**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído**

Projeto Integrado

Relatório Técnico

MASA-Webaligner 2.0

Bernardo Costa Nascimento

Belo Horizonte

Dezembro, 2021.

# Projeto Integrado – Arquitetura de Software Distribuído

**Sumário**

[Projeto Integrado – Arquitetura de Software Distribuído 2](#_heading=h.3dy6vkm)

[1.](#_heading=h.z337ya) Introdução 3

[2.](#_heading=h.3j2qqm3) Cronograma do Trabalho 5

[3.](#_heading=h.1y810tw) Especificação Arquitetural da solução 6

[3.1](#_heading=h.4i7ojhp) Restrições Arquiteturais 6

[3.2](#_heading=h.e4qgsirqegbo) Requisitos Funcionais 6

[3.3](#_heading=h.2xcytpi) Requisitos Não-funcionais 7

[3.4](#_heading=h.1ci93xb) Mecanismos Arquiteturais 7

[4.](#_heading=h.3whwml4) Modelagem Arquitetural 8

[4.1](#_heading=h.2bn6wsx) Diagrama de Contexto 8

[4.2](#_heading=h.qsh70q) Diagrama de Container 9

[4.3](#_heading=h.3as4poj) Diagrama de Componentes 9

[5.](#_heading=h.1pxezwc) Prova de Conceito (PoC) 11

[5.1](#_heading=h.49x2ik5) Integrações entre Componentes 11

[5.2](#_heading=h.2p2csry) Código da Aplicação 11

[6.](#_heading=h.147n2zr) Avaliação da Arquitetura (ATAM) 13

[6.1.](#_heading=h.3o7alnk) Análise das abordagens arquiteturais 13

[6.2.](#_heading=h.ihv636)  Cenários 13

[6.3.](#_heading=h.32hioqz) Evidências da Avaliação 14

[6.4. Resultados Obtidos 15](#_heading=h.1hmsyys)

[7.](#_heading=h.41mghml) Avaliação Crítica dos Resultados 16

[8.](#_heading=h.2grqrue) Conclusão 17

[Referências 18](#_heading=h.vx1227)

## Introdução

A Bioinformática vem ganhando cada vez mais importância, especialmente nos últimos dois anos com a pandemia de COVID-19. Cientistas, e até mesmo cidadãos comuns, uniram-se nos esforços computacionais para auxiliar em pesquisas que buscavam por novos remédios e tratamentos para o COVID-19 e outras doenças [1].

Uma das possíveis áreas de aplicação de Bioinformática são os alinhamentos genéticos, utilizados para determinar similaridades entre sequências genéticas, determinar funções, estruturas e informações evolutivas. Para isso, é importante que se obtenha o melhor alinhamento possível, chamado de “alinhamento ótimo” [2].

Existem vários algoritmos capazes de encontrar alinhamentos ótimos [2]. Porém, esses algoritmos possuem um alto custo computacional - temporal e de memória - e, muitas vezes, não podem ser obtidos em tempo razoável ou por falta de recursos.

Para contornar essas limitações, pesquisadores desenvolveram algoritmos heurísticos, como o FASTA [3] e o BLAST [4]. No entanto, apesar de conseguirem alinhar sequências longas em tempo razoável e com consumo moderado de memória, este tipo de algoritmo não garante a geração de alinhamentos ótimos.

Neste contexto, a MASA Genetics, uma empresa fictícia, em parceria com a Universidade de Brasília, desenvolveu o CUDAlign [6] (*Compute Unified Device Architecture*), visando utilizar a tecnologia CUDA [5], de placas gráficas da Nvidia, para obter paralelismo no processamento dos alinhamentos. Posteriormente, a empresa decidiu evoluir o CUDAlign para o Multi-Platform Architecture for Sequence Aligner (MASA) [6], uma plataforma que propõe facilitar o desenvolvimento de soluções de alinhamento para outras plataformas, além da arquitetura CUDA. Um exemplo é o MASA OpenMP (*Open Multi-Processing*), que pode utilizar processadores comuns para a realização dos alinhamentos.

Inicialmente o MASA seria um software para utilização em terminal. Contudo, a equipe do MASA Genetics percebeu que, embora seu software possuísse grande potencial, sua utilização por biólogos - público alvo da aplicação - era limitada, devido às dificuldades de uso por usuários leigos. Dessa maneira, a MASA Genetics decidiu evoluir seu software, transformando-o em uma plataforma Web, abstraindo toda a necessidade de instalação e aprendizado de linhas de comando complexas, e oferecendo uma interface de usuário simples e intuitiva para utilização. Assim, nasceu o MASA-Webaligner [7].

No entanto, após o lançamento do produto, verificou-se que o sistema de filas implementado para realização dos alinhamentos apresentava problemas quando os alinhamentos requisitados eram muito longos [7]. Ainda, a utilização em larga escala da plataforma apresentava gargalos devido à natureza das tecnologias utilizadas [7]. Também, ao solicitar os resultados, os cálculos realizados no *front-end* podem ter alto custo computacional [7], sendo necessário avaliar novas implementações para melhorar a experiência dos usuários. Por fim, a API do MASA-Webaligner foi construída em forma de um monólito, o que traz impactos negativos na manutenção e performance do sistema [7].

A MASA Genetics já possui, na forma do MASA, um produto líder de mercado. Em testes após o desenvolvimento, o MASA-CUDAlign conseguiu realizar o alinhamento dos cromossomos 5 do homem e do chimpanzé, com 249 milhões de bases, em apenas 53 minutos [6]. No entanto, devido às questões apresentadas acima, os usuários do MASA-Webaligner ainda não conseguem utilizar todo o potencial da ferramenta e, portanto, evoluções são necessárias na aplicação.

Assim, a MASA Genetics optou por realizar uma refatoração do MASA-Webaligner para que os problemas citados sejam sanados. Dessa maneira, o objetivo deste trabalho é apresentar a descrição do novo projeto arquitetural do MASA-Webaligner. Assim, foram definidos 3 objetivos para a evolução da aplicação:

* Permitir a requisição de alinhamentos genéticos longos, sem que haja perda da tarefa devido a *timeouts*;
* Eliminar gargalos por acessos a APIs externas;
* Tornar o MASA-Webaligner escalável e robusto, de forma que seja possível sua utilização em larga escala;

Assim, serão apresentados neste documento os requisitos arquiteturais, funcionais e não funcionais, bem como os diagramas da solução para o desenvolvimento da nova versão do MASA-Webaligner, que compreenda os objetivos acima.

## Cronograma do Trabalho

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| --- | --- | --- | --- |
| **De** | **Até** |
| 15 / 06 / 2022 | 15 / 06 / 2022 | 1. Definição do cronograma de atividades | Construção da Tabela de Cronograma |
| 15 / 06 / 2022 | 15 / 06 / 2022 | 2.Contextualização do projeto | Capítulo 1 deste trabalho |
| 15 / 06 / 2022 | 26 / 06 / 2022 | 3.Definição das restrições arquiteturais | Capítulo 3.1 deste trabalho |
| 15 / 06 / 2022 | 26 / 06 / 2022 | 4.Definição dos requisitos funcionais | Capítulo 3.2 deste trabalho |
| 15 / 06 / 2022 | 26 / 06 / 2022 | 5.Definição dos requisitos não funcionais | Capítulo 3.3 deste trabalho |
| 15 / 06 / 2022 | 26 / 06 / 2022 | 6.Definição dos mecanismos arquiteturais | Capítulo 3.4 deste trabalho |
| 15 / 06 / 2022 | 26 / 06 / 2022 | 7.Construção do diagrama de contexto | Diagrama de Contexto no modelo C4 |
| 15 / 06 / 2022 | 26 / 06 / 2022 | 8.Revisão do conteúdo da Etapa I | Arquivo .PDF para entrega da Etapa I |
| 15 / 06 / 2022 | 27 / 06 / 2022 | 9.Construção da apresentação da Etapa I em PPT | Apresentação da Etapa I do projeto |
| 15 / 06 / 2022 | 01 / 07 / 2022 | 10.Gravação da apresentação da Etapa I em vídeo | Vídeo da apresentação da Etapa I |
| 01 / 07 / 2022 | 01 / 07 / 2022 | 11.Publicação dos artefatos gerados, na Etapa I, no Github | Disponibilização do material para primeira avaliação dos professores |
| 27 / 06 / 2022 | 15 / 08 / 2022 | 12.Construção do diagrama de Contêineres | Diagrama de Contêineres no modelo C4 |
| 02 / 07 / 2022 | 03 / 07 / 2022 | 13.Construção do diagrama de Componentes | Diagrama de Componentes no modelo UML |
| 04 / 07 / 2022 | 06 / 07 / 2022 | 14.Construção dos Wireframes da *Proof of Concept* | Protótipos das telas |
| 07 / 07 / 2022 | 14 / 08 / 2022 | 15.Implementação dos 3 requisitos principais do projeto | Principais requisitos implementados |
| 15 / 08 / 2022 | 15 / 08 / 2022 | 16.Publicação dos artefatos, produzidos na Etapa II, no Github | Disponibilização do material para segunda avaliação dos professores |
| 16 / 08 / 2022 | 13 / 10 / 2022 | 17.Análise das abordagens arquiteturais | Capítulo 6.1 deste trabalho |
| 16 / 08 / 2022 | 13 / 10 / 2022 | 18.Demonstração dos cenários de teste | Capítulo 6.2 deste trabalho |
| 16 / 08 / 2022 | 13 / 10 / 2022 | 19.Evidências da avaliação | Capítulo 6.3 deste trabalho |
| 16 / 08 / 2022 | 13 / 10 / 2022 | 20. Resultados obtidos | Capítulo 6.4 deste trabalho |
| 16 / 08 / 2022 | 13 / 10 / 2022 | 21. Avaliação crítica dos resultados | Capítulo 7 deste trabalho |
| 16 / 08 / 2022 | 13 / 10 / 2022 | 22. Conclusão | Capítulo 8 deste trabalho |
| 14 / 10 / 2022 | 14 / 10 / 2022 | 23. Gravação da apresentação da Etapa III em vídeo | Vídeo da apresentação da Etapa III |
| 15 / 10 / 2022 | 15 / 10 / 2022 | 24. Publicação dos artefatos gerados, na Etapa III, no Github | Disponibilização do material para terceira avaliação dos professores |

## Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permite visualizar a macro arquitetura da solução.

## Restrições Arquiteturais

Ao implementar uma solução, deve-se pensar além do código e da infraestrutura que será utilizada [8].

Restrições arquiteturais podem impactar na maneira como uma solução é construída. Arquitetos devem, no início de um projeto, definir as restrições para que sejam contornadas e, em alguns casos, desafiadas [8].

Abaixo estão listadas as restrições arquiteturais para implementação da nova versão do MASA-Webaligner.

| **ID** | **Descrição** |
| --- | --- |
| R01 | Deve-se utilizar tecnologias da Microsoft para toda a plataforma MASA-Webaligner. |
| R02 | As APIs devem seguir o padrão RESTful. |
| R03 | Toda comunicação entre os sistemas do MASA-Webaligner deve ser feita por meio da plataforma AIS (*Azure Integration Services*). |
| R04 | Deve-se utilizar a ferramenta Azure DevOps (Boards, Git, CI e CD) para gerenciar todo o ciclo de desenvolvimento e sustentação da plataforma. |
| R05 | Para o desenvolvimento Web, deve-se utilizar o *framework* Angular, que já é de conhecimento da equipe de desenvolvimento. |

## Requisitos Funcionais

Requisitos funcionais ditam o que um dado sistema deve ser capaz de realizar [9], ou seja, descrevem as funcionalidades do sistema. Esses requisitos irão variar de acordo com o tipo de software, seus usuários, etc. Abaixo estão listados os requisitos funcionais do MASA-Webaligner.

| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A)\*** | **Prioridade**  **(B/M/A)\*** |
| --- | --- | --- | --- |
| RF01 | As requisições de alinhamentos devem ser enfileiradas e atendidas, sem que haja *timeout* após o início da execução. | M | A |
| RF02 | Os resultados dos alinhamentos, de usuários cadastrados, devem ser armazenados. | M | M |
| RF03 | Os resultados dos alinhamentos, de usuários cadastrados, devem ficar disponíveis para acesso a qualquer momento. | M | M |
| RF04 | Usuários podem, opcionalmente, se cadastrar no sistema MASA-Webalign. | B | B |
| RF05 | Usuários cadastrados podem se identificar por meio de suas credenciais. | M | B |
| RF06 | As requisições de alinhamentos via NCBI não devem ser blocantes. | M | A |
| RF07 | Usuários podem fornecer arquivos de sequências genéticas para alinhamentos. | M | M |
| RF08 | Usuários podem fornecer sequências genéticas textuais para alinhamentos. | B | B |
| RF09 | Usuários têm acesso ao total de requisições esperando para serem processadas pelo sistema. | A | M |
| RF10 | Ao finalizar uma requisição de alinhamento, o usuário requerente deve ser informado via e-mail. | M | A |
| RF11 | Usuários podem optar por alinhamentos via MASA-OpenMP (sequências de até 1MB em tamanho). | M | A |
| RF12 | Usuários podem optar por alinhamentos via MASA-CUDAlign. | M | A |
| RF13 | Usuários cadastrados podem acessar o histórico de alinhamentos requisitados. | B | M |
| RF14 | Usuários cadastrados podem verificar o *status* de cada alinhamento requisitado. | M | M |
| RF15 | Usuários cadastrados podem alterar informações do seu perfil. | B | B |
| RF16 | Usuários podem selecionar o tipo do alinhamento (global ou local). | B | A |
| RF17 | Usuários podem optar pela execução apenas do Estágio I do alinhamento. | B | A |
| RF18 | Usuários podem optar pela remoção dos caracteres desconhecidos (‘N’) no alinhamento. | B | A |
| RF19 | Usuários podem optar por desabilitar a otimização ‘Block Pruning’. | B | A |
| RF20 | Usuários podem optar por substituir as bases pelo seu complemento em qualquer uma das sequências ou em ambas. | B | A |
| RF21 | O usuário deve poder realizar o *download* dos resultados do alinhamento. | B | A |
| RF22 | O usuário deve ter acesso a um gráfico interativo dos resultados do alinhamento. | B | A |

\*B=Baixa, M=Média, A=Alta.

## Requisitos Não-funcionais

Em contraponto aos requisitos funcionais, os não-funcionais não estão diretamente ligados às funcionalidades do sistema [9]. Requisitos não funcionais estão ligados a questões como confiabilidade, usabilidade, entre outros [9]. Ainda, podem definir restrições ao sistema [9]. Abaixo, seguem os requisitos não-funcionais do MASA-Webaligner.

| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| --- | --- | --- |
| RNF01 | O sistema deve ser capaz de requisitar execuções do MASA-OpenMP. | A |
| RNF02 | O sistema deve ser capaz de requisitar execuções do MASA-CUDAlign. | A |
| RNF03 | O sistema deve habilitar autenticação e autorização por meio do Azure Active Directory | M |
| RNF04 | O sistema deve estar disponível 24 x 7 x 365 | A |
| RNF05 | O sistema deve ser compatível com a maioria dos navegadores modernos. | A |
| RNF06 | As notificações por e-mail devem ser realizadas por meio de filas de mensagens | A |

\*B=Baixa, M=Média, A=Alta.

## Mecanismos Arquiteturais

Mecanismos arquiteturais são definidos como “soluções comuns para um problema frequentemente encontrado” e apresentam quatro estágios: requisitos arquiteturais, mecanismos de análise, mecanismos de desenho e, por fim, mecanismos de implementação.

Como visto anteriormente (Seções 3.2 e 3.3), os requisitos fornecem descrições das funcionalidades e propriedades do sistema. Um exemplo de requisito poderia ser: disponibilizar dados de maneira indefinida.

Mecanismos de análise são influenciados pelos requisitos e definem uma solução para o problema imposto por tal requisito. Partindo do exemplo de requisito usado anteriormente, um mecanismo de análise seria a utilização de algum tipo de persistência de dados.

Mecanismos de desenho são o resultado do refinamento dos mecanismos de análise. Eles são influenciados por restrições ao desenho. Portanto, considerando o mecanismo de persistência, poderíamos ter como opções: bancos de dados relacionais, bancos de dados não relacionais e arquivos. No entanto, as restrições ao desenho podem limitar nossas opções a apenas os bancos de dados relacionais.

Por fim, os mecanismos de implementação são obtidos após refinar os nossos mecanismos de desenho. Portanto, definida a utilização dos bancos de dados relacionais, devemos, com base em nossas restrições à implementação, definir qual implementação de banco de dados será utilizada: MySQL, SQL Server, Postgre, etc.

Abaixo estão listados os três tipos de mecanismos arquiteturais para o sistema MASA-Webaligner.

| **Análise** | ***Design*** | **Implementação** |
| --- | --- | --- |
| Persistência | Banco de Dados Relacional | SQL Server |
| Persistência | ORM | Entity Framework Core |
| Persistência | ORM | Dapper |
| Persistência | Arquivos | Azure Blob Storage |
| Front end | Single Page Application | Angular |
| Back end | Microsserviços | ASP.NET Core |
| Back end | Serverless | Azure Functions |
| Integração | PaaS | Azure Integration Services |
| Log do sistema | Telemetria | Azure Application Insights |
| Teste de Software | Testes Unitários | xUnit |
| Teste de Software | Testes Integração | xUnit |
| Teste de Software | Testes Unitários | Karma |
| Confiabilidade | Eventos | Azure Event Grid |
| Confiabilidade | Service Bus | Azure Service Bus |
| Autenticação | OAuth2 | Azure Active Directory |
| Autorização | OAuth2 | Azure Active Directory |
| Distribuição | Integração e Entrega Contínuas (CI/CD) | Azure DevOps |

## 

## Modelagem Arquitetural

Modelos servem como uma simplificação da realidade e têm várias utilidades, dentre elas: prover estrutura para solução de um problema, produzir abstrações para tratar complexidade, facilitar comunicação entre pessoas, etc.

Ao implementar modelos, devem-se seguir os quatro princípios da modelagem, quais sejam:

1. Escolher o modelo mais adequado ao seu problema;
2. Use níveis diferentes de precisão;
3. Procure conectar o modelo à realidade;
4. Nenhum modelo é perfeito.

Existem diversos tipos de modelos: modelos de forma livre, modelos UML (em suas mais diversas possibilidades), modelos C4, etc.

Neste capítulo serão apresentados três modelos - diagrama de contexto, diagrama de container e diagrama de componentes -, que modelam a arquitetura do sistema MASA-Webaligner. Esses modelos irão servir de base para implementação da Prova de Conceito(PoC) (Seção 5).

## 4.1 Diagrama de Contexto

O MASA-Webaligner será composto de quatro serviços internos, são eles: MASA-Webaligner, MASA-Runner, MASA-Notifications e MASA-Sequences. Além desses serviços, serão utilizados o Azure Active Directory, MASA-CUDAlign, MASA-OpenMP e NCBI API como serviços externos.

Apesar de serem considerados serviços externos, o MASA-CUDAlign e o MASA-OpenMP são de responsabilidade da MASA Genetics, mas são sistemas utilizados para subrotinas externas.

O sistema irá funcionar a partir de requisições de usuários, cadastrados ou não, por alinhamentos de sequências genéticas. Caso o usuário deseje se cadastrar ou acessar sua conta, o fará por meio do Azure Active Directory, solução da Microsoft para lidar com autenticação e autorização de usuários.

Ao realizar a requisição, o MASA-Webaligner dará início a todo o processo de alinhamento. Este serviço ficará responsável por guardar as informações sobre os alinhamentos requisitados e processados pelo sistema.

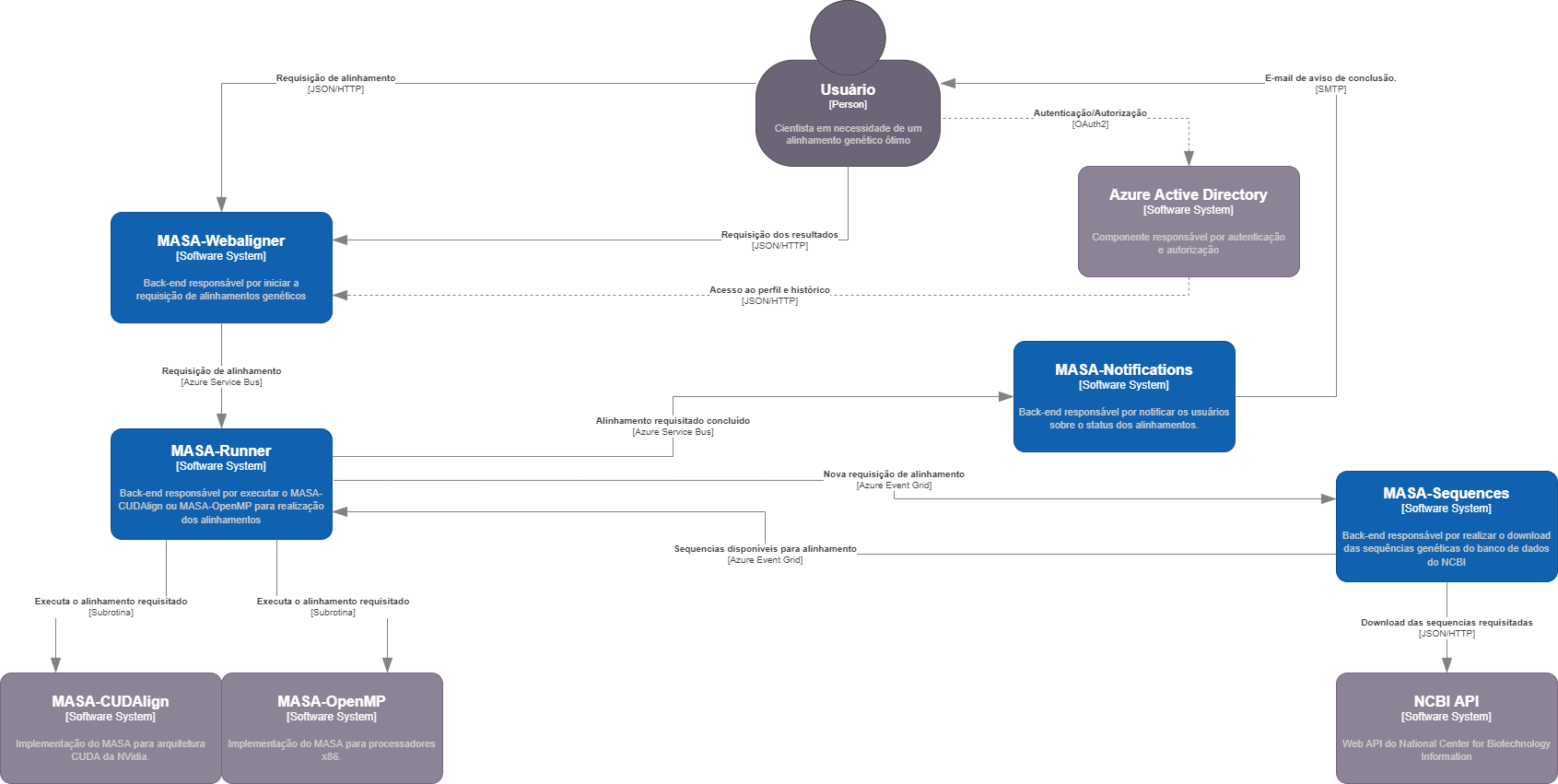
Após receber uma requisição, o MASA-Webaligner irá publicar um evento de novo alinhamento requisitado para o serviço MASA-Runner. Quando processar o novo evento, o MASA-Runner deverá publicar um novo evento de solicitação de arquivo de sequência genética.

O MASA-Sequences irá receber eventos de solicitação de arquivos de sequência genética e garantir que os arquivos estejam disponíveis para alinhamento, opcionalmente o MASA-Sequences pode acessar a NCBI API para *download* de arquivos de sequência genética.

Quando o MASA-Sequences tiver garantido os arquivos, irá publicar um novo evento para o MASA-Runner informando que os arquivos estão disponíveis para processamento.

O MASA-Runner irá buscar os arquivos disponibilizados pelo MASA-Sequences e irá chamar uma subrotina para processar o alinhamento, por meio do MASA-CUDAlign ou MASA-OpenMP. Apenas uma única subrotina poderá estar ativa a qualquer momento, para que todo alinhamento seja processado com máxima performance.

Após concluir o processamento do alinhamento, um último evento é publicado para que o MASA-Notifications dispare um e-mail para o usuário requerente, avisando-o de que seu alinhamento está concluído e pronto para acesso dos resultados.



***Figura 1 - Visão Geral da Solução MASA-Webaligner.***

## Etapa 2 - Pendente

<Data para entrega: 15/08/2022>

## Etapa 3 - Pendente

<Data para entrega: 15/10/2022>

## Referências

[1] COVID-19 - Folding@ home <https://foldingathome.org/diseases/infectious-diseases/covid-19/?lng=en>, acesso em 2022-06-13.

[2] Mount, David W.: Bioinformatics - Sequence and Genome Analysis. Cold Spring Harbor Laboratory Press, 2001.

[3] Pearson, William R. e David J. Lipman: Improved tools for biological sequence comparison. 85:2444–2448, 1988.

[4] Blast: Basic local alignment search tool. <https://blast.ncbi.nlm.nih.gov/> Blast.cgi, acesso em 2022-06-13.

[5] What is CUDA. <https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/>, acesso em 2022-06-13.

[6] Oliveira Sandes, Edans Flávius de: Algoritmos Paralelos Exatos e Otimizações para Alinhamento de Sequências Biológicas Longas em Plataformas de Alto Desempenho. Tese de Doutoramento, Universidade de Brasília, 2015.

[7] Nascimento, Bernardo Costa: MASA-Webaligner: Uma plataforma Web para Execução e Visualização de Alinhamentos de Sequências de DNA. Monografia, Universidade de Brasília, 2020.

[8] Lidando com restrições na arquitetura de software, <https://eximia.co/lidando-com-restricoes-na-arquitetura-de-software/>, acesso em 2022-06-15.

[9] Sommerville, Ian - Engenharia de Software. Pearson Addison-Wesley, 2007.

[10] Repositório do Github para o projeto MASA-Webaligner-PUC. <https://github.com/bernas1104/MASA-Webaligner-PUC>, acesso em 2022-07-01.

[11] PUC Minas - Arquitetura de Software Distribuído - Etapa I. <https://youtu.be/fH1ByNdUV1Y>, acesso em 2022-07-01.