da execução do *script*. Se a execução for bem sucedida, o status é definido como concluído. Em caso de falha, o status indica um erro.

Vale destacar que a função *runFileProcess* é anotada com *@Async*, o que significa que ela é executada em uma *thread* separada. Isso é feito para garantir que a função não bloqueie a *thread* principal do serviço e permita que outras solicitações sejam processadas de forma eficiente, melhorando a escalabilidade do ms-medpipe. O Código 3.1 é responsável por executar o Medpipe.

Código 3.1 – Codigo que executa o Medpipe.

```
@Async
fun runScript(
    fileResult: String,
    cellWall: String,
    organismGroup: String,
    epitopeLength: String,
    email: String,
    membraneCitoplasm: String,
    medpipeControl: MedpipeControl
) {
    try {
      val command = "sh medpipe $fileResult $cellWall
                     $organismGroup $epitopeLength $email
                     $membraneCitoplasm"
      log.info("[runScript] - Start exec: "
               + LocalDateTime.now() + " commad: "
               + command)
      val process = Runtime.getRuntime().exec(command)
      val processEnd = process.waitFor()
      val result = BufferedReader(
                   InputStreamReader(process.inputStream))
                   .readText()
      log.info("[runScript] - Time: "
               + LocalDateTime.now()
               + " Result: " + result)
      log.info("[runScript] End process: $processEnd")
      unzipFileResult(medpipeControl.directory)
      updateStatus(Status.FINISHED, medpipeControl)
    } catch (e: Exception) {
```

```
            updateStatus(Status.ERROR, medpipeControl)
            throw RuntimeException(e.message)
        }
    }
```

3.4.2 Busca Status do Processamento

A busca de status é um *endpoint* que lida com solicitações HTTP *GET* na rota /v1/medpipe/{id}/status. Sua principal finalidade é consultar o status de um processo Medpipe com base no ID fornecido como parâmetro no Localizador Uniforme de Recursos (URL) da solicitação. Seu funcionamento é relativamente simples.

Tipo de requisição: *GET*

Rota: /v1/medpipe/{id}/status

Parâmetro de Entrada

- ***id***: Parâmetro referente ao processamento do Medpipe, com ele é possível buscar na base H2 as informações de status e diretório de geração do arquivo.

Etapas da rota

- **Recebimento do ID**: A função recebe o ID do processo como parte da URL da solicitação HTTP. Esse ID é extraído automaticamente pelo *Spring Framework* devido à anotação *@PathVariable*.
- **Consulta de Status**: A função invoca o método *findStatusProcess* do serviço *MedpipeService*, passando o ID como argumento. Essa chamada ao serviço é responsável por consultar o status do processo no banco de dados.
- **Retorno de Resultado**: O status do processo, que é retornado pelo serviço, é então retornado como resultado da função *getStatus*. O status é do tipo *Long*, permitindo que retorne um valor inteiro representando o status do processo.

Esse status pode assumir três valores já pré estabelecidos.

- **Status com valor zero**: Indica que o processo foi concluído com sucesso.
- **Status com valor 1**: Indica que o processo ainda está em execução e deve-se aguardar.

- **Status com valor -500**: Indica que aconteceu um erro no processamento.
- **Status com valor -404**: Indica que não foi encontrado na base de dados um processo referente ao ID recebido pela requisição.

O *endpoint* oferece aos usuários a capacidade de acessar informações em tempo real sobre o andamento dos processos Medpipe. Ela desempenha um papel fundamental na transparência e no monitoramento do processamento de dados do *script* Medpipe. O Código 3.2 é responsável por buscar o status de processamento.

Código 3.2 – Codigo que busca o status de processamento.

```
fun findStatusProcess(id: Long): Long? {
    log.info("[findStatusProcess] - ID: $id")
    val result = medpipeControlRepository.findById(id)
    return if (result.isPresent)
           result.get().status?.statusCode
           else Status.NOT_FOUND?.statusCode
}
```

3.4.3 Busca das predições do Medpipe

A busca das predições geradas pelo Medpipe é um *endpoint* que lida com solicitações HTTP utilizando o verbo *GET* na rota /v1/medpipe/{id}/predictions. Seu objetivo principal é permitir aos clientes a visualização das predições geradas pelo Medpipe.

Tipo de requisição: *GET*

Rota: /v1/medpipe/{id}/predictions

Parâmetro de Entrada

- ***id***: Parâmetro referente ao processamento do Medpipe, com ele é possível buscar na base H2 as informações de status e diretório de geração do arquivo.

Etapas da rota

- **Recebimento do ID**: A função recebe o ID do processo como parte da URL da solicitação HTTP. Esse ID é extraído automaticamente pelo *Spring Framework* devido à anotação *@PathVariable*. O ID recebido é usado na função *getPrediction* para obter as previsões de processamento do script Medpipe com base no ID fornecido.

- **Busca do processo**: Internamente a função *getPrediction* utiliza a função *findProcess* para obter um objeto *MedpipeControl* com base no id. Se o processo não for encontrado uma exceção é lançada indicando um status HTTP 404 (NOT FOUND).
- **Busca do Arquivo**: Após a busca do processamento na base de dados, tem se o objeto *MedpipeControl* que contém a informação do diretório onde foi gerado o arquivo resultante para extração das predições.
- **Montagem do resultado**: Com o caminho do arquivo obtido a função lê o arquivo de resultados linha por linha usando um *BufferedReader*. Cada linha é processada e retirada informações como id do gene, predição numérica e tipo da proteína e são adicionadas a uma instância de *StringBuilder*.

O *endpoint getPredictions* oferece aos clientes a capacidade de acessar e baixar os resultados do processamento Medpipe de forma eficiente. Ele desempenha um papel crucial na disponibilização dos resultados para análise subsequente ou armazenamento local, tornando o microsserviço ms-medpipe uma ferramenta versátil e útil para os usuários. O Código 3.3 é responsável por buscar as predições geradas pelo Medpipe.

Código 3.3 – Codigo que busca as predições geradas pelo Medpipe.

```
fun getPrediction(id: Long): StringBuilder {
    log.info("[getPrediction] - Start exec id: $id")
    val medpipeControl = findProcess(id) ?: throw
    RuntimeException(HttpStatus.NOT_FOUND.toString())
    val fileResult = medpipeControl.directory + "/" +
    ResultFile.TARGET_FASTA_RESULT_SORT.description

    log.info("[getPrediction] - File Result: $fileResult")
    var line = ""
    val result = StringBuilder()
    if (!FileUtil.fileExists(fileResult)){
        return result
    }
    BufferedReader(FileReader(fileResult)).use { br ->
      while (br.readLine()?.also { line = it } != null) {
        val values: Array<String> = line.split(" ")
                                    .toTypedArray()
        result.append(values[0] + " " +
                values[4].substring(4)
                + " " + values[6] + "\n")
      }
```

```
        }
        return result
    }
```

3.5 Instalação ms-medpipe

Nesta seção será detalhado como é o processo de geração do executável do ms-medpipe e instalação no servidor. O código fonte do microsserviço se encontra no GitHub.

Dependências

- **Máquina Virtual Java**: A linguagem Kotlin tem seu processo de compilação na Máquina Virtual Java (JVM), portanto é preciso que uma versão do Java esteja instalada no servidor, para esse projeto a versão necessária é a 11, além de configurar a variável de ambiente "JAVA_HOME".
- **Maven**: O Maven como ferramenta de compilação precisa estar configurado adequadamente para execução. Necessário realizar download da versão 3.8, o arquivo baixado vai estar compactado, realizar a extração para o diretório desejado. Configure a variável de ambiente "M2_HOME" com o diretório onde se encontra a instalação do maven, por exemplo: "M2_HOME=/home/apache-maven-3.8". Em seguida inclua a variável "M2_HOME" no Path do sistema apontando para a pasta bin do maven, por exemplo:"PATH=$PATH:$M2_HOME/bin". Feito isso o comando "mvn" já pode ser usado no terminal do servidor.

3.5.1 Processo de compilação

Para compilar o projeto é necessário usar o comando "mvn" do maven que já deve estar instalado no servidor. Usando a linha de comando do sistema vá até o diretório raiz do projeto ms-medpipe e digite "mvn clean install". Esse comando irá baixar as dependências do projeto, compilar suas classes, criar uma pasta target e dentro dela teremos um arquivo ".jar" gerado. O arquivo gerado segue a nomenclatura que foi configurada no pom.xml do projeto (Código 3.4), o resultado é a combinação dos campos *name* e *version* gerando o arquivo "ms-medpipe-1.0.0-Release.jar".

Código 3.4 – Configuração do arquivo pom.xml.

```
<groupId>com.tcc</groupId>
<artifactId>ms-medpipe</artifactId>
<version>1.0.0-Release</version>
<name>ms-medpipe</name>
```

3.5.2 Instalação

Após compilação do projeto 0 arquivo gerado "ms-medpipe-1.0.0-Release.jar" deve ser copiado para a pasta raiz do Medpipe. A execução do projeto é feita pelo comando:

"java -jar ms-medpipe-1.0.0-Release.jar"

Detalhes do comando

- **java**: É o executável da JVM, responsável por executar programas Java.
- **-jar**: Indica à JVM que o arquivo seguinte é um arquivo JAR executável. Isso significa que o conteúdo do arquivo JAR contém todas as informações necessárias para executar a aplicação.
- **ms-medpipe-1.0.0-Release.jar**: Este é o nome do arquivo JAR que será executado. Pode variar dependendo do contexto, mas geralmente indica que se trata de uma aplicação relacionada ao microsserviço chamado Medpipe, na versão 1.0.0, na versão de lançamento (Release).

Portanto, o comando completo está instruindo a JVM a executar a aplicação contida no arquivo JAR ms-medpipe-1.0.0-Release.jar iniciando o microsserviço.

3.6 Integração Pannotator

Nesta seção será descrito como o Pannotator se integra ao Medpipe através do ms-medpipe, utilizando os *endpoints*.

O Pannotator e o Medpipe são basicamente *scripts* bash que podem ser executados por linha de comando ou por suas interfaces web. Mas para que o Pannotator pudesse aproveitar as predições que o Medpipe gera, os dois sistemas precisam estar no mesmo servidor. Com o desenvolvimento do ms-medpipe o Pannotator passa a executar o Medpipe através de uma chamada de API exposta na internet. Na Figura 2 é mostrado como é feita essa integração.

A execução do Medpipe é feita pela da rota /v1/medpipe/run e foi incluída no Pannotator após a atualização do arquivo *target.fasta*. O formato FASTA é usado para armazenar sequências de nucleotídeos e proteínas.

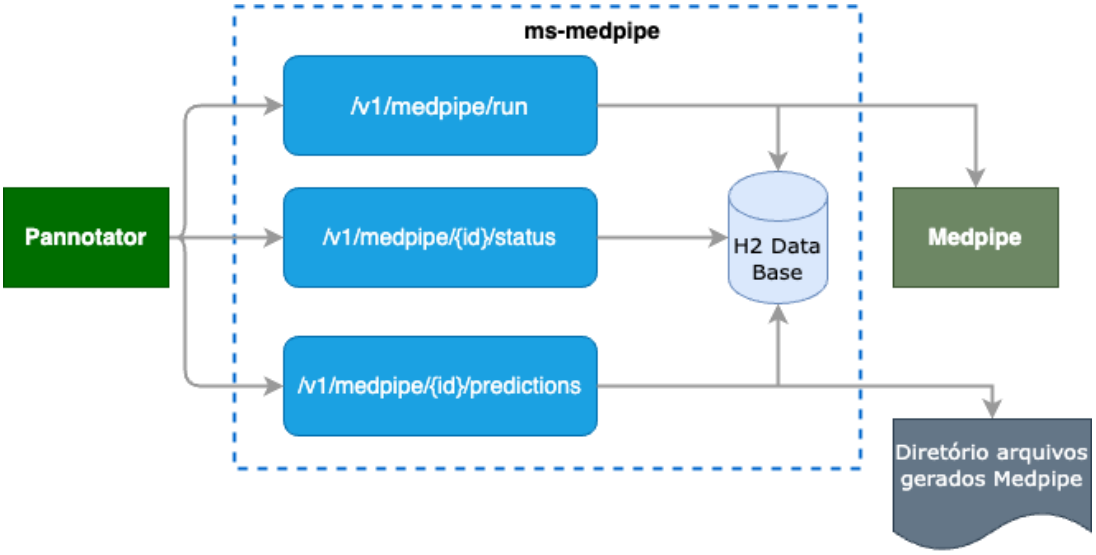

Figura 2 – Integração entre Pannotator e Medpipe.

O Código 3.5 define a variável *medpipePostURL* para armazenar um comando *curl* usado para enviar uma solicitação *POST* para a rota /v1/medpipe/run. Isso inclui a especificação da URL, do arquivo FASTA a ser enviado e dos parâmetros necessários para a análise. O comando especificado na variável *medpipePostURL* é executado e armazena a saída na variável *medpipeexec*. O retorno da API é composto pelo ID do processamento do Medpipe.

Código 3.5 – Código para chamada do *endpoint* Medpipe.

```
medpipePostURL="curl --location localhost:8190/v1/medpipe/run
                --form file=@$dirfastaFile
                --form cellWall=$cellWall
                --form organismGroup=$organismGroup
                --form email=$7"
processId=`$medpipePostURL`
```

O ID retornado pelo serviço de execução é utilizado para obter as informações do processo do Medpipe que foi iniciado. O *endpoint* de status recebe esse ID como parâmetro e retorna o status que se encontra o processamento. O Código 3.6 executa o comando GET para a API de status do processamento do Medpipe. Após a execução do Medpipe finalizar e o status for zero, o próximo passo é buscar as predições geradas pelo Medpipe.

Código 3.6 – Código chamada do *endpoint* de status.

```
getStatusUrl="curl --location
              localhost:8190/v1/medpipe/$processId/status"
statusexec=`$getStatusUrl`
```

O Código 3.7 executa a API de busca das predições geradas pelo Medpipe. Essas predições são usadas pelo Pannotator para enriquecer o resultado final de sua execução.

Código 3.7 – Código da chamada do *endpoint* de busca das predições.

```
getPredictionsUrl="curl --location
                   localhost:8190/v1/medpipe/
                   $processId/predictions"
medpipePredictions=‘$getPredictionsUrl‘
```

Nesta seção foi mostrado como foi construído o ms-medpipe atuando na integração do Pannotator ao Medpipe e quais alterações foram necessárias no Pannotator.

3.7 Documentação

Para facilitar o entendimento e a utilização do microsserviço, é essencial fornecer documentação clara. A seguir, são apresentados detalhes sobre como interagir com os *endpoints* do ms-medpipe.

3.7.1 Documentação dos *Endpoints*

A documentação dos *endpoints* REST está disponível na API Swagger, acessível em “http://localhost:[porta configurada]/swagger-ui.html”. Essa documentação fornece informações detalhadas sobre cada *endpoint*, conforme Figura 3, incluindo os parâmetros necessários, os métodos HTTP permitidos e as respostas esperadas.

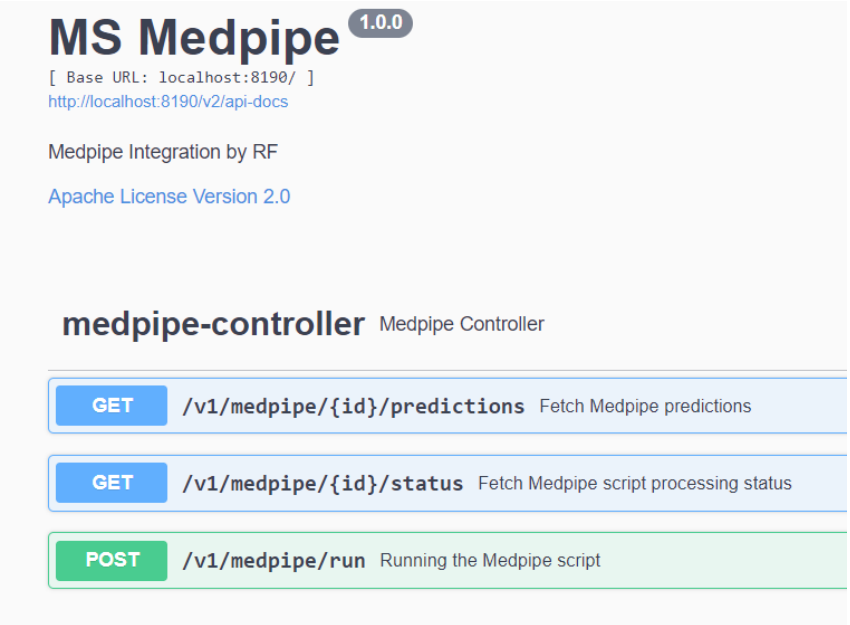

Figura 3 – API Swagger para documentação do ms-medpipe.

Isso facilita aos desenvolvedores e usuários compreenderem como interagir com o microsserviço.

3.8 Logs

Garantir uma visibilidade adequada das operações e do desempenho do microsserviço é fundamental. Para isso, foram implementados recursos de logs para análises de erros.

3.8.1 Logs Detalhados

O ms-medpipe gera logs detalhados, registrando informações relevantes sobre cada operação. Isso inclui informações sobre solicitações recebidas, respostas enviadas, eventos de erro e quaisquer atividades críticas, um exemplo pode ser verificado no Código 3.8.

Código 3.8 – Log de execução do Medpipe pelo do ms-medpipe.

```
[runFileProcess] - Init run...
[runFileProcess] - directoryRoot: /tmp/DB1703543511
[saveFile] - Start - File: target.fasta directoryRoot:
/tmp/DB1703543511
[saveFile] - Result File: /tmp/DB1703543511/target.fasta
[runFileProcess] - fileResult: /tmp/DB1703543511/target.fasta
[saveControl] - Start - medpipe process: DB1703543511
[runFileProcess] - process:
com.tcc.medpipe.MedpipeControl@22671154
[runFileProcess] - script terminated: /tmp/DB1703543511;1
[runScript] - Start exec: 2023-12-25T19:35:20.242966 commad:
sh medpipe /tmp/DB1703543511/target.fasta 50 1 9 teste@gmail.com
[runScript] - Time: 2023-12-25T21:19:27.948890 Result:
2) Cell wall thicnkess for local subcellular prediction of
/tmp/DB1703543511/target.fasta3) Bacterial Gram for local
subcellular prediction of /tmp/DB1703543511/target.fasta4)
Epitope length for MHC prediction
/tmp/DB1703543511/target.fasta 5)
Eliminate extra text from fasta headers /usr/local/medpipe
6) GRAMP 1
7) Prepare to run surfG. Configuring current directory of
instalation
<br />MEDPIPE results:<br />mygenome_00779 n=84
[50-avg(aff)]=32.67 d=131 MED=20.94 FOLD=1.56 PSE
```

4 Resultados

Neste capítulo são apresentados os resultados obtidos por meio da aplicação da metodologia proposta, que integra o sistema Medpipe ao Pannotator. Os resultados fornecem evidências de que a solução desenvolvida foi efetiva e trouxe informações detalhadas e valiosas sobre a anotação de genes e produtos genéticos.

4.1 Amostra de dados

Para ilustrar o impacto dessa integração foi gerado dois arquivos pelo Pannotator usando um arquivo fasta de DNA único com um corte de similaridade de 70% conforme Figura 4.

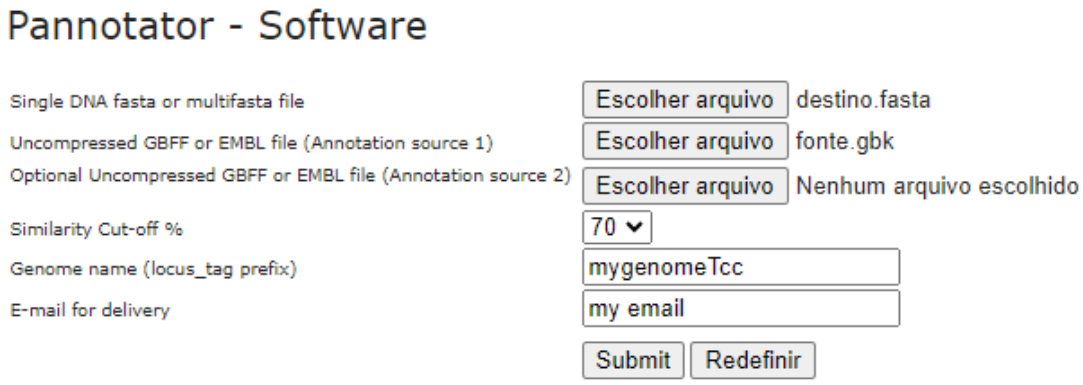

Figura 4 – Página Web de execução do Pannotator.

Foi retirado um trecho de cada arquivo gerado para fins de comparação, sendo o primeiro gerado exclusivamente pelo Pannotator (Código 4.1) e o segundo enriquecido com informações do Medpipe via integração com o ms-medpipe (Código 4.2). Esses trechos revelam como a anotação detalhada pode aprimorar a compreensão dos genes e proteínas de um organismo.

4.1.1 Resultados sem integração com ms-medpipe

Abaixo segue trecho gerado pelo Pannotator (Código 4.1) com o código original sem a integração feita neste trabalho. Nota-se que o trecho correspondente às informações são de uma anotação genômica em que o gene em questão está localizado no intervalo de coordenadas 1463551 a 1463883 e está identificado com o locus_tag mygenomeTcc_01444.

Código 4.1 – Resultado do Pannotator original.

```
FT    gene            1463551..1463883
FT                    /locus_tag="mygenomeTcc_01444"
FT                    /gene=""
FT                    /product="hypothetical protein YdbL precursor"
FT                    /annotation_source="ECP_1409"
FT    CDS             1463551..1463883
FT                    /gene=""
FT                    /product="hypothetical protein YdbL precursor"
FT                    /annotation_source="ECP_1409"
FT                    /colour=3
```

Na Figura 5 podemos ver o trecho selecionado através de uma ferramenta de anotação.

Figura 5 – Resultado Somente com Pannotator.

4.1.2 Resultados com integração com ms-medpipe

No próximo trecho do Código 4.2, tem-se uma parte do resultado gerado pelo Pannotator utilizando informações também geradas pelo Medpipe, tais informações foram possíveis graças a integração feita através dos *endpoints* do ms-medpipe.

Código 4.2 – Resultado Pannotator com integração ao ms-medpipe.

```
FT   gene            1463551..1463883
FT                   /locus_tag="mygenomeTcc_01444"
FT                   /gene=""
FT                   /product="hypothetical protein YdbL precursor"
FT                   /annotation_source="ECP_1409"
FT   CDS             1463551..1463883
FT                   /note="Mature Epitope Density (MED): 5.12"
FT                   /GO_component="SECRETED"
FT                   /gene=""
FT                   /product="hypothetical protein YdbL precursor"
FT                   /annotation_source="ECP_1409"
FT                   /colour=3
```

Na Figura 6 podemos ver o trecho selecionado com as informações geradas pelo Medpipe.

Figura 6 – Resultado com Pannotator integrado ao ms-medpipe.

4.1.3 Comparação de resultados

A comparação entre esses trechos destaca a importância das informações fornecidas pelo Medpipe na forma de anotações adicionais. Na Figura 6, nota-se a inclusão da anotação "Mature Epitope Density (MED): 5.12" e "GO_component=SECRETED" que fornece informações detalhadas sobre a densidade de epítopos maduros e a classificação da proteína em relação ao seu componente no *Gene Ontology*.

4.2 Discussão

A integração do Medpipe com o Pannotator enriqueceu significativamente nossa análise genômica, fornecendo informações adicionais que são essenciais para a compreensão da biologia e funcionalidade dos organismos em estudo.

Essas informações têm o potencial de impulsionar descobertas científicas e pesquisas futuras, contribuindo para uma compreensão mais abrangente e detalhada da genômica e das proteínas de interesse.

O fato da revisão de literatura sobre ferramentas similares ao Medpipe, nos últimos dez anos, ter retornado tão poucos resultados nos faz endossar a hipótese que as indústrias farmacêuticas não estariam interessadas em implementar vacinas contra muitas das doenças infectocontagiosas transmitidas por bactérias para as quais hoje possuímos antibióticos. O desenvolvimento de uma vacina pode durar décadas e envolver bilhões de dólares em gastos (RAPPUOLI et al., 2018). O motivo deste desinteresse por soluções definitivas contra doenças infecciosas estaria na venda de antibióticos e anti-inflamatórios, um dos principais pilares de sustentação das indústrias farmacêuticas e da empregabilidade de médicos que, em teoria, controlam a liberação destes medicamentos. A venda de antibióticos não cessa nunca, ao passo que poucas doses de uma vacina significam o fim do comércio de milhões ou bilhões em antibióticos.

Outro motivo é o fato de muitas das doenças bacterianas impactarem países subdesenvolvidos ou em desenvolvimento (doenças negligenciadas pelos países ricos). Os principais interessados em vacinas são países que não podem desenvolvê-las. Ainda assim, estes países serão clientes de longa data de empresas farmacêuticas ao comprarem antibióticos e anti-inflamatórios.

5 Conclusão

A integração do Medpipe ao Pannotator representa um avanço significativo na capacidade de análise e interpretação de dados genômicos. Os resultados obtidos evidenciam a eficácia dessa integração, fornecendo informações detalhadas sobre a anotação de genes e produtos genéticos, incluindo a densidade de epítopos maduros (MED) e a classificação da proteína em relação ao seu componente no *Gene Ontology* (GO).

A automatização do processo, proporcionada pela integração com o microsserviço ms-medpipe, simplifica e agiliza as análises genômicas.

A aplicabilidade da integração se estende a projetos e pesquisas que demandam uma compreensão mais profunda e abrangente dos dados genômicos. A disponibilização de *endpoints* do ms-medpipe permite futuras expansões e integrações com outras ferramentas e serviços, proporcionando uma plataforma flexível e adaptável às necessidades em constante evolução da pesquisa genômica.

Portanto, a integração bem-sucedida do Medpipe ao Pannotator utilizando um microsserviço chamado ms-medpipe representa um avanço significativo na capacidade de análise e interpretação de dados genômicos, proporcionando uma abordagem mais completa, informada e eficiente para a anotação genômica.

Referências

ABBY, S. S.; ROCHA, E. P. C. Identification of protein secretion systems in bacterial genomes using macsyfinder. **Methods in molecular biology (Clifton, N.J.)**, United States, v. 1615, p. 1–21, 2017. Citado na página 20.

Apache Software Foundation. **Maven**. [S.l.], 2023. Disponível em: <https://maven.apache.org/>. Acesso em: 12 out 2023. Citado na página 23.

BRETTIN, T.; DAVIS, J. J.; DISZ, T.; EDWARDS, R. A.; GERDES, S.; OLSEN, G. J.; OLSON, R.; OVERBEEK, R.; PARRELLO, B.; PUSCH, G. D.; SHUKLA, M.; THOMASON, J. A. r.; STEVENS, R.; VONSTEIN, V.; WATTAM, A. R.; XIA, F. Rasttk: a modular and extensible implementation of the rast algorithm for building custom annotation pipelines and annotating batches of genomes. **Scientific reports**, England, v. 5, p. 8365, Feb 2015. Citado na página 17.

BRITO, R. T. d.; SOARES, J. A. R. Alinhamento de sequências biológicas. 2003. Citado na página 12.

CASIMIRO-SORIGUER, C. S.; MUÑOZ-MÉRIDA, A.; PÉREZ-PULIDO, A. J. Sma3s: A universal tool for easy functional annotation of proteomes and transcriptomes. **Proteomics**, Germany, v. 17, Jun 2017. Citado na página 18.

FOWLER, M. Microservices. **MartinFowler.com**, 2014. Disponível em: <https://martinfowler.com/articles/microservices.html>. Acesso em: 12 out 2023. Citado 2 vezes nas páginas 13 e 16.

FOWLER, S. J. **Microsserviços prontos para a produção: Construindo sistemas padronizados em uma organização de engenharia de software**. [S.l.]: Novatec Editora, 2017. Citado na página 14.

GONÇALVES, A. R. A bioinformática como aliada da biotecnologia agrícola. **BIOINFO**, v. 3, p. 07, 2023. ISSN 2764-8273. Citado na página 12.

GUPTA, R.; PRADHAN, D.; JAIN, A. K.; RAI, C. S. Tid: Standalone software for mining putative drug targets from bacterial proteome. **Genomics**, United States, v. 109, p. 51–57, Jan 2017. Citado na página 20.

H2 Database Engine. **H2**. [S.l.], 2023. Disponível em: <https://h2database.com/>. Acesso em: 12 out 2023. Citado na página 23.

HE, Y.; RACZ, R.; SAYERS, S.; LIN, Y.; TODD, T.; HUR, J.; LI, X.; PATEL, M.; ZHAO, B.; CHUNG, M.; OSTROW, J.; SYLORA, A.; DUNGARANI, P.; ULYSSE, G.; KOCHHAR, K.; VIDRI, B.; STRAIT, K.; JOURDIAN, G. W.; XIANG, Z. Updates on the web-based violin vaccine database and analysis system. **Nucleic acids research**, England, v. 42, p. D1124–32, Jan 2014. Citado na página 20.

HILLPOT, J. 4 microservices examples: Amazon, netflix, uber, and etsy. **DreamFactory Blog**, 2023. Disponível em: <https://blog.dreamfactory.com/microservices-examples/>. Citado na página 14.

HOSSAIN, M. D.; SULTANA, T.; AKHTER, S.; HOSSAIN, M. I.; THU, N. T.; HUYNH, L. N.; LEE, G.-W.; HUH, E.-N. The role of microservice approach in edge computing: Opportunities, challenges, and research directions. **ICT Express**, Elsevier, 2023. Citado na página 13.

JAISWAL, V.; CHANUMOLU, S. K.; GUPTA, A.; CHAUHAN, R. S.; ROUT, C. Jenner-predict server: prediction of protein vaccine candidates (pvcs) in bacteria based on host-pathogen interactions. **BMC bioinformatics**, England, v. 14, p. 211, Jul 2013. Citado na página 19.

KREMER, F. S.; ESLABÃO, M. R.; DELLAGOSTIN, O. A.; PINTO, L. d. S. Genix: a new online automated pipeline for bacterial genome annotation. **FEMS microbiology letters**, England, v. 363, Dec 2016. Citado na página 18.

LEUNG, E.; HUANG, A.; CADAG, E.; MONTANA, A.; SOLIMAN, J. L.; ZHOU, C. L. E. Protein sequence annotation tool (psat): a centralized web-based meta-server for high-throughput sequence annotations. **BMC bioinformatics**, England, v. 17, p. 43, Jan 2016. Citado na página 18.

LOBO, I. Basic local alignment search tool (blast). **Nature Education**, v. 1, n. 1, 2008. Citado na página 13.

MENDE, D. R.; LETUNIC, I.; HUERTA-CEPAS, J.; LI, S. S.; FORSLUND, K.; SUNAGAWA, S.; BORK, P. progenomes: a resource for consistent functional and taxonomic annotations of prokaryotic genomes. **Nucleic acids research**, England, v. 45, p. D529–D534, Jan 2017. Citado na página 18.

NAGPAL, G.; USMANI, S. S.; RAGHAVA, G. P. S. A web resource for designing subunit vaccine against major pathogenic species of bacteria. **Frontiers in immunology**, Switzerland, v. 9, p. 2280, 2018. Citado na página 19.

Netflix Technology Blog. Rebuilding netflix video processing pipeline with microservices. **Netflix TechBlog**, 2023. Disponível em: <https://netflixtechblog.com/rebuilding-netflix-video-processing-pipeline-with-microservices-4e5e6310e359>. Citado na página 14.

NEWMAN, S. **Building microservices**. [S.l.]: "O'Reilly Media, Inc.", 2021. Citado na página 14.

RAPPUOLI, R.; PIZZA, M.; MASIGNANI, V.; VADIVELU, K. Meningococcal B vaccine (4CMenB): the journey from research to real world experience. **Expert Review of Vaccines**, Taylor & Francis, v. 17, n. 12, p. 1111–1121, 2018. ISSN 17448395. Disponível em: <https://doi.org/10.1080/14760584.2018.1547637>. Citado na página 37.

RODRIGUES, D. Desafios na padronização da anotação genômica. **BIOINFO**, v. 3, n. 05, 2023. ISSN 2764-8273. Citado na página 12.

SALLET, E.; GOUZY, J.; SCHIEX, T. Eugene: An automated integrative gene finder for eukaryotes and prokaryotes. **Methods in molecular biology (Clifton, N.J.)**, United States, v. 1962, p. 97–120, 2019. Citado na página 18.

SANTOS, A.; BARBOSA, E.; FIAUX, K.; ZURITA-TURK, M.; CHAITANKAR, V.; KAMAPANTULA, B.; ABDELZAHER, A.; GHOSH, P.; TIWARI, S.; BARVE, N.; JAIN, N.; BARH, D.; SILVA, A.; MIYOSHI, A.; AZEVEDO, V. Pannotator: an automated tool for annotation of pan-genomes. **Genetics and molecular research : GMR**, v. 12, p. 2982–9, 08 2013. Citado na página 12.

SANTOS, A.; PEREIRA, V.; BARBOSA, E.; BAUMBACH, J.; PAULING, J.; RöTTGER, R.; TURK, M.; SILVA, A.; MIYOSHI, A.; AZEVEDO, V. Mature epitope density - a strategy for target selection based on immunoinformatics and exported prokaryotic proteins. **BMC Genomics**, v. 14, p. S4, 10 2013. Citado na página 12.

SAUDATE, A. **APIs REST em Kotlin: Seus serviços prontos para o mundo real**. [S.l.]: Casa do Código, 2021. Acesso em: 08 jul 2021. Citado na página 22.

SEEMANN, T. Prokka: rapid prokaryotic genome annotation. **Bioinformatics**, v. 30, n. 14, p. 2068–2069, 03 2014. ISSN 1367-4803. Disponível em: <https://doi.org/10.1093/bioinformatics/btu153>. Citado na página 17.

SOUZA, E. V. de; DALBERTO, P. F.; MACHADO, V. P.; CANEDO, A.; SAGHATELIAN, A.; MACHADO, P.; BASSO, L. A.; BIZARRO, C. V. µproteins-a proteogenomics pipeline for finding novel bacterial microproteins encoded by small orfs. **Bioinformatics (Oxford, England)**, England, v. 38, p. 2612–2614, Apr 2022. Citado na página 19.

Spring. **Spring Boot**. 2023. Disponível em: <https://spring.io/projects/spring-boot>. Acesso em: 10 ago 2023. Citado na página 22.

TANIZAWA, Y.; FUJISAWA, T.; ARITA, M.; NAKAMURA, Y. Generating publication-ready prokaryotic genome annotations with dfast. **Methods in molecular biology (Clifton, N.J.)**, United States, v. 1962, p. 215–226, 2019. Citado na página 18.

TOBES, R.; PAREJA-TOBES, P.; MANRIQUE, M.; PAREJA-TOBES, E.; KOVACH, E.; ALEKHIN, A.; PAREJA, E. Gene calling and bacterial genome annotation with bg7. **Methods in molecular biology (Clifton, N.J.)**, United States, v. 1231, p. 177–89, 2015. Citado na página 17.

VELEPUCHA, V.; FLORES, P. A survey on microservices architecture: Principles, patterns and migration challenges. **IEEE Access**, IEEE, 2023. Citado na página 13.

YERO, D.; CONCHILLO-SOLé, O.; DAURA, X. Antigen discovery in bacterial panproteomes. **Methods in molecular biology (Clifton, N.J.)**, United States, v. 2183, p. 43–62, 2021. Citado na página 20.

ZHAO, B.; KATUWAWALA, A.; OLDFIELD, C. J.; DUNKER, A. K.; FARAGGI, E.; GSPONER, J.; KLOCZKOWSKI, A.; MALHIS, N.; MIRDITA, M.; OBRADOVIC, Z.; SöDING, J.; STEINEGGER, M.; ZHOU, Y.; KURGAN, L. Describeprot: database of amino acid-level protein structure and function predictions. **Nucleic acids research**, England, v. 49, p. D298–D308, Jan 2021. Citado na página 18.