P.PORTO

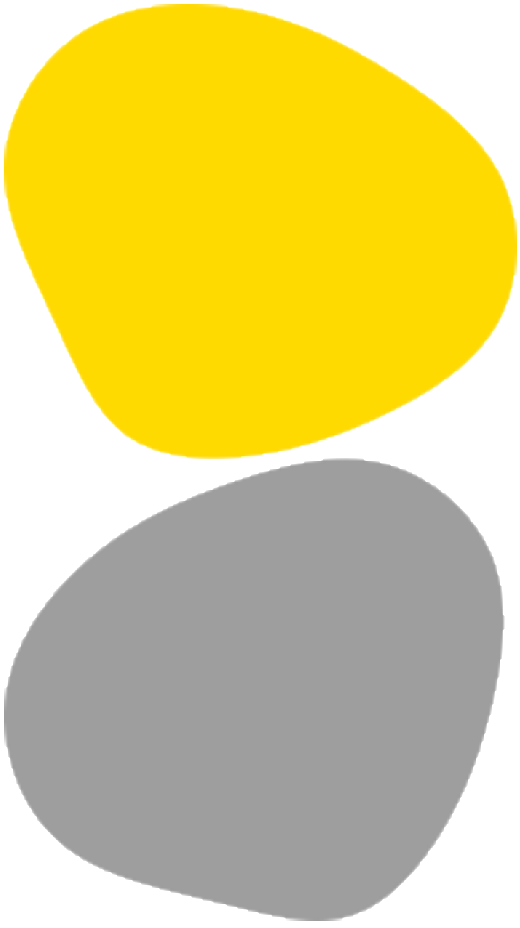
ESCOLA SUPERIOR DE SAÚDE

M MESTRADO

BIOESTATÍSTICA E BIOINFORMÁTICA APLICADAS À SAÚDE

Nova Biblioteca Biopython para suporte da Área da Biologia Molecular

Patrícia Maria Fernandes Nogueira



09/**2025**

Nova Biblioteca Biopython para Suporte da Área da Biologia Molecular

Autor

Patrícia Maria Fernandes Nogueira

Orientador(es)

Doutor / Vítor Júlio da Silva e Sá / Escola de Saúde – Politécnico do Porto

*Projeto apresentado para cumprimento dos requisitos necessários à obtenção do grau de Mestre em **Bioestatística e Bioinformática Aplicadas à Saúde** pela Escola Superior de Saúde do Instituto Politécnico do Porto.*

Agradecimentos

Agradeço à minha família, em especial ao meu marido, pela paciência que tiveram com toda a minha ausência enquanto estive a estudar e a assistir a aulas e, no entanto, sempre me encorajaram a chegar até aqui.

Agradeço também aos professores que partilharam o seu conhecimento de forma a conseguir articular tudo o que aprendi e canalizar neste projeto. Em especial, o meu muito obrigado ao Professor Vítor Sá, que me acompanhou nesta jornada e me deu liberdade para conduzir esta aventura da forma que achasse mais conveniente, dando-me sempre o apoio necessário.

Por fim, agradeço aos meus três colegas e amigos Joana Ribeiro, Rubén Ausina e Tânia Viana que fizeram parte dos grupos de trabalho ao longo do mestrado. Foi um grupo que se apoiou sempre e nunca nos deixou desistir.

Resumo

Introdução: O Biopython é uma biblioteca que facilita o desenvolvimento de aplicações para bioinformática, utilizando a linguagem de programação Python. É mantida por uma associação internacional de programadores, a Open Bioinformatics Foundation, que oferece ferramentas para a análise de sequências biológicas e acesso a bases de dados *online*, como o NCBI. A biblioteca conta com módulos para alinhamento de sequências, estruturas de proteínas, genética de populações, entre outros.

Objectivo: Sendo uma biblioteca de código aberto e contando com a colaboração de vários programadores, o objetivo do projeto centrou-se na criação de uma nova biblioteca que incorporou algumas ferramentas bioinformáticas que podem ser usadas individualmente ou em conjunto para obter informação acerca de uma sequência genética mistério. Assim, conseguiu-se que um único módulo fizesse referência a múltiplos serviços, obtivesse resultados e gerasse um relatório final para uma sequência pesquisada.

Métodos: Foi feito um estudo para perceber que ferramentas fariam sentido incorporar no projeto de forma a recolher informação sobre uma sequência genética. Foram então consideradas as seguintes ferramentas: ExPASy Translate Tool para fazer a tradução de uma sequência genética na sua proteína, Blast Tool para analisar a similaridade entre a proteína e uma sequência de referência, UniProt que através do Id retornado pela ferramenta anterior permite obter toda a informação sobre a proteína e variantes possíveis e patologias associadas, e por fim o DrugBank para pesquisar que fármacos podem ser relevantes para as patologias anteriores. Após a escolha de ferramentas, foi feita uma análise da biblioteca Biopython para perceber o que poderia oferecer já desenvolvido para cada uma destas ferramentas. Foi encontrado um módulo para o Blast, mas estava já descontinuado e foi substituído pelo ElasticBlast baseado na *cloud*, que acelera as pesquisas, mas com custos associados. UniProt, ExPASy e DrugBank não tinham ainda os serviços necessários e foram feitos de raiz.

Resultados: Como resultado deste projeto, foram adicionadas novas funcionalidades à biblioteca Biopython. Para apresentar os resultados de forma *user-friendly*, foram desenvolvidas aplicações *web* e *desktop*.

Conclusão: Sendo o Biopython uma biblioteca *open source*, será possível a incorporação de novas funcionalidades que podem ser úteis e gratuitas para escolas, laboratórios, investigação e programadores que as queiram usar e também melhorar.

Palavras-chave: Biopython; Blast; DrugBank; ExPASy; UniProt;

Abstract

Introduction: Biopython is a library that facilitates the development of bioinformatics applications using the Python programming language. It is maintained by an international association of programmers. The Open Bioinformatics Foundation offers tools for the analysis of biological sequences and access to *online* databases, such as NCBI. The library includes modules for sequence alignment, protein structures, and population genetics, among others.

Objective: As an open-source library with contributions from various programmers, the project's objective focused on creating a new library that incorporated several bioinformatics tools. These tools can be used individually or together to obtain information about an unknown genetic sequence. Thus, a single module was able to reference multiple services, obtain results, and generate a final report for a queried sequence.

Methods: A study was conducted to determine which tools would be most logical to incorporate into the project for gathering information about a genetic sequence. The following tools were considered: the ExPASy Translate Tool to translate a genetic sequence into its protein, the Blast Tool to analyze the similarity between that protein and a reference sequence, UniProt, which uses the ID returned by the previous tool to retrieve all information about the protein, including possible variants and associated pathologies, and finally, DrugBank to search for drugs that may be relevant to those pathologies. After selecting the tools, an analysis of the Biopython library was performed to see what it already offered for each of these tools. A module for Blast was found, but it was already discontinued and had been replaced by the cloud-based ElasticBlast, which speeds up searches but has associated costs. UniProt, ExPASy, and DrugBank did not yet have the necessary services, so they were built from scratch.

Results: As a result of this project, new functionalities were added to the Biopython library. To present the results in a user-friendly manner, *web* and *desktop* applications were developed.

Conclusion: As Biopython is an open-source library, it will be possible to incorporate new features that can be useful and free for schools, laboratories, research, and programmers who wish to use and improve them.

Keywords: Biopython; Blast; DrugBank; ExPASy; UniProt;

Índice

Índice de Figuras

Índice de Tabelas

Índice de Abreviaturas e Siglas

Abreviatura/Acrónimo	Significado
API	Application Programming Interface
AWS	Amazon *Web* Services
BLAST	Basic Local Alignment Search Tool
BWA	Burrows-Wheeler Aligner
BWA-MEM	Burrows-Wheeler Alignment algorithm
BWT	Burrows-Wheeler Transform
cDNA	Complementarity DNA
DDBJ	DNA Data Bank of Japan
DNA	Deoxyribonucleic acid, sendo também chamada de ácido desoxirribonucleico em português (ADN)
EMBL-EBI	European Molecular Biology Laboratory – European Bioinformatics Institute
ENA	European Nucleotide Archive
ExPASy	Expert Protein Analysis System
GATK	Genome Analysis Toolkit
GCP	Google Cloud Platform
GFF	General Feature Format
IDE	Integrated Development Environment (Ambiente de Desenvolvimento Integrado)
INDEL	INserções e DELeções
INSDC	International Nucleotide Sequence Database Collaboration
JSON	JavaScript Object Notation (Notação de Objetos JavaScript)
NCBI	National Center for Biotechnology Information
NGS	Sequenciação de Nova Geração
NIH	National Institutes of Health

OLC	Sobreposição-Layout-Consenso
ORF	Open Reading Frame
PyDPI	Drug-protein interaction with Python
PIR	Protein Information Resource
SCOP	Structural Classification of Proteins
SIB	Swiss Institute of Bioinformatics
SNP	Single Nucleotide Polymorphism
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
UniProt	Universal Protein
UniProtKb	Universal Protein Knowledgebase
URL	Uniform Resource Locator
VEP	Variant Effect Predictor
XML	Xtensible Markup Language (Linguagem de Marcação Extensível)

Glossário

Termo	Significado
API	É um conjunto de regras e definições que permite que diferentes programas comuniquem entre si.
Endpoint	Localização digital exposta pela API a partir da API que recebe e responde às consultas. Cada *endpoint* é uma URL (Uniform Resource Locator) que fornece a localização de um recurso no servidor da API.
FASTA	O formato FASTA é um formato de ficheiro usado para armazenar sequências biológicas, como sequências de ADN, ARN ou proteínas. É um dos formatos mais comuns em bioinformática.
Fork	No GitHub, um *fork* é uma cópia de um repositório que é criada na conta pessoal para ser possível editar livremente o código sem afetar o repositório original. O *fork* é usado quando alguém tem a intenção de contribuir para o projeto sem precisar de permissão imediata no repositório principal.
JSON	É leve e fácil de ler/escrever, sendo suportado em praticamente todas as linguagens de programação e é mais usado que XML em aplicações.
ORF	Uma ORF é uma sequência contínua de nucleótidos (no ADN ou ARN) que pode potencialmente ser traduzida numa proteína. Começa normalmente com um codão de início, que é geralmente o AUG (codifica o aminoácido metionina). Termina num codão de paragem (como UAA, UAG ou UGA). Entre o início e o fim, não pode haver codões de paragem — daí ser "aberta". É usada para identificar regiões codificantes no genoma (ou seja, onde podem estar os genes que codificam proteínas). É fundamental em áreas como a biologia molecular, genómica e bioinformática.
PyMOL	Sistema de visualização molecular.
Parsing	Processo de pegar em dados brutos, geralmente texto, e transformá-los numa estrutura de dados organizada e lógica que um programa de computador consegue entender e utilizar.
SQL	É uma linguagem de programação padronizada utilizada especificamente para gerir e manipular bases de dados relacionais.
Throttling	Processo para controlar a quantidade de pedidos que um cliente ou um conjunto de clientes pode fazer a uma API num determinado período de tempo. É um mecanismo de controlo de tráfego implementado pelo servidor da API.
URL	Termo técnico que foi traduzido para a língua portuguesa como "localizador uniforme de recursos". Um URL refere-se ao endereço de rede no qual se encontra algum recurso informático.
XML	A estrutura é hierárquica (como uma árvore). Muito usado em configurações, troca de dados entre sistemas (como APIs), documentos, etc. Usa uma sintaxe baseada em pares chave-valor (como em objetos do JavaScript).

1. Introdução

1.1. Contextualização

A Bioinformática é uma área em constante evolução, impulsionada pelo crescimento exponencial de dados biológicos e pela necessidade de métodos computacionais capazes de processar, analisar e interpretar informação complexa. Existem diversas ferramentas e bibliotecas desenvolvidas no âmbito da biologia molecular ao longo do tempo, tais como o ChemoPy , PyDPI, PyMOL, Biopython, entre outros. O **Biopython** ocupa um lugar de destaque, permitindo análises de sequências biológicas, acesso a bases de dados e integração de *pipelines* bioinformáticos.

Apesar da sua relevância, muitas tarefas em bioinformática continuam a ser executadas através do uso isolado de ferramentas como ExPASy, Blast, UniProt e DrugBank, o que torna o processo moroso, fragmentado e propenso a erros humanos. Esta realidade representa um desafio significativo para estudantes, investigadores e profissionais de saúde que necessitam de resultados rápidos e consistentes.

1.2. Problema de Identificado

O problema identificado é a ausência de uma solução integrada que permita conjugar, de forma automatizada, diferentes ferramentas bioinformáticas num só fluxo de trabalho. Atualmente, a execução manual destas etapas implica:

- recolher sequências em diferentes plataformas;
- interpretar resultados em múltiplos formatos;
- consolidar manualmente os *outputs* para gerar conclusões.

Este processo consome tempo, exige conhecimento avançado de cada ferramenta e pode levar a inconsistências na análise.

1.3. Proposta de Solução ao Problema: A Biblioteca Found Sequence

Para colmatar o problema identificado no ponto anterior, propõe-se o desenvolvimento da biblioteca ***"Found Sequence"***, uma solução integrada e de alto nível em Python. O objetivo desta biblioteca não é reinventar as ferramentas existentes, mas sim orquestrá-las de forma inteligente, criando um *pipeline* coeso e automatizado que abstrai a complexidade técnica do utilizador.

Esta biblioteca irá:

1. Agregar chamadas a ferramentas e APIs fundamentais como o ExPASy Translate Tool (para tradução de sequências), Blast+ (para pesquisa de homologia local), UniProt (para obtenção de anotação funcional rica) e DrugBank (para informação farmacogenómica), garantindo que as interfaces estejam sempre atualizadas e otimizadas.
2. Automatizar o fluxo de trabalho completo, desde a entrada de uma sequência de DNA ou proteína até à geração de um relatório final consolidado, eliminando a necessidade de passos manuais intermédios.
3. Incluir lógica analítica adicional para realizar análises de valor acrescentado, como a predição da ORF mais longa, a identificação de variantes em relação a uma sequência de referência e a sua anotação funcional automática.
4. Disponibilizar interfaces gráficas (*Desktop* e *Web*) para facilitar o seu uso por investigadores e estudantes que não sejam especializados em programação, democratizando o acesso a análises bioinformáticas complexas.

1.4. Comparação Sistemática – Ferramentas Existentes vs. Solução Proposta

Para evidenciar as vantagens da nova biblioteca que terá como nome "*Found Sequence*", a Tabela 1 apresenta uma comparação sistemática entre três abordagens comuns para a análise de sequências: o uso manual de ferramentas isoladas, a utilização da biblioteca Biopython para criar *scripts* customizados e a nova biblioteca proposta.

Tabela 1 – Comparação Sistemática – Ferramntas Existentes vs. Solução Proposta

Característica	Ferramentas Isoladas (Uso manual)	Biopython	Found Sequence
Fluxo de trabalho	Sequência Manual	Oferece blocos de construção, mas a integração é manual	Integrado e automatizado ("*All in One*")
Ferramentas suportadas	ExPASy, Blast, UniProt, etc.	Módulos para Blast (descontinuado), ExPASy, UniProt (limitados)	ExPASy, Blast+, UniProt, DrugBank (otimizados para o fluxo)

Acesso ao Drugbank	Via *website*	Inexistente	Integrado (via base de dados local)
Lógica Adicional	Inexistente (depende do utilizador)	Funções básicas de *parsing*	Identificação de maior ORF, deteção de variantes, etc.
Interface	Múltiplas interfaces *web*	Nenhuma (é uma biblioteca de código)	Aplicações *Desktop* e *Web*
Output	Múltiplos ficheiros/resultados	Objetos Python	Relatório unificado em PDF

A biblioteca "*Found Sequence*" representa uma evolução significativa na forma como os dados genéticos são processados e interpretados, promovendo a eficiência, reprodutibilidade e acessibilidade da análise bioinformática.

1.5. Estrutura da Dissertação

Este documento encontra-se organizado em sete capítulos. O primeiro capítulo introduz o tema, o problema de investigação, a hipótese, as perguntas e os objetivos do trabalho. O segundo capítulo apresenta o estado da arte, com a revisão da literatura e análise crítica das ferramentas existentes. O terceiro capítulo descreve a metodologia de investigação adotada, com particular foco na Metodologia de Investigação Construtiva. O quarto capítulo detalha o desenvolvimento da solução proposta. O quinto capítulo apresenta a avaliação da solução através de diferentes métricas. O sexto capítulo discute os resultados obtidos, as contribuições e as limitações. Finalmente, o sétimo capítulo apresenta as conclusões e propostas de trabalho futuro.

2. Estado da Arte

A bioinformática é uma área que está em constante evolução, tem vindo também a ser impulsionada pelo aumento exponencial de dados biológicos e pela necessidade de avanços tecnológicos a cada momento que passa. Atualmente as ferramentas de bioinformática existentes abrangem diversas áreas:

- Sequenciamento genético;
- Integração de dados, e bases de conhecimento;
- Fluxos de trabalho científico;
- Otimização de alguns *softwares* existentes.

2.1. Revisão de Literartura

A bioinformática consolidou-se como uma disciplina indispensável na biologia moderna, e é a ponte entre a enorme quantidade de dados biológicos e a sua interpretação funcional. O crescimento das tecnologias de Sequenciação de Nova Geração (NGS) no início do século XXI provocou uma revolução, reduzindo drasticamente os custos e o tempo necessários para sequenciamento de um genoma. Esta revolução gerou uma quantidade "astronómica" de dados que tornou a análise manual impraticável, exigindo o desenvolvimento de algoritmos e poder computacional robusto para extrair conhecimento biológico significativo.

A análise de dados genéticos tornou-se, assim, uma área central da investigação, com a bioinformática a desempenhar um papel crucial no desenvolvimento contínuo de ferramentas que permitem a reconstrução, alinhamento, anotação e interpretação de genomas. Um fluxo de trabalho típico em genómica computacional pode ser dividido em etapas sequenciais e interdependentes, cada uma com desafios próprios e um ecossistema de ferramentas especializadas. Estas etapas incluem a montagem do genoma (reconstruir o *puzzle* genético a partir de pequenos fragmentos), o alinhamento de sequências (comparar os fragmentos com um mapa de referência), a chamada de variantes (identificar diferenças genéticas) e a anotação funcional (atribuir um significado biológico a essas sequências e variantes).

Esta revisão foca-se nas principais ferramentas, bases de dados e bibliotecas que sustentam estes fluxos de trabalho bioinformático, oferecendo uma análise detalhada das suas funcionalidades, dos algoritmos subjacentes e das suas aplicações no campo da genética.

2.1.1. Sequenciação de Genomas

A sequenciação de genomas a partir de milhões de leituras curtas ou longas geradas por sequenciadores é uma etapa crítica e computacionalmente intensiva na análise genómica. A escolha do *assembler* depende fundamentalmente da tecnologia de sequenciação utilizada (leituras curtas como Illumina ou leituras longas como PacBio e Oxford Nanopore) e do tipo de genoma em análise (bacteriano, viral, eucariótico ou metagenoma).

As abordagens algorítmicas mais comuns são baseadas em grafos de Bruijn, predominantes para leituras curtas, e em estratégias de Sobreposição–*Layout*–Consenso (OLC) para leituras longas (Zhang et al., 2011). Ferramentas como SPAdes, MEGAHIT e Canu são amplamente utilizadas e representam diferentes otimizações para desafios específicos.

- **SPAdes – St. Petersburg genome assembler (Bankevich et al., 2012)**: Originalmente desenvolvido para sequenciamento de genomas bacterianos e virais, o SPAdes tornou-se particularmente eficaz em dados de célula única (*single-cell*). A principal inovação está relacionada com a utilização de múltiplos *k-mers* (*substrings* de comprimento k) para construir grafos de Bruijn de forma iterativa. Esta abordagem permite-lhe obter, tanto regiões conservadas (com *k-mers* longos) como regiões complexas e repetitivas (com *k-mers* curtos). Integra algoritmos avançados de correção de erros e combina a precisão das leituras curtas da Illumina com o alcance das leituras longas da PacBio.
- **MEGAHIT (Li et al., 2015)**: Em estudos de metagenómica, o desafio não é sequenciar um genoma, mas centenas ou milhares de genomas em simultâneo a partir de uma amostra ambiental complexa. O MEGAHIT foi projetado para esta tarefa, sendo ultrarrápido e com baixo consumo de memória, capaz de lidar com volumes massivos de dados em servidores de nó único. A sua eficiência advém da implementação de **grafos de Bruijn sucintos**, uma estrutura de dados que comprime o grafo de forma inteligente, mantendo apenas a informação essencial para a montagem. Esta otimização de memória torna-o ideal para análises de microbiomas (ex: intestinal, marinho), onde a diversidade de espécies é imensa.
- **Canu (Koren et al., 2017)**: Com a grande tendência das tecnologias de leitura longa (PacBio e Oxford Nanopore), que podem gerar sequências com dezenas de milhares de bases, surgiram novos desafios, como a elevada taxa de erro inerente a estas tecnologias. O Canu é um sequenciador especializado em leituras longas que aborda este problema através de um *pipeline* com três fases: correção, *trimming* e montagem. Na primeira fase, as leituras são alinhadas umas contra as outras para se corrigirem mutuamente. Na segunda, as extremidades de baixa

qualidade são aparadas. Finalmente, a sequenciação é realizada usando uma abordagem OLC otimizada. Esta metodologia robusta torna o Canu extremamente eficaz na resolução de regiões repetitivas complexas, permitindo a montagem de genomas completos e de cromossomas eucarióticos de alta continuidade.

2.1.2. Alinhamento de Sequências: Mapeamento no Genoma de Referência

Após a sequenciação , o alinhamento de sequências genéticas contra um genoma de referência é uma etapa essencial para identificar a origem genómica das leituras e detetar variantes, regiões conservadas ou níveis de expressão genética (no caso de dados de RNA-Seq). O desafio é realizar este mapeamento de forma extremamente rápida e precisa para milhares de milhões de leituras. Para tal, os alinhadores modernos utilizam estruturas de dados indexadas do genoma de referência.

- **BWA (Burrows-Wheeler Aligner)**: O BWA é um dos alinhadores mais conhecidos para leituras curtas. A sua eficiência deve-se ao uso do algoritmo **Burrows-Wheeler Transform (BWT)**, que reorganiza o genoma de referência numa estrutura que permite uma pesquisa de *substrings* extremamente rápida. O BWA-MEM, o seu algoritmo mais recente, é versátil, suportando alinhamentos locais (mapeando apenas uma parte da leitura) e globais (mapeando a leitura inteira), sendo também capaz de lidar com leituras mais longas e de identificar "*split-alignments*", onde uma única leitura mapeia em duas localizações genómicas distintas, o que é crucial para detetar variantes estruturais. É uma ferramenta padrão em muitos *pipelines* de análise de NGS.
- **Bowtie2 (Langmead & Salzberg, 2012)**: Tal como o BWA, o Bowtie2 utiliza um índice FM para um alinhamento rápido. No entanto, foi otimizado para permitir tolerância a *gaps* (lacunas) e desalinhamentos, tornando-o ideal para dados que possam conter mais erros ou para leituras um pouco mais longas que as da primeira geração de sequenciadores Illumina. Utiliza programação dinâmica acelerada para encontrar o melhor alinhamento possível, equilibrando velocidade e sensibilidade. É frequentemente preferido em aplicações onde a velocidade é crítica e onde se esperam algumas divergências em relação à referência.
- **Minimap2 (Li, 2018)**: O Minimap2 representa a nova geração de alinhadores, sendo otimizado tanto para leituras curtas como, especialmente, para as leituras longas e ruidosas da PacBio e Oxford Nanopore. A arquitetura baseia-se numa abordagem de ***seed-chain-align***. Primeiro, identifica pequenas sequências de correspondência exata (*minimizers*) entre as leituras e a referência. Em seguida, agrupa estes "*seeds*" em cadeias para formar regiões de alinhamento

candidatas. Finalmente, realiza um alinhamento detalhado apenas nessas regiões. Esta estratégia é altamente eficiente em termos de tempo e memória, tornando o Minimap2 a ferramenta de eleição para o mapeamento de DNA genómico, transcritos (cDNA) e para alinhamentos de genoma a genoma.

2.1.3. Chamada de Variantes

A deteção de variantes genéticas (SNPs, InDels, variantes estruturais) a partir de dados de sequenciação alinhados é fundamental para a maioria dos estudos de genética, desde a investigação de doenças hereditárias e cancro até à biologia evolutiva. Este processo não se limita a encontrar discrepâncias entre as leituras e a referência, requer modelos estatísticos sofisticados para distinguir verdadeiras variantes biológicas de erros de sequenciação e de mapeamento. No panorama atual da bioinformática, diversas ferramentas foram desenvolvidas para esta finalidade, destacando-se algumas pela sua robustez, precisão e ampla adoção pela comunidade científica:

- **GATK – Genome Analysis Toolkit (Van der Auwera & O'Connor, 2020):** O GATK é considerado o "padrão-ouro" para a chamada de variantes hereditárias e somáticas (adquiridas, como no cancro). Desenvolvido pelo Broad Institute, é um conjunto de ferramentas extremamente robusto que implementa um *pipeline* de "*Best Practices*". Módulos como o **HaplotypeCaller** realizam uma sequenciação local nova das regiões que mostram evidências de variação, permitindo uma deteção muito mais precisa de SNPs e, especialmente, de InDels. Para garantir a fiabilidade dos resultados, o GATK inclui ferramentas como o **VariantRecalibrator**, que utiliza *machine-learning* para construir um modelo de erro e atribuir uma pontuação de qualidade a cada variante, filtrando potenciais falsos positivos. A sua arquitetura é compatível com ambientes de computação em nuvem e suporta workflows reprodutíveis, o que é essencial para a investigação clínica.
- **FreeBayes (Garrison & Marth, 2012):** O FreeBayes oferece uma abordagem alternativa à chamada de variantes, utilizando um modelo estatístico bayesiano. Em vez de avaliar cada posição genómica de forma isolada, o FreeBayes analisa pequenos segmentos de leituras (haplótipos) para calcular a probabilidade de um determinado genótipo existir, dadas as observações (as leituras alinhadas). Esta abordagem baseada em haplótipos torna-o particularmente poderoso na deteção de variantes complexas, como múltiplos SNPs e InDels próximos uns dos outros, e em regiões com múltiplos alelos (polialélicas). É uma ferramenta de código aberto, flexível e amplamente utilizada pela comunidade académica.

2.1.4. Anotação Genómica

Uma vez que um genoma é montado e as suas variantes são identificadas, a etapa seguinte é a anotação, que consiste em atribuir significado biológico às sequências de DNA. Este processo divide-se em duas fases: anotação estrutural (identificar a localização de genes, exões, intrões, etc.) e anotação funcional (determinar a função biológica dos produtos desses genes, como as proteínas). A complexidade e o volume de dados genómicos tornam a anotação manual impraticável, sendo essencial o uso de ferramentas bioinformáticas robustas para automatizar este processo. Algumas das principais são:

- **Prokka (Seemann, 2014)**: Especialmente desenhado para genomas procarióticos (bactérias e arqueias), o Prokka automatiza o processo de anotação de forma rápida e eficiente. Ele integra um conjunto de ferramentas de predição de genes (como o Prodigal), de identificação de RNAs ribossomais e de transferência, e de pesquisa em bases de dados hierárquicas (como UniProt) para atribuir nomes e funções às proteínas preditas. No final, produz ficheiros de anotação em formatos padrão (como GenBank e GFF), que são compatíveis com a maioria dos visualizadores genómicos e podem ser submetidos diretamente a bases de dados públicas.
- **InterProScan (Paysan-Lafosse et al., 2023)**: A anotação funcional de proteínas é crucial para entender o seu papel na célula. O InterProScan é uma ferramenta poderosa que não se baseia numa única base de dados, mas sim integra múltiplas fontes de assinaturas proteicas. Ele analisa uma sequência de proteína e pesquisa por domínios funcionais, famílias de proteínas e sítios ativos em bases de dados como Pfam, SMART, PROSITE, entre outras. Ao consolidar os resultados de várias fontes, fornece uma visão muito mais completa da função da proteína. A sua versão mais recente foi atualizada para incluir também dados estruturais preditos pela revolucionária ferramenta AlphaFold, adicionando uma camada de informação tridimensional à anotação funcional.
- **SnpEff (Cingolani et al., 2012)**: Para estudos de variação genética, não basta encontrar uma variante; é preciso prever o seu impacto biológico. O SnpEff é uma ferramenta de anotação de variantes que realiza exatamente isso. A partir de um ficheiro de variantes e de um modelo genómico de referência, o SnpEff classifica o impacto de cada mutação. Determina se a variante se localiza numa região codificante ou não, e, se for codificante, prediz o seu efeito na proteína: pode ser uma mutação *sinónima* (não altera o aminoácido), *missense* (troca um aminoácido por

outro), *nonsense* (cria um codão de paragem prematuro) ou *frameshift* (altera a matriz de leitura). Esta classificação é vital para priorizar variantes em estudos de doenças genéticas.

2.1.5. Bases de Dados Genómicas

A análise bioinformática depende criticamente da existência de bases de dados públicas, centralizadas e bem curadas, que servem como repositórios de sequências, anotações, variantes e conhecimento funcional. No centro deste universo de dados encontram-se algumas bases de dados essenciais que servem de referência para a comunidade científica mundial. Entre elas, destacam-se:

- **GenBank (Sayers et al., 2025)**: Mantido pelo National Center for Biotechnology Information (NCBI) nos EUA, o GenBank é o maior e mais antigo repositório de sequências nucleotídicas do mundo. Faz parte de uma colaboração internacional (INSDC) com o European Nucleotide Archive (ENA) e o DNA Data Bank of Japan (DDBJ). Contém dados de milhões de espécies, desde vírus até humanos, e é atualizado diariamente por investigadores de todo o mundo, sendo um pilar da ciência aberta e da reprodutibilidade.
- **Ensembl (Harrison et al., 2024)**: Focado principalmente em genomas de vertebrados e outros organismos modelo, o Ensembl é uma base de dados genómica que se destaca pela sua curadoria e anotação de alta qualidade. Para além de fornecer sequências genómicas, o Ensembl oferece anotações detalhadas de genes, transcritos, variantes genéticas e elementos regulatórios. Uma das ferramentas mais poderosas é o *Variant Effect Predictor (VEP),* que, à semelhança do SnpEff, regista variantes genéticas com base no vasto repositório de dados.
- **UCSC Genome Browser (Raney et al., 2024)**: Mais do que uma base de dados, o UCSC Genome Browser é uma plataforma de visualização interativa que permite aos investigadores explorar genomas de forma gráfica e intuitiva. A sua principal característica é o sistema de "*tracks*" (pistas), onde diferentes tipos de dados (genes, variantes, dados de expressão, níveis de conservação entre espécies) podem ser sobrepostos e visualizados no seu contexto genómico. Suporta o *upload* de dados personalizados, a transferência de anotações entre diferentes versões de genomas (liftOver) e a integração com APIs para acesso programático.
- **UniProt (UniProt Consortium, 2025)**: O UniProt é a principal e mais abrangente base de dados de proteínas. É o resultado de uma colaboração entre o Swiss Institute of Bioinformatics (SIB), o European Bioinformatics Institute (EBI) e o Protein Information Resource (PIR). Oferece sequências de proteínas com uma anotação funcional extremamente rica, incluindo função

molecular, processos biológicos, localização subcelular, domínios, modificações pós-traducionais e muito mais. A base de dados divide-se em duas secções: a **Swiss-Prot**, que é curada manualmente por especialistas e contém informação de alta qualidade, e a **TrEMBL**, que contém anotações automáticas de proteínas traduzidas a partir de sequências de nucleótidos no GenBank, oferecendo uma cobertura muito mais ampla.

2.1.6. Bibliotecas de Programação

Para além de ferramentas de linha de comando e interfaces *web*, a automação de análises complexas e a criação de fluxos de trabalho personalizados dependem de bibliotecas de programação. Entre as linguagens mais populares para esta finalidade, o Python destaca-se, em grande parte devido a um ecossistema de bibliotecas robustas e especializadas como o Biopython (Cock et al., 2009): no ecossistema da linguagem Python, o Biopython é a biblioteca de bioinformática mais fundamental e amplamente utilizada. Não se trata de uma única ferramenta, mas de um vasto conjunto de módulos que fornecem as "peças de construção" para a análise bioinformática. Oferece funcionalidades para a manipulação de sequências e os seus formatos (FASTA, GenBank), leitura e escrita de ficheiros de alinhamento, acesso programático a bases de dados *online* como as do NCBI (via Entrez), execução e *parsing* de resultados de ferramentas locais como o Blast, e manipulação de estruturas 3D de proteínas. A sua flexibilidade permite a criação de *pipelines* bioinformáticos totalmente personalizados, tornando-o um recurso indispensável para qualquer bioinformata.

2.2. Lacunas Identificadas na Integração de Ferramentas Bioinformáticas

Apesar da impressionante diversidade e sofisticação das ferramentas bioinformáticas atualmente disponíveis, persiste uma lacuna significativa no que diz respeito à integração funcional e à automação de ponta a ponta. A realidade diária de um bioinformático envolve frequentemente a utilização de múltiplas ferramentas de forma sequencial, onde o *output* de uma precisa ser formatado e processado antes de poder ser utilizado como *Input* para a seguinte. Este processo manual não só é ineficiente, mas também propenso a erros e dificulta a reprodutibilidade da análise.

A maioria dos *softwares* e bibliotecas existentes, como o Biopython, oferece módulos individuais para tarefas específicas (por exemplo, acesso ao NCBI ou manipulação de sequências), mas não proporciona uma solução unificada que permita a execução automatizada de fluxos de trabalho complexos de forma

nativa. O **Biopython**, por exemplo, embora seja uma biblioteca poderosa e fundamental, apresenta limitações notáveis na integração direta e moderna com ferramentas e bases de dados essenciais como ExPASy, Blast+, UniProt e DrugBank. Muitos dos seus módulos de acesso a recursos *online* estão desatualizados, foram descontinuados (como o *parser* para o Blast *web*) ou requerem configuração manual e a escrita de código adicional para funcionar corretamente com as APIs modernas, que evoluem constantemente.

Um exemplo crítico desta lacuna é a ausência de suporte nativo para acesso estruturado ao DrugBank, uma base de dados crucial que liga informação de alvos proteicos (genes) a dados farmacológicos (fármacos). Para investigadores em farmacogenómica e descoberta de fármacos, a capacidade de cruzar variantes genéticas com informação sobre a eficácia de medicamentos é vital, e a falta de uma interface de programação simples representa um obstáculo significativo.

Outro ponto crítico é a ausência de lógica analítica adicional pré-implementada que permita análises mais avançadas e interpretativas. Tarefas comuns como a identificação da maior ORF (Open Reading Frame) numa sequência de DNA, a deteção e anotação automatizada de variantes genéticas, ou a agregação de informação funcional de múltiplas fontes, não são triviais. Estas tarefas exigem frequentemente a combinação de várias ferramentas, o *parsing* de diferentes formatos de *output* e a implementação de lógica customizada, o que implica um esforço técnico elevado por parte do utilizador, desviando o foco da interpretação biológica para a engenharia de *software*.

3. Metodologia e Estrutura Conceptual

3.1. Introdução

A metodologia é um elemento fundamental em qualquer dissertação de mestrado, pois permite enquadrar cientificamente as etapas seguidas na investigação, justificando as opções tomadas e assegurando a validade dos resultados obtidos.

O presente trabalho insere-se no domínio da investigação aplicada em bioinformática, com uma forte componente de desenvolvimento de *software*. Nesse sentido, foi adotada a Metodologia de Investigação Construtiva (Constructive Research Approach – CRA) (Lukka, 2003), complementada por práticas de engenharia de *software*, nomeadamente um modelo de desenvolvimento em cascata modificado.

3.2. Metodologia de Investigação Construtiva

A Investigação Construtiva (CRM), proposta por Lukka (2003), é amplamente utilizada em áreas como gestão, engenharia e informática, quando o objetivo da investigação é criar e validar um artefato inovador que resolva um problema prático relevante.

As suas etapas principais foram aplicadas da seguinte forma neste projeto:

1. Identificação do problema prático
    - As ferramentas bioinformáticas mais usadas (ExPASy, Blast, UniProt, DrugBank) são executadas de forma manual e isolada, o que torna a análise de sequências genéticas morosa, fragmentada e sujeita a erros.
2. Compreensão teórica e revisão de literatura
    - Foi efetuada uma revisão do estado da arte em bibliotecas bioinformáticas, identificando limitações e lacunas em soluções como o Biopython.
3. Construção do artefacto
    - Desenvolvimento do módulo "*Found Sequence*", integrado no Biopython, que agrega de forma unificada as quatro ferramentas mencionadas.
    - Criação de aplicações *desktop* e *web* para demonstração da aplicabilidade.
4. Demonstração da funcionalidade
    - Testes técnicos com diferentes sequências em formato FASTA, ilustrados com fluxogramas e casos de uso.
5. Avaliação da solução
    - Medições de desempenho comparativas (solução integrada vs. ferramentas isoladas).

6. Reflexão teórica e contribuições
    - Discussão do impacto científico e prático da solução, limitações identificadas e possibilidades de extensão futura.
7. Generalização do conhecimento
    - A biblioteca proposta pode ser replicada, adaptada e expandida por outros investigadores, contribuindo para a comunidade científica da bioinformática.

3.3. Modelo de Desenvolvimento Adotado

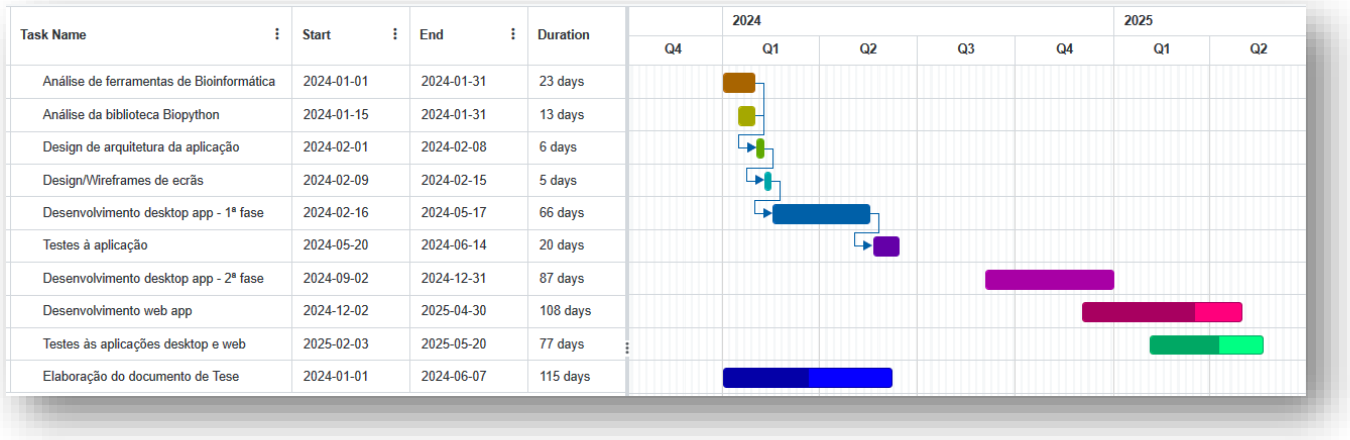
Embora a metodologia CRM oriente a investigação de forma global, o desenvolvimento técnico da solução seguiu práticas de engenharia de *software*, recorrendo a um modelo de desenvolvimento em cascata modificado.

O modelo tradicional em cascata caracteriza-se por uma sequência linear de fases (análise → *design* → implementação → testes → manutenção). No entanto, para mitigar a rigidez deste processo, foi adotada uma variante modificada, que permite retrocessos controlados entre fases, de modo a incorporar ajustes sempre que necessário.

As fases aplicadas foram as seguintes:

1. Análise de Requisitos – identificação das funcionalidades necessárias, avaliação da biblioteca Biopython e análise das ferramentas bioinformáticas a integrar.
2. *Design* – definição da arquitetura, fluxogramas e *wireframes* das interfaces.
3. Implementação – desenvolvimento do submódulo "*Found Sequence*" e das aplicações *desktop* e *web*.
4. Testes – verificação de funcionalidade e integração através de casos de uso, validação preliminar com utilizadores.
5. Documentação e Avaliação – elaboração da dissertação e análise crítica dos resultados obtidos.

As tarefas de todo o projeto foram planeadas conforme ilustra a Figura 1:



Task Name	Start	End	Duration
Análise de ferramentas de Bioinformática	2024-01-01	2024-01-31	23 days
Análise da biblioteca Biopython	2024-01-15	2024-01-31	13 days
Design de arquitetura da aplicação	2024-02-01	2024-02-08	6 days
Design/Wireframes de ecrãs	2024-02-09	2024-02-15	5 days
Desenvolvimento desktop app - 1ª fase	2024-02-16	2024-05-17	66 days
Testes à aplicação	2024-05-20	2024-06-14	20 days
Desenvolvimento desktop app - 2ª fase	2024-09-02	2024-12-31	87 days
Desenvolvimento web app	2024-12-02	2025-04-30	108 days
Testes às aplicações desktop e web	2025-02-03	2025-05-20	77 days
Elaboração do documento de Tese	2024-01-01	2024-06-07	115 days

Figura 1 – Planeamento de tarefas macro associadas ao projeto

3.4. Justificação da Metodologia

A opção pela metodologia CRM justifica-se por:

- Permitir criar um artefato novo (biblioteca bioinformática integrada) em resposta a um problema prático real.
- Conciliar rigor científico (através da revisão de literatura e formulação de perguntas de investigação) com utilidade prática (ferramenta testada em casos reais).
- Favorecer a transferibilidade dos resultados, possibilitando que outros investigadores utilizem e expandam a solução.

Por sua vez, o modelo em cascata modificado foi escolhido por ser um processo estruturado e sequencial, adequado para projetos académicos com tempo limitado, mas que incorpora a flexibilidade necessária para ajustar etapas consoante os resultados obtidos durante o desenvolvimento.

3.5. Síntese

A combinação da Investigação Construtiva com um modelo de desenvolvimento em cascata modificado assegura que este trabalho cumpre uma dupla exigência:

- Académica, pela formulação de problema, hipótese e perguntas de investigação.
- Prática, pela entrega de um artefato funcional validado com utilizadores e especialistas.

Assim, a metodologia aplicada garante a robustez científica e a relevância prática da dissertação.

4. Desenvolvimento

4.1. Análise

Nesta secção inicial de desenvolvimento, procedeu-se a uma análise detalhada das ferramentas e tecnologias que servem de base a este projeto. O objetivo foi avaliar as capacidades e limitações das soluções existentes para, a partir daí, definir os requisitos e a estratégia de desenvolvimento da nova biblioteca. Esta análise dividiu-se em duas frentes: uma investigação à biblioteca Biopython, para identificar que funcionalidades poderiam ser aproveitadas ou melhoradas, e uma análise aprofundada a cada uma das ferramentas bioinformáticas selecionadas (ExPASy, NCBI Blast+, UniProt e DrugBank), para compreender os seus dados, formatos de *output* e possibilidades de integração.

4.1.1. Biblioteca Biopython

Foi realizada uma análise da biblioteca Biopython com o objetivo de determinar a sua aplicabilidade para interagir com as ferramentas bioinformáticas consideradas relevantes para este projeto: ExPASy, UniProt, NCBI Blast e DrugBank. A avaliação focou-se em dois aspetos principais:

1. A existência de módulos específicos na biblioteca Biopython para cada uma destas ferramentas.
2. A adequação das funcionalidades oferecidas e dos dados retornados por esses módulos face aos requisitos do projeto.

4.1.1.1 Resultados da Análise

- ExPASy e UniProt: Verificou-se que a biblioteca possui módulos relacionados, nomeadamente Bio.ExPASy e Bio.SwissProt (para lidar com dados do UniProt). Contudo, uma análise mais detalhada revelou que as funções específicas disponíveis nestes módulos ou os dados por eles retornados não correspondiam exatamente às necessidades de informação e interação pretendidas para o fluxo de trabalho do projeto.
- NCBI Blast: O módulo tradicionalmente utilizado para consultas Blast *online*, Bio.Blast.NCBIWWW, encontra-se atualmente obsoleto. A alternativa recomendada pela NCBI é o Elastic Blast (Camacho, Christiam, Grzegorz M. Boratyn, Victor Joukov, Roberto Vera Alvarez,

and Thomas L. Madden, 2023), um serviço baseado na nuvem que, no entanto, acarreta custos associados à sua utilização, tornando-o inviável no contexto deste projeto.

- DrugBank: Não foi identificado qualquer módulo na biblioteca Biopython que ofereça suporte direto para interagir com a base de dados DrugBank.

4.1.1.2 Conclusão e Próximos Passos

Dado que a utilização direta dos módulos da biblioteca Biopython apresentou limitações significativas – quer por funcionalidades inadequadas quer por custos associados –, a estratégia de integração foi redirecionada. A análise focou-se na investigação da existência e da acessibilidade de utilização direta das APIs públicas (Interfaces de Programação de Aplicações) fornecidas por cada uma das ferramentas individualmente (ExPASy, UniProt, NCBI Blast e DrugBank).

4.1.2. Ferramentas Bioinformáticas

Conforme justificado na secção de motivação deste documento, a seleção das ferramentas baseou-se na necessidade de interligação funcional e na relevância do conteúdo abordado na unidade curricular Análise Computacional de Genomas e Proteomas. Deste modo, as ferramentas escolhidas para melhor se adequarem a estes requisitos foram: ExPASy Translate Tool, NCBI Blast+, UniProt e DrugBank.
A integração destas ferramentas segue um fluxo de trabalho sequencial específico:

1. Tradução da Sequência: Inicialmente, utiliza-se o ExPASy Translate Tool para traduzir uma sequência de nucleótidos fornecida, gerando as possíveis sequências de aminoácidos correspondentes.
2. Identificação da maior ORF: A partir dos resultados da tradução, identifica-se a maior *Open Reading Frame* (ORF), que representa a sequência codificante mais provável.
3. Análise de Similaridade: A sequência de aminoácidos da ORF identificada é submetida ao NCBI Blast+ para avaliar a sua similaridade com sequências proteicas de referência que constam nas bases de dados públicas.
4. Deteção de Variantes: Analisam-se os resultados do Blast+ para detetar possíveis variantes (substituições de aminoácidos) e *gaps* (inserções/deleções) na sequência em estudo, registando a sua posição exata em relação à sequência de referência.

5. Associação a Patologias: Caso sejam detetadas variantes, consulta-se a base de dados UniProt para verificar se essas alterações, na posição correspondente, estão documentadas e associadas a alguma patologia ou condição clínica.
6. Identificação de Fármacos: Se for encontrada uma associação a uma doença no UniProt, recorre-se à base de dados DrugBank para procurar fármacos conhecidos que possam ser relevantes para o tratamento ou mitigação dessa patologia específica.

De seguida será descrito o tipo de análise feita para cada ferramenta que irá integrar este projeto.

4.1.2.1 ExPASy Translate Tool – Análise e Identificação de Necessidades

A Tabela 2 mostra todos os pontos relevantes para a análise da ferramenta ExPASy e resultados. A partir dos resultados são identificadas as necessidades de processamento adicional.

Tabela 2 - Pontos de análise e resultados - ExPASy Translate Tool

Pontos de Análise	Resultados
Verificação da disponibilidade de acesso programático (API ou similar)	Confirmou-se que é possível obter os resultados da tradução de forma programática (por exemplo, através de URLs específicas ou *scripting*), funcionando efetivamente como uma interface de programação (API), embora possa não ser uma API REST formalmente documentada.
Identificação do tipo de dados retornados pela ferramenta	A ferramenta retorna as sequências de aminoácidos resultantes da tradução da sequência de ácido nucleico de entrada, considerando os seis quadros de leitura (*reading frames*) possíveis.
Determinação dos formatos de *output* suportados	Os formatos de *output* disponíveis incluem texto simples (.txt) e, mais importante para integração bioinformática, o formato FASTA.
Avaliação da complexidade e necessidade do tratamento dos dados de *output*	Para este projeto, selecionou-se o formato FASTA. A estrutura do ficheiro resultante segue o padrão FASTA, onde cada quadro de leitura é apresentado sequencialmente.

Necessidades

Após a obtenção dos dados do ExPASy Translate Tool, foram identificadas duas necessidades principais de processamento adicional:

- *Parsing* do Formato FASTA: O *output* da ferramenta, embora estruturado, contém quebras de linha (\n) dentro das sequências de aminoácidos (como no exemplo abaixo):

  VIRT-1382238:3'5' Frame 1\nFFFFFFHFSTRFIISKKVYYSADIIPKTFYRTISSLPAQQTFIQFPYFNV

  É fundamental desenvolver um algoritmo de *parsing* que leia este ficheiro, identifique corretamente os cabeçalhos (>) e concatene todas as linhas de sequência correspondentes a cada *frame*, removendo as quebras de linha internas, para obter as seis sequências de aminoácidos completas e lineares.
- Identificação da *Open Reading Frame* (ORF) mais longa: A ferramenta **não** identifica as *Open Reading Frames* (ORFs) biologicamente relevantes dentro dessas traduções (ou seja, os segmentos contínuos que começam com um codão de iniciação – geralmente Metionina 'M' – e terminam num codão de paragem – representado por '*' ou similar). É necessário implementar um algoritmo que analise cada uma das seis sequências de aminoácidos obtidas após o *parsing*. Este algoritmo deve procurar por todas as ORFs possíveis (segmentos M...*) em cada *frame* e identificar a ORF mais longa encontrada entre todos os *frames*. Esta ORF mais longa é geralmente considerada a candidata mais provável a representar a sequência codificante funcional.

4.1.2.2 NCBI Blast+ – Análise e Necessidades

A Tabela 3 mostra todos os pontos relevantes para a análise da ferramenta NCBI Blast+ e resultados. A partir dos resultados são identificadas as necessidades de processamento adicional.

Tabela 3 – Pontos de análise e resultados – NCBI Blast+

Pontos de Análise	Resultados
Verificação da disponibilidade de acesso programático (API ou similar)	Confirmou-se que é possível obter os resultados da tradução de forma programática (por exemplo, através de URLs específicas ou *scripting*), funcionando efetivamente como uma interface de programação

	(API). Esta API encontra-se documentada e pode ser consultada no seguinte URL: NCBIBLAST+HelpandDocumentation-RESTAPI
Identificação do tipo de dados retornados pela ferramenta	A ferramenta apresenta vários resultados com percentagem de similaridade para várias espécies. Dentro destes resultados é encontrada variada informação sendo que a mais importante para o projeto é a seguinte: • Tipo (proteína, DNA), • Tamanho da sequência, • Definição ou identificação da sequência, • *Hit accession* para aceder ao UniProt, • Descrição da sequência, • Espécie (ex.: Homo sapiens), • Sequência pesquisada, • Sequência de referência.
Determinação dos formatos de *output* suportados	Os formatos de *output* disponíveis incluem XML ou JSON.
Avaliação da complexidade e necessidade do tratamento dos dados de *output*	Para este projeto, selecionou-se o formato JSON. A partir daqui será necessário fazer a leitura de ficheiro, alinhamento de sequências e obter variantes, caso existam. Para isto foram levantadas algumas necessidades detalhadas no ponto seguinte.

Necessidades

Após a obtenção dos dados do NCBI Blast+, foram identificadas três necessidades principais de processamento adicional:

- Leitura do ficheiro JSON: É fundamental desenvolver um algoritmo para leitura do ficheiro JSON e recolha da informação relevante para apresentação.
- Alinhamento de sequências: as sequências de pesquisa e de referência não vêm alinhadas em diferentes linhas, por isso é necessário desenvolver um algoritmo que possa alinhar as sequências em várias linhas de forma a facilitar a sua leitura.

- Identificação de variantes: não são identificadas as diferenças entre a sequência pesquisada e a de referência, assim, é necessário desenvolver um algoritmo que faça esta comparação e identifique a variante e a respetiva posição.

4.1.2.3 UniProt – Análise e Necessidades

A Tabela 4 mostra todos os pontos relevantes para a análise da ferramenta UniProt e resultados. A partir dos resultados são identificadas as necessidades de processamento adicional.

Tabela 4 – Pontos de análise e resultados – UniProt

Pontos de Análise	Resultados
Verificação da disponibilidade de acesso programático (API ou similar)	Confirmou-se que é possível obter os resultados da tradução de forma programática (por exemplo, através de URLs específicas ou *scripting*), funcionando efetivamente como uma interface de programação (API). Esta API encontra-se documentada e pode ser consultada no seguinte URL: api/doc/
Identificação do tipo de dados retornados pela ferramenta	A ferramenta apresenta um único resultado, sendo a pesquisa feita diretamente pelo seu *Hit Accession*. Ex.: P00441. Existe uma quantidade imensa de informação retornada, da qual a mais importante para o projeto é: • Nome científico, • Nome comum, • Id, • Função de proteína, • Atividade catalítica, • Lista de variantes, posição e doenças associadas.
Determinação dos formatos de *output* suportados	O formato de *output* disponível é em ficheiro JSON.

Avaliação da necessidade e complexidade do tratamento dos dados de *output*	Sendo o formato JSON o utilizado, será necessário desenvolver algo para leitura do mesmo, o que será descrito no ponto seguinte.

Necessidades

Após a obtenção dos dados do UniProt doi identificada uma necessidade principal de processamento adicional:

- Leitura do ficheiro JSON: É fundamental desenvolver um algoritmo para leitura do ficheiro JSON e recolha da informação relevante para apresentação.

4.1.2.4 DrugBank – Análise e Necessidades

A Tabela 5 mostra todos os pontos relevantes para a análise da ferramenta DrugBank e resultados. A partir dos resultados são identificadas as necessidades de transformação da informação possível.

Tabela 5 – Pontos de análise e resultados – DrugBank

Pontos de Análise	Resultados
Verificação da disponibilidade de acesso programático (API ou similar)	Acesso Programático: confirmou-se que é possível obter os resultados da tradução de forma programática, contudo, a API não é pública e para obtenção de acesso existem custos. No entanto, a DrugBank fornece um ficheiro XML que é uma licença de estudantes/investigadores. Para obter este ficheiro foi necessário fazer o registo na DrugBank e explicar qual seria o caso de uso.

Necessidades

Após obter o XML com todos os dados, concluiu-se que a sua leitura ia ser difícil e massiva uma vez que o XML contém uma quantidade considerável de informação. Assim a solução adotada foi a seguinte:

- Criação de base de dados: Após análise do XML, foi criada uma base de dados com diversas tabelas e respetivas relações para os dados necessários à aplicação.

- Criação de algoritmo de leitura de XML e migração de dados necessários para as tabelas, para que seja mais fácil a consulta de dados.

O desenvolvimento neste projeto divide-se em três partes:

1. *Fork* do projeto Biopython e desenvolvimento das ferramentas necessárias;
2. Aplicação *desktop* que serve de interface com o utilizador
3. Aplicação *Web* que serve também de interface com o utilizador, mas é também uma forma de demonstração de que a biblioteca pode e deve ser "servida" para ser acedida qualquer que seja o tipo de aplicação.

O desenvolvimento teve por base algumas ferramentas essenciais:

- Interpretador Python: O projeto foi desenvolvido utilizando a versão [v3.11.19] do Python, instalada no sistema.
- GIT: O sistema de controlo de versões GIT foi instalado para permitir o versionamento do código e a gestão do '*fork*' da biblioteca Biopython.
- IDE: O principal Ambiente de Desenvolvimento Integrado (IDE) utilizado foi o Visual Studio Code da Microsoft, configurado para o desenvolvimento em Python.
- Frameworks para desenvolvimento das aplicações, *desktop* e *web*.

De seguida serão desenvolvidos os seguintes tópicos que integram o desenvolvimento:
Arquitetura das aplicações, fluxogramas das funcionalidades, *wireframes* com esboços de estrutura e organização de ecrãs, detalhe de cada uma das aplicações bem como o trabalho feito no '*fork*' do Biopython dando enfoque ao que foi desenvolvido e o porquê da existência de duas aplicações para interação com a biblioteca, detalhes das *frameworks* usadas e algumas imagens finais dos ecrãs/páginas e a sua interação com a biblioteca Biopython.

4.2. Arquitetura

A Figura 2 esquematiza a arquitetura geral da solução desenvolvida. O componente central é a 'Nova biblioteca Biopython', que representa a extensão ao projeto Biopython existente, criada através de um *fork* (uma cópia para desenvolvimento independente). Dentro desta, reside o módulo "*Found Sequence*" que é a peça principal do desenvolvimento, que contém a lógica para interagir com as diversas ferramentas."

O fluxo de informação segue os seguintes passos:

1. Interação com o Utilizador: O utilizador interage com o sistema através de uma interface (aplicação *Desktop* ou *Web*) fornecendo os dados de entrada (como uma sequência genética).
2. Comunicação via "*Found Sequence*": A interface envia o pedido para o módulo "*Found Sequence*" dentro da biblioteca Biopython.
3. Acesso às Ferramentas Externas:
    - Para ExPASy, Blast e UniProt, o "*Found Sequence*" utiliza as APIs públicas destas ferramentas. Envia os pedidos necessários pela Internet e recebe os resultados (ex.: tradução da sequência, resultados de similaridade, informação da proteína). Esta comunicação é bilateral.
    - Para o DrugBank, devido às restrições de acesso à API, optou-se por uma abordagem diferente: o "*Found Sequence*" consulta uma base de dados local que contém a informação sobre fármacos.
4. Apresentação de Resultados: O "*Found Sequence*" agrega toda a informação recolhida e devolve-a à Interface do Utilizador, que a apresenta de forma organizada. O sistema permite também gerar um relatório consolidado em PDF.

Esta arquitetura permite modularidade (cada ferramenta é um componente) e facilita a integração de dados de múltiplas fontes num único fluxo de trabalho acessível ao utilizador.

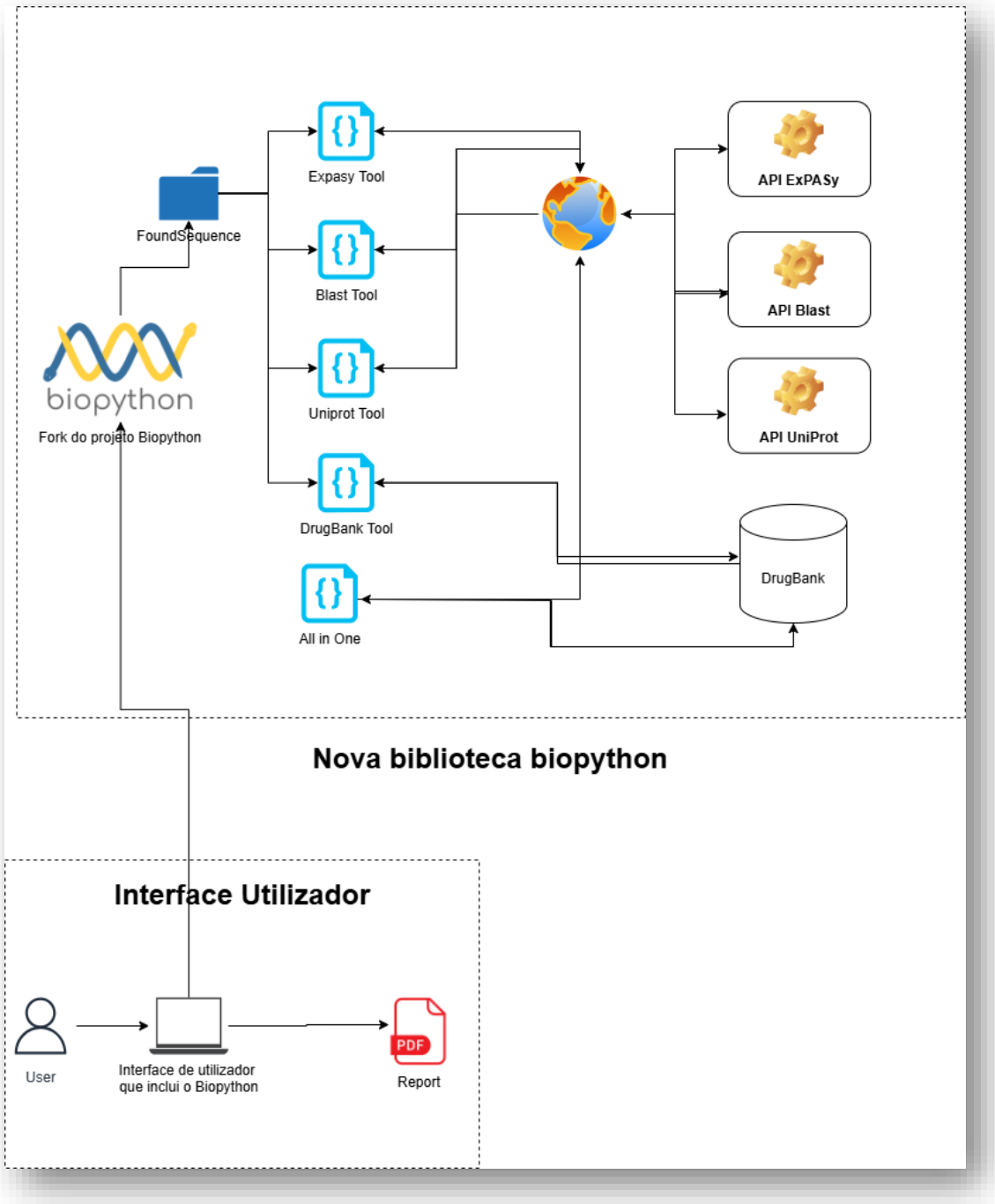


Figura 2 – Arquitetura da aplicação

4.3. Fluxogramas

Cada uma das funcionalidades desenvolvidas pode ser usada individualmente, isto é, apenas fazer a tradução para proteína de uma sequência genética, ou através de uma sequência genética fazer a comparação com uma sequência de referência no Blast, saber mais acerca de uma proteína através do identificador no UniProt ou saber mais acerca de fármacos para uma determinada patologia. Contudo, e como maior propósito do projeto pode juntar-se todas estas ferramentas num só pedido e obter toda a informação sobre uma sequência. De seguida são apresentados os fluxogramas de cada uma destas funcionalidades.

4.3.1. ExPASy

O seguinte fluxograma ilustrado na Figura 3 representa o processo de obtenção da tradução de uma sequência genética na sua proteína. Principais elementos do fluxograma:

1. Entrada de dados: Ficheiro em formato .FASTA.
2. Decisão: verificar se o ficheiro é válido.
3. Saídas possíveis: Efetuar tradução ou exibir erro se o ficheiro for inválido no seu formato/extensão ou se estiver vazio.
4. Decisão: Verificar se a tradução foi possível e com sucesso.
5. Saídas possíveis: Apresentar resultados e possibilidade de gerar um relatório com os mesmos ou exibir mensagem de não obtenção de resultados.

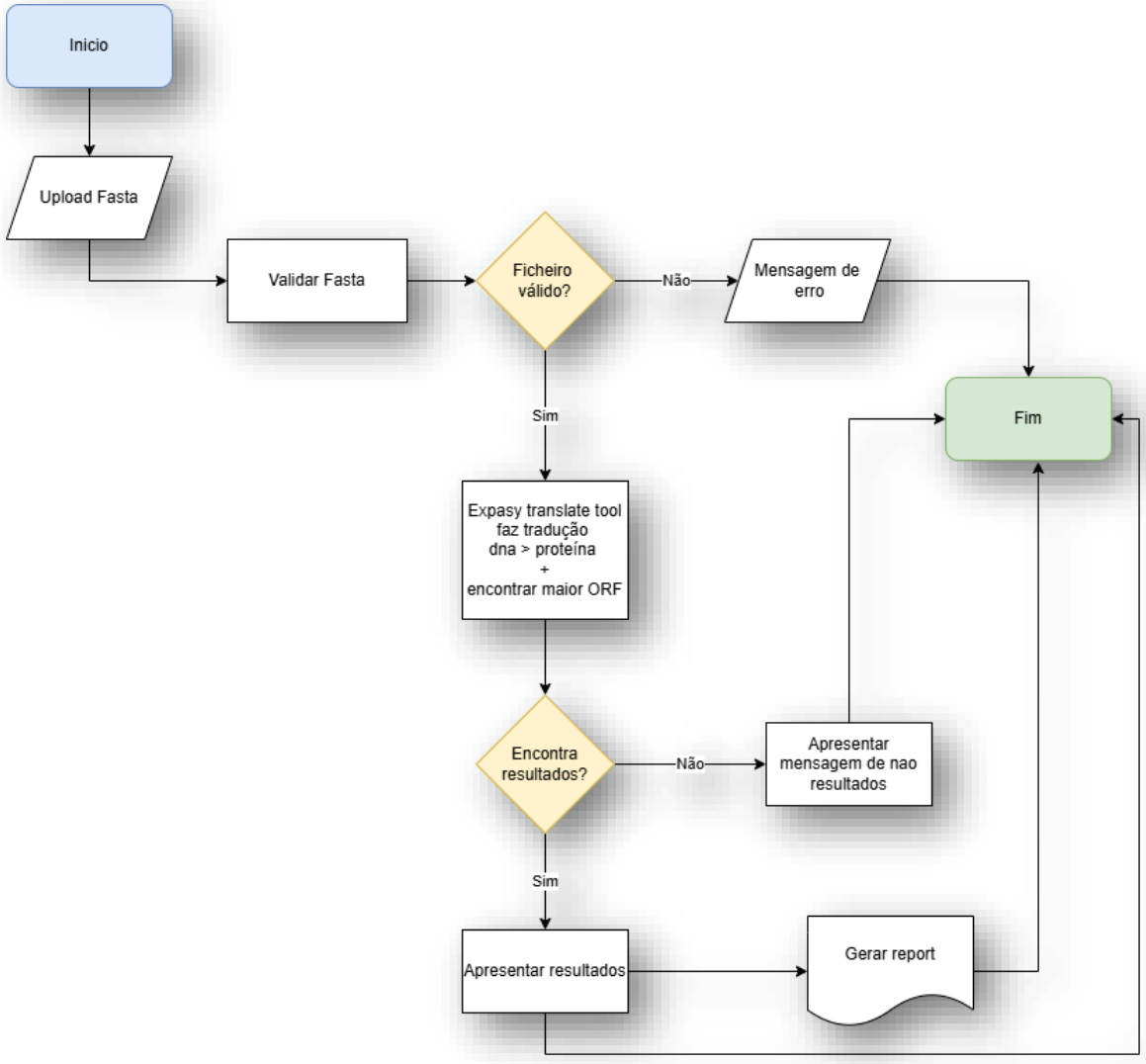


Figura 3 - Fluxograma para a funcionalidade com recurso ao ExPASy Translate Tool

4.3.2. NCBI Blast +

O seguinte fluxograma ilustrado na Figura 4 representa o processo de análise de similaridade entre uma sequência de DNA/proteína e uma sequência de referência.

Principais elementos do fluxograma:

1. Entrada de dados: Sequência DNA/proteína.
2. Decisão: verificar se a sequência é válida.
3. Saídas possíveis: Fazer análise de similaridade, alinhar sequência e encontrar possíveis variantes ou exibir erro se a sequência for inválida.
4. Decisão: Verificar se resultados foram encontrados com sucesso.

5. Saídas possíveis: Apresentar resultados e possibilidade de gerar um relatório com os mesmos ou exibir mensagem de não obtenção de resultados.

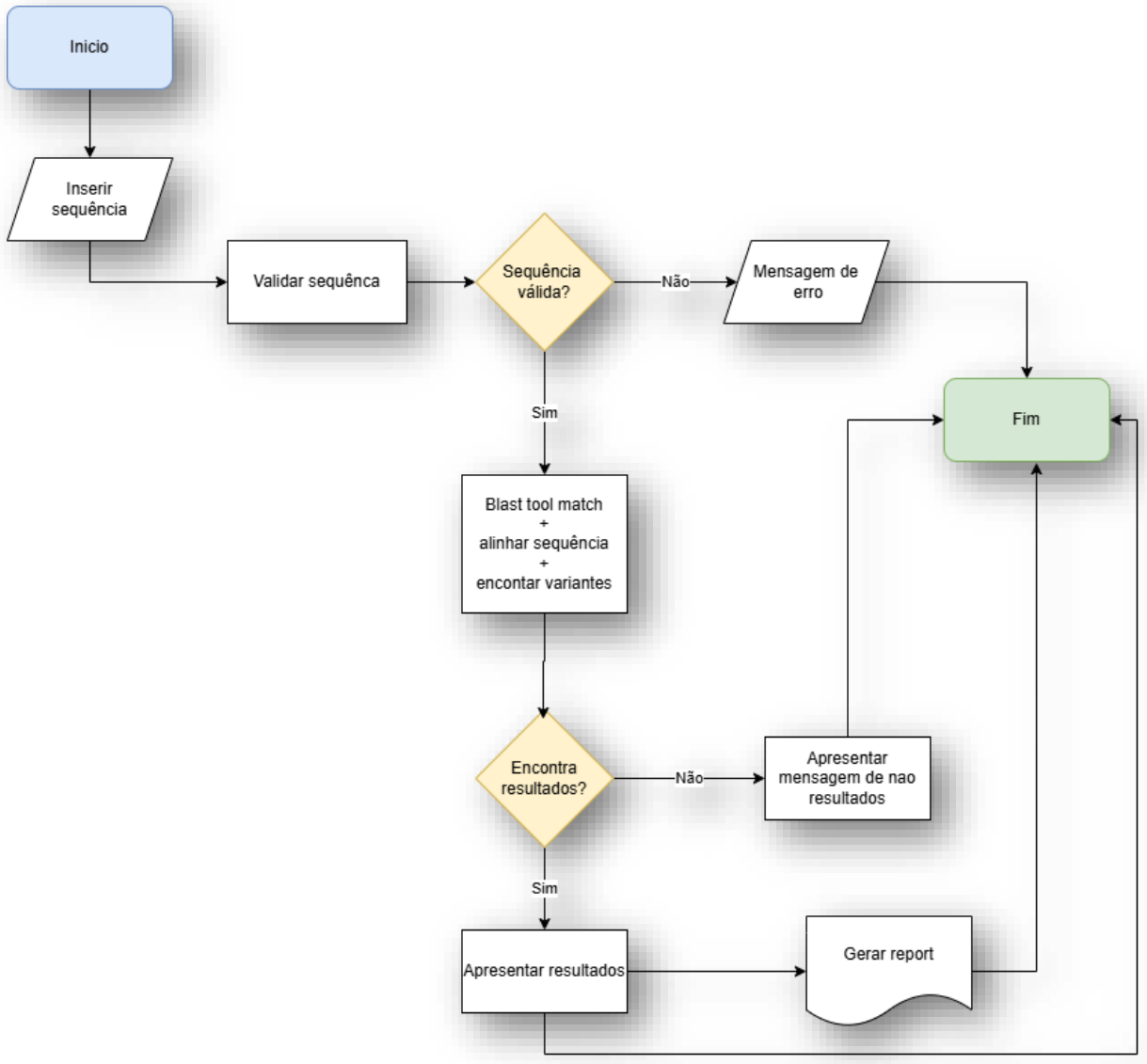


Figura 4 - Fluxograma para a funcionalidade com recurso ao NCBI Blast+

4.3.3. UniProt

O seguinte fluxograma ilustrado na Figura 5 representa o processo de obtenção de informação sobre uma proteína a partir do seu identificador.

Principais elementos do fluxograma:

1. Entrada de dados: Inserir identificador de proteína (*hit accession*).

2. Decisão: Verificar se resultados foram encontrados com sucesso.
3. Saídas possíveis: Apresentar resultados e possibilidade de gerar um relatório com os mesmos ou exibir mensagem de não obtenção de resultados.

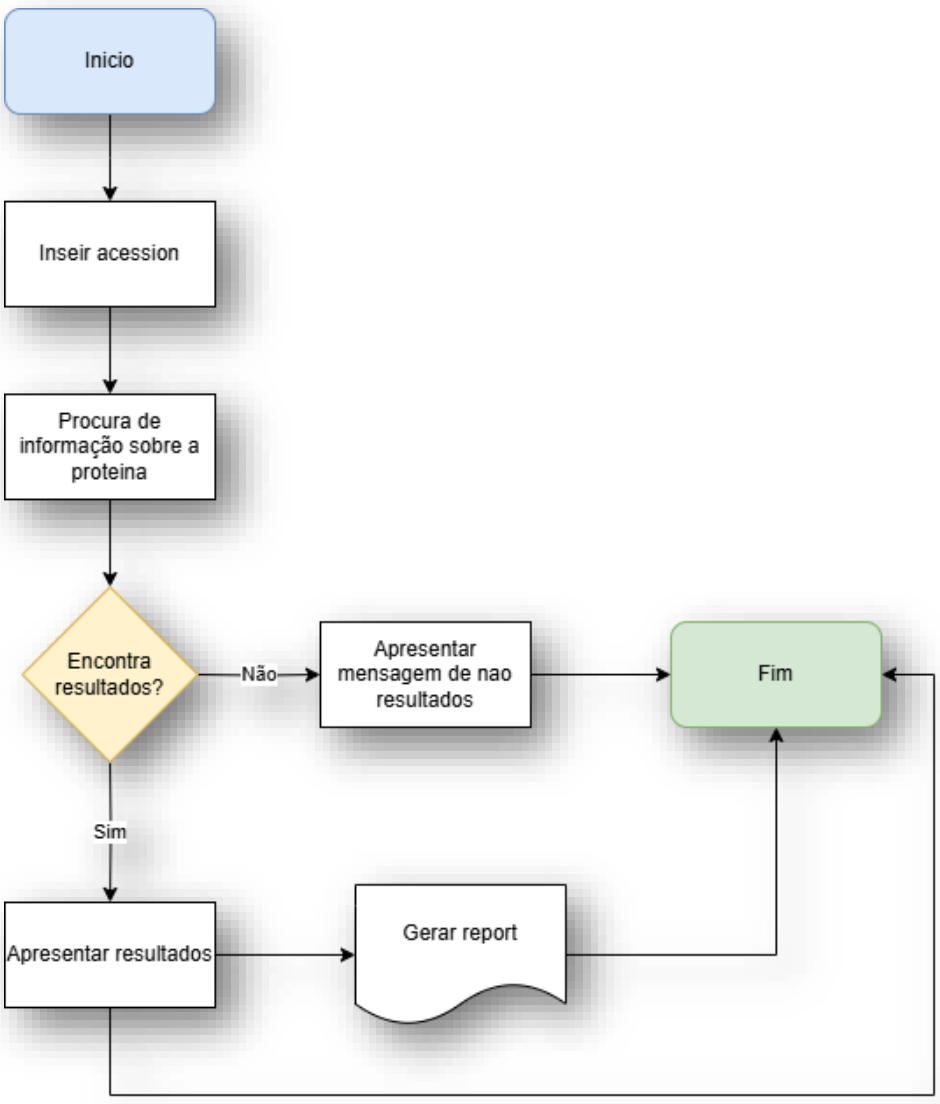


Figura 5 – Fluxograma para a funcionalidade com recurso ao UniProt

4.3.4. DrugBank

O seguinte fluxograma ilustrado na Figura 6 representa o processo de obtenção de fármacos e informação sobre os mesmos para uma determinada patologia.

Principais elementos do fluxograma:

1. Entrada de dados: Inserir patologia.
2. Decisão: Verificar se resultados foram encontrados com sucesso.
3. Saídas possíveis: Apresentar resultados e possibilidade de gerar um relatório com os mesmos ou exibir mensagem de não obtenção de resultados.

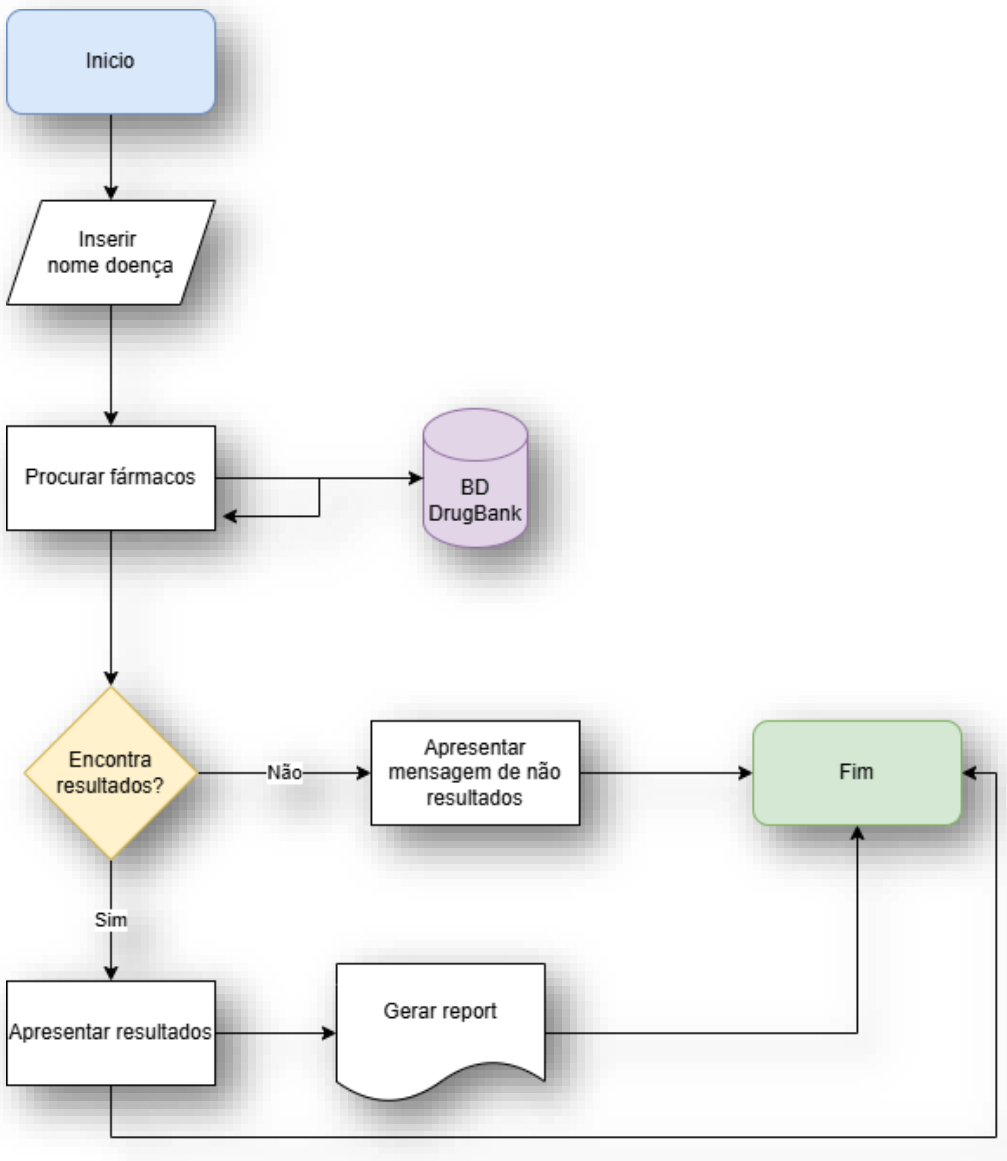


Figura 6 – Fluxograma para a funcionalidade com recurso aos dados Drugbank

4.3.5. *All in One*

O seguinte fluxograma ilustrado na Figura 7 representa o processo de junção de todas as ferramentas para obter um só resultado

Principais elementos do fluxograma:

1. Entrada de dados: Ficheiro em formato .FASTA.
2. Decisão: verificar se o ficheiro é válido.
3. Execução de tarefa: Efetuar tradução ou exibir erro se o ficheiro for inválido no seu formato/extensão ou se estiver vazio.
4. Decisão: Verificar se a tradução foi possível e com sucesso.
6. Execução de tarefa:
    - Exibir mensagem de não obtenção de resultados e terminar aqui a execução da aplicação.
    - Ou enviar sequência da proteína para o próximo serviço.
5. Decisão: Verificar se a sequência é válida.
6. Execução de tarefa: Fazer análise de similaridade, alinhar sequência e encontrar possíveis variantes ou exibir erro se a sequência for inválida.
7. Decisão: Verificar se resultados foram encontrados com sucesso.
8. Execução de tarefa:
    - Apresentar mensagem de não obtenção de resultados e terminar aqui a execução da aplicação.
    - Ou se existirem resultados, mas sem variantes encontradas, estes são apresentados e termina também a execução, sendo possível gerar aqui o relatório.
    - Ou se existem variantes, prossegue para o UniProt com o seu identificador e procure doenças associadas às variantes encontradas.
9. Decisão: Verificar se resultados foram encontrados com sucesso.
10. Saídas possíveis:
    1.1 Exibir mensagem de não obtenção de resultados.
    2.1 Ou então apresentar resultados caso não tenham sido encontradas doenças para as variantes.
    3.1 Ou se foram encontradas doenças, prosseguir para o DrugBank com a pesquisa de fármacos para as doenças.
11. Decisão: Verificar se resultados foram encontrados com sucesso.

12. Saídas possíveis: Apresentar resultados e possibilidade de gerar um relatório mesmo que não tenham sido encontrados fármacos ou então apresentar resultados completos com os dados de todos os serviços.

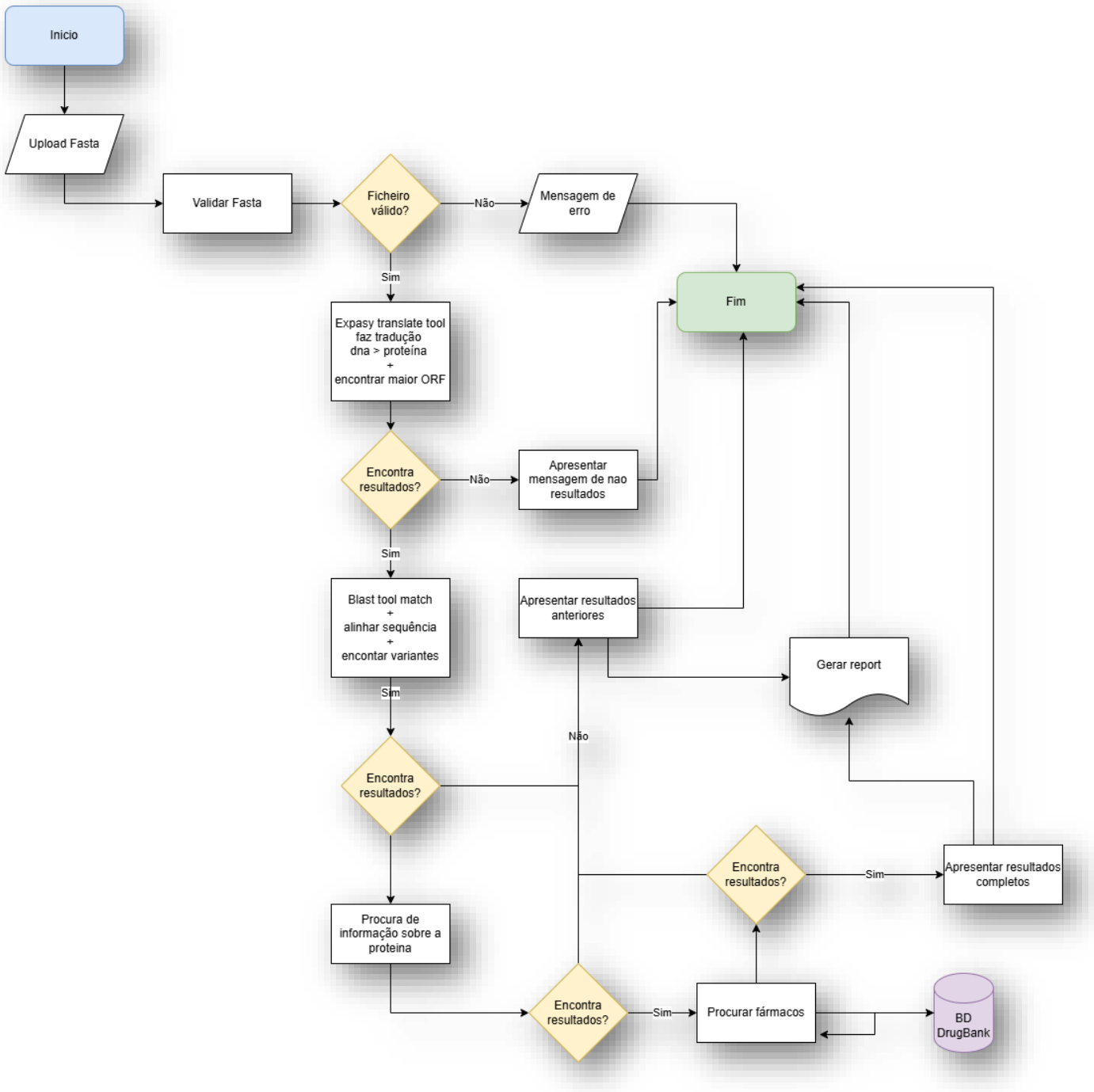


Figura 7 – Fluxograma para a funcionalidade com recurso a todos os serviços

4.4. *Wireframes*

Por forma a ajudar no desenvolvimento da aplicação e tendo em conta a experiência do utilizador e a interface, foram desenhados alguns esboços de estrutura e organização de ecrãs. O seu resultado final pode não ficar exatamente como o desenhado, uma vez que os *wireframes* são usados em UX/UI *Design* para validação de ideias antes de iniciar o desenvolvimento real.

Os *wireframes* aqui apresentados foram desenvolvidos com recurso à ferramenta Figma e são orientados à aplicação *desktop*.

4.4.1. Ecrã de entrada

O *wireframe* do ecrã de entrada, ou *landing page, ilustrado na Figura 8*, estabelece a navegação principal da aplicação. Apresenta o nome e o logótipo da aplicação no topo e cinco secções interativas principais. Cada uma das quatro primeiras secções corresponde a uma ferramenta individual (ExPASy, Blast+, UniProt, DrugBank), incluindo um resumo da sua funcionalidade. A quinta secção é dedicada à funcionalidade "*All in One*", que integra todos os serviços. Este *design* permite que o utilizador escolha entre realizar uma análise específica e executar o fluxo de trabalho completo a partir de um único ponto de partida.

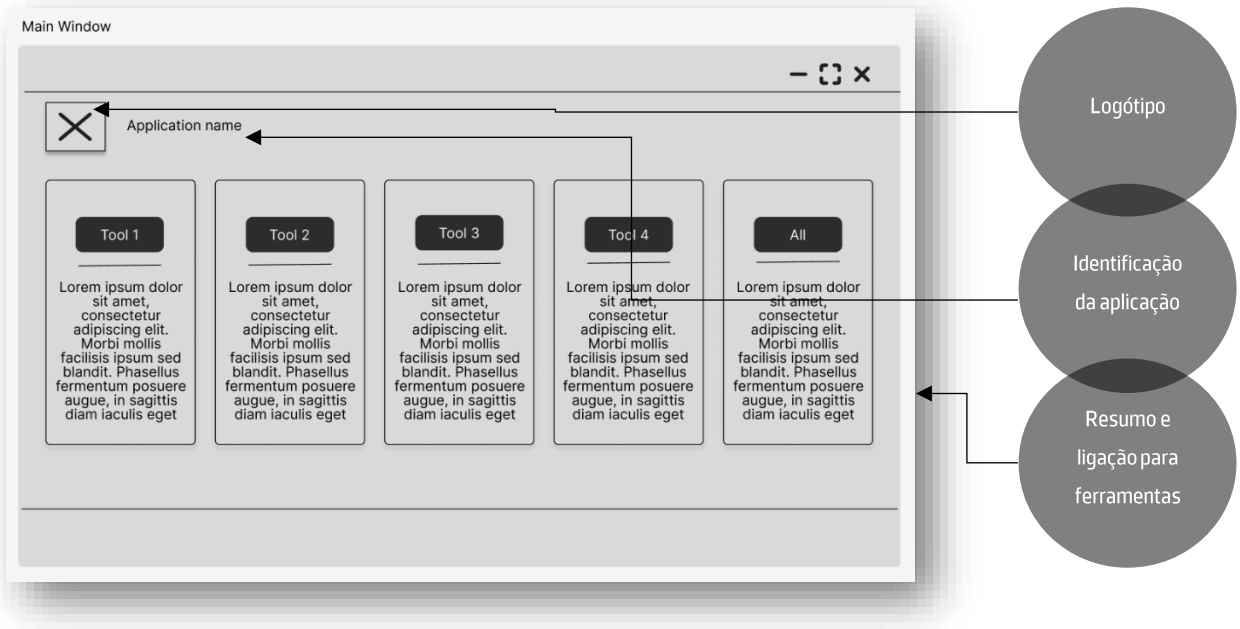


Figura 8 – Wireframe para ecrã de entrada ou landing page

4.4.2. Ecrã para tradução de proteína (ExPASy Translate Tool)

Este ecrã foi projetado para a funcionalidade de tradução de sequências de ADN em proteínas. O *design* ilustrado na Figura 9 inclui três componentes principais: uma área de *upload* de ficheiros (*file uploader*), onde o utilizador pode carregar a sua sequência em formato FASTA, botões de ação para iniciar a tradução e para gerar um relatório dos resultados, e uma área de resultados para visualizar as sequências de aminoácidos traduzidas e a maior ORF identificada.

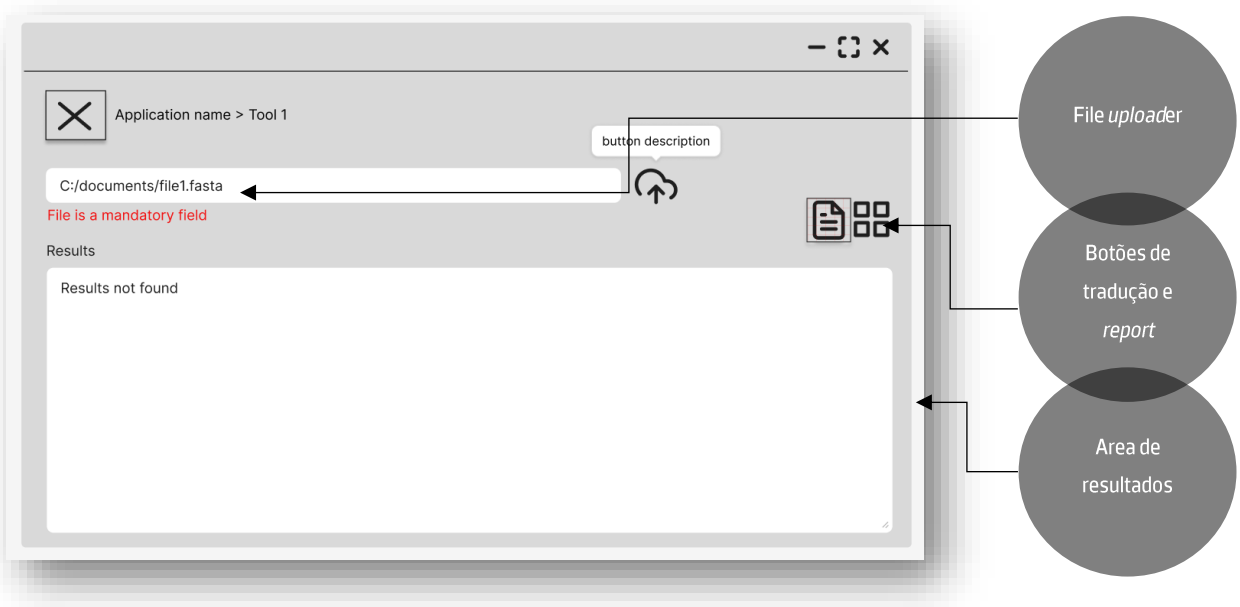


Figura 9 – Wireframe para ecrã/página para tradução DNA > Proteína

4.4.3. Ecrã para análise de similaridade (NCBI Blast +)

O *wireframe* ilustrado na Figura 10 idealizado para a ferramenta de análise de similaridade define uma interface focada na comparação de sequências. A estrutura é composta por uma área de entrada de dados, onde o utilizador pode inserir a sequência a ser analisada; botões dedicados a iniciar a análise e a exportar um relatório ; e uma área de resultados, projetada para apresentar a informação da sequência de referência, o alinhamento e as variantes encontradas.

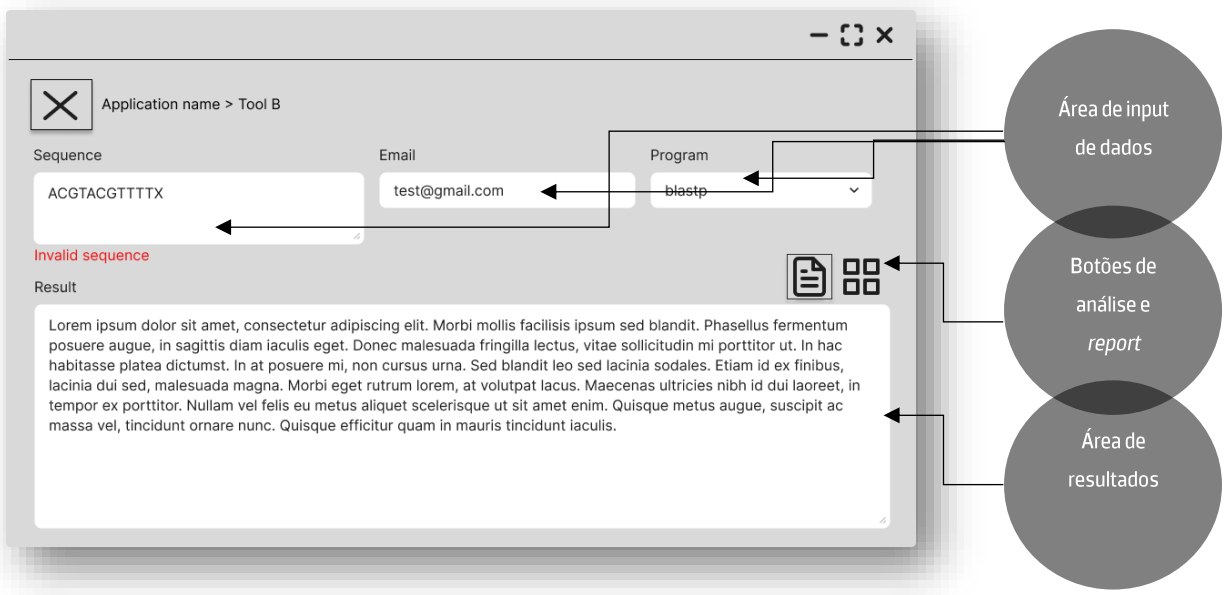


Figura 10 – Wireframe para ecrã/página para pesquisa de similaridade

4.4.4. Ecrã para análise de informação de proteína (UniProt)

O ecrã representado na Figura 11 foi desenhado para a consulta de informações detalhadas sobre proteínas na base de dados UniProt. A interface inclui uma área de entrada de dados para que o utilizador insira o identificador da proteína (*hit accession*). Abaixo, encontram-se os botões de análise e de geração de relatório. A maior parte do ecrã é dedicada à área de resultados, onde serão exibidas informações como a função da proteína, variantes e patologias associadas.

Figura 11 – Wireframe para ecrã/página de pesquisa de informação de proteína

4.4.5. Ecrã para procura de fármacos (DrugBank)

O *wireframe* representado na Figura 12, idealizado para a funcionalidade de pesquisa de fármacos, é direcionado para a consulta na base de dados DrugBank. O *layout* apresenta uma área para a inserção de dados, onde o utilizador deve indicar a patologia de interesse. Assim como nos outros ecrãs, existem botões para executar a pesquisa e gerar um relatório. A área de resultados foi concebida para listar os fármacos relevantes encontrados, juntamente com as suas informações.

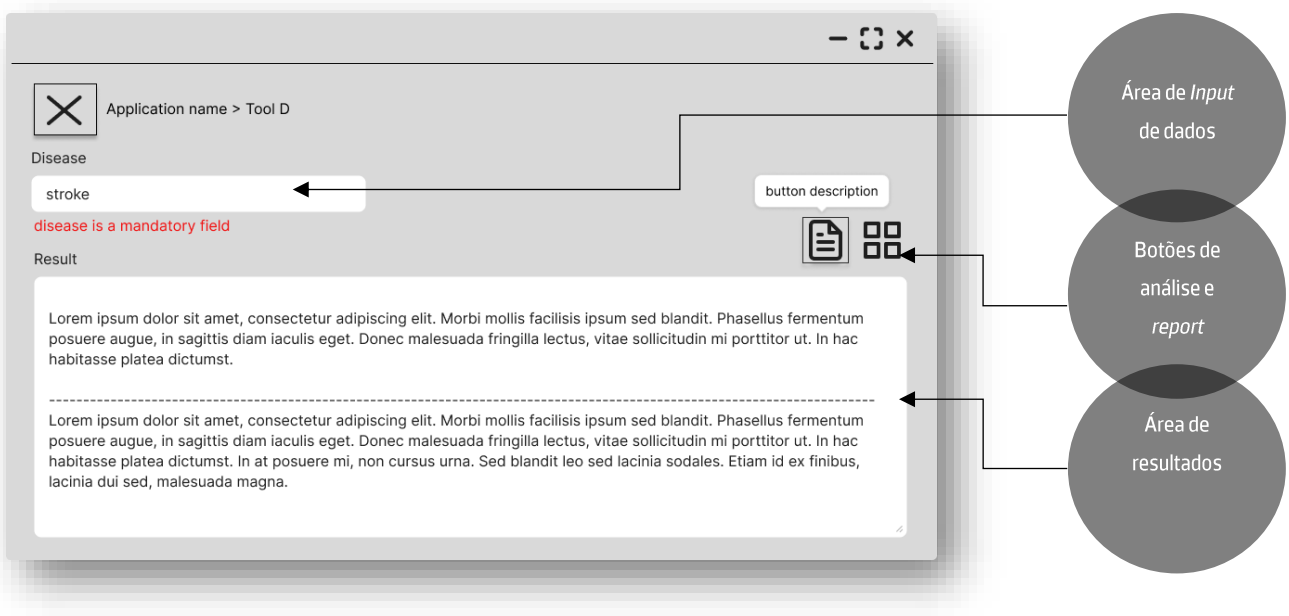


Figura 12 – Wireframe para ecrã/pagina de pesquisa de fármacos

4.4.6. Ecrã com junção de todas as ferramentas (*All in One*)

O ecrã ilustrado na Figura 13 representa a funcionalidade central e integradora da aplicação, combinando todos os serviços num fluxo único. O *design* inclui uma área de entrada de dados para o *upload* da sequência inicial. Abaixo, encontram-se os botões para iniciar a análise completa e para exportar o relatório consolidado. A área de resultados é a mais complexa, projetada para apresentar uma resposta única e organizada que engloba toda a informação recolhida sequencialmente das ferramentas ExPASy, Blast+, UniProt e DrugBank.

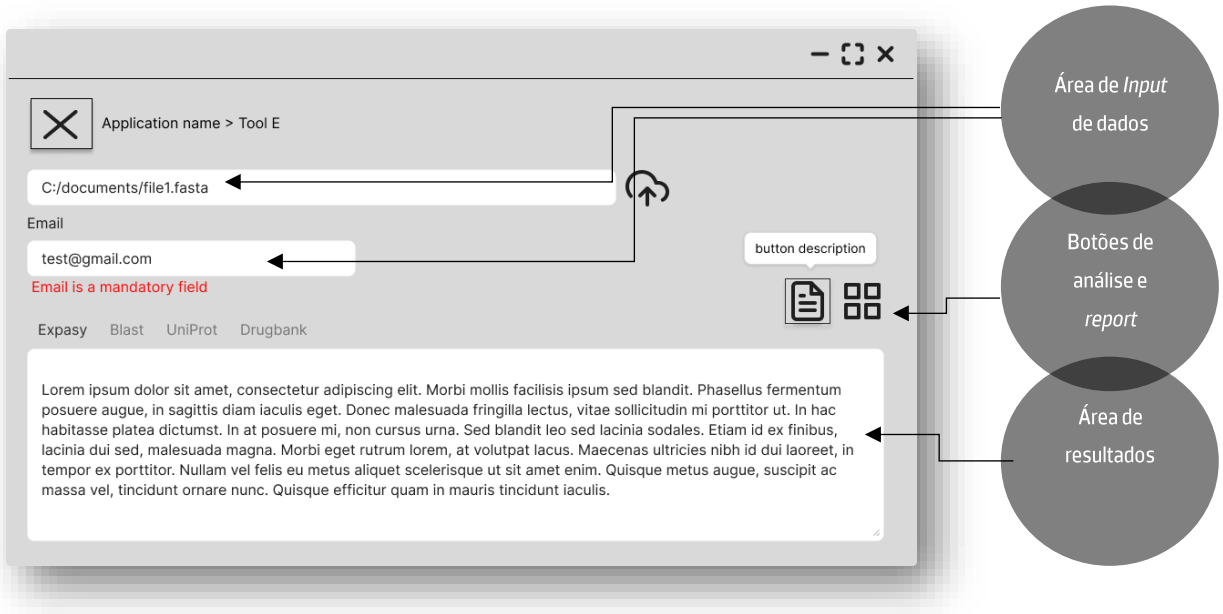


Figura 13 – Wireframe para ecrã/página que engloba todos os serviços e devolve uma resposta única com toda a informação

4.5. Soluções Desenvolvidas

4.5.1. Biopython – "*Found Sequence*"

O "*Found Sequence*" é o novo submódulo que ficará inserido dentro do módulo já existente *Bio* da biblioteca Biopython. Foi opção a sua inserção neste módulo, uma vez que a maior parte das funcionalidades usadas está aqui incluída. Na Figura 14 é ilustrada a estrutura do novo submódulo:

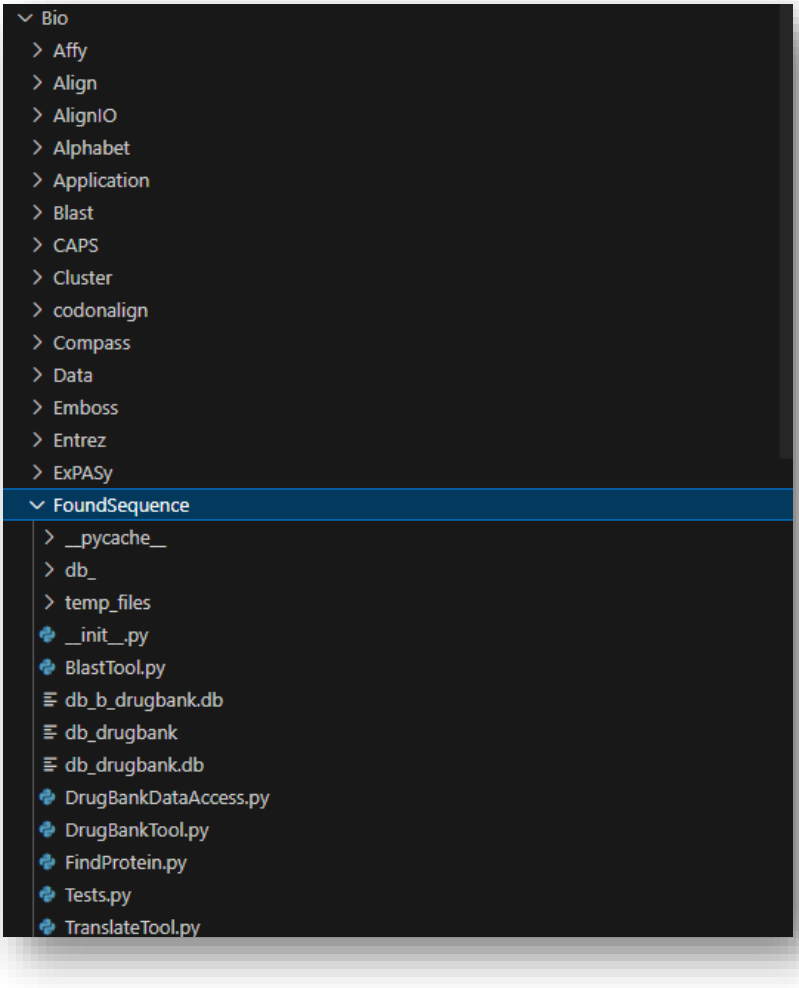


Figura 14 – Estrutura de ficheiros no novo sub-módulo desenvolvido

Neste submódulo temos quatro funcionalidades desenvolvidas e as respetivas funções para dar resposta às necessidades do projeto. As funcionalidades são as seguintes:

- Translate Tool;
- Blast Tool;
- Find Protein;
- DrugBank Tool.

De seguida, cada uma destas funcionalidades será detalhada relativamente à sua implementação e às principais funções.

4.5.1.1 Translate Tool

Foi implementado o *script TranslateTool.py*, em que a sua principal função é processar um ficheiro de entrada no formato FASTA, que contém sequências de nucleótidos (DNA/RNA). O *script* utiliza as funcionalidades da Biopython para ler estas sequências e invocar a chamada à API que irá realizar a tradução genética para sequências de aminoácidos, considerando os seis quadros de leitura (*reading frames*) possíveis. O resultado da tradução é então enviado também em formato FASTA, apresentando as sequências proteicas correspondentes a cada *frame*. De seguida é apresentada na Tabela 6 as funções contidas no *script TranslateTool.py*:

Tabela 6 – Tabela de funções e seus objetivos contidas no script TranslateTool.py

Função	Objetivo
`def validate_FileFormat(file):`	Função que valida se a extensão do ficheiro é válida, isto é, se é uma FASTA. Esta função é somente usada para a aplicação *desktop*, uma vez que na aplicação *web* essa validação é feita no controlo de *upload file*.
`def validate_FileEmpty(file)`	Função que valida se o ficheiro, apesar de ter a extensão correta, está vazio.
`def validate_Nucleotide_Sequence(file,web)`	Função que valida se a sequência de nucleótidos é válida, ou seja, se apenas existem os nucleótidos A, C, T e G.
`def expasy_Translate_Tool(file,web)`	Função que recebe o ficheiro e, se está a ser invocada via *web* ou *desktop*, aplica as validações necessárias em cada um dos casos e, de seguida, envia para a API o pedido de tradução e recebe a resposta com a tradução nos respetivos *frames*.
`def get_BigORF(protein)`	Após obter a resposta válida da função anterior, esta função vai ler todos os *frames* e obter qual é o maior ORF existente.

4.5.1.2 Blast Tool

Foi implementado o *script BlastTool.py*, em que a sua principal função é receber uma sequência, seja de nucleótidos ou aminoácidos, escolher o programa adequado, ou seja, blastp ou blastn, e enviar a sequência para o Blast+ para procurar a similaridade com a sequência de referência. Com estes dados é feita a chamada à API do Blast+ que devolve um ficheiro em formato JSON com o resultado. Caso seja encontrado um resultado válido, a sequência pesquisada é alinhada com a sequência de referência e é feita uma comparação entre as duas para deteção de diferenças que se traduzem em variantes. No final é devolvida a resposta com a identificação da sequência entre outros dados e variantes caso existam. De seguida é apresentada na Tabela 7 as funções contidas no *script BlastTool.py*:

Tabela 7 – Tabela de funções e seus objetivos contidas no script BlastTool.py

Função	Objetivo
`def validate_Empty_Email(email)`	Validação de envio de email que é *Input* obrigatório para a API do Blast+.
`def validate_Email_Format(email)`	Validação de formato válido de email.
`def find_Variants(query,subject)`	Dada a sequência pesquisada e a de referência, irá ser feita a comparação em tuplos de nucleótidos de uma e outra e perceber se há diferença e qual a posição onde a mesma ocorre.
`def validate_Nucleotide_Sequence(sequence)-> bool`	Se a sequência enviada para procura de similaridade é uma sequência de nucleótidos, então é feita a validação de cada um deles para despistar um nucleótido inválido.
`def validate_Protein_Sequence(sequence) -> bool`	Se a sequência enviada para procura de similaridade é uma sequência de aminoácidos, então é feita a validação de cada um destes para despistar um aminoácido inválido.
`def blast(email, program, matrix, alignments, scores, exp, dropoff, match_scores, gapopen, gapext, filter, seqrange, gapalign, compstats, align,`	Função que recebe todos os dados necessários à API do Blast+, e vai fazer a chamada da mesma enviando todos os *Inputs*.. A resposta é recebida em formato JSON, seja válida ou inválida. No caso de ser uma resposta válida, de seguida será chamada a função para identificar, caso existam, variantes na sequência.

stype, sequence, database)	

4.5.1.3 Find Protein

Foi implementado o *script FindProtein.py.*, em que a sua principal função é receber um identificador de uma proteína no UniProt, ou *hit accession*. Ex: P00441.

Com este identificador é feita a chamada à API do UniProt que irá devolver um ficheiro JSON com toda a informação referente à proteína se esta estiver ativa, caso contrário existe a indicação de que o registo encontra-se inativo. De seguida é apresentada na Tabela 8 as funções contidas no *script FindProtein.py*:

Tabela 8 – Tabela de funções e seus objetivos contidas no script FindProtein.py

Função	Objetivo
`def found_Uniprot_Protein(hit_accession)`	Função que recebe como *Input* o identificador da proteína, faz a chamada à API do UniProt e vai ter como resposta um JSON com a informação da proteína.

4.5.1.4 DrugBank Tool

Foi implementado o *script DrugBankTool.py e DrugBankDataAccess.py*, em que a sua principal função é receber uma patologia e, de seguida, fazer ligação à base de dados, executar a pergunta à mesma sobre quais fármacos podem ajudar ou minimizar os impactos da patologia e devolver todos os registos encontrados. De seguida é apresentada na tabela 9 as funções contidas no *script DrugBankTool.py*:

Tabela 9 – Tabela de funções e seus objetivos contidas nos scripts DrugBankTool.py e DrugbankDataAccess.py

Função	Objetivo
`def openConnection()`	Função cujo objetivo é abrir uma ligação para a base de dados. Será chamada sempre que existir uma questão – get_drug_by_disease.
`def closeConnection(connection)`	Função que faz o fecho da ligação à base de dados após ter sido feita a questão necessária e devolvida a resposta.
`def get_drug_by_disease(connection,disease)`	Função que recebe a patologia e executa a pergunta (*query*) em formato SQL à base de dados e recebe a resposta com os resultados.

`def found_Drug(disease)`	Função principal que chama as anteriores e devolve a resposta final.

4.5.1.5 *All in One* Tool

Foi implementado __init__.*py* em que a sua principal função é receber todos os dados necessários para a chamada das várias APIs e da base de dados dos serviços detalhados anteriormente. Pode receber uma resposta completa, isto é, uma tradução, uma similaridade com variantes, informação sobre a proteína e respetiva doença associada a variantes e um ou mais fármacos associados à minimização ou resolução da patologia. Pode também existir apenas uma resposta parcial, isto é, imaginemos que não existem variantes na pesquisa de similaridade, não fará sentido continuar com a pesquisa. A resposta é uma estrutura complexa onde consta toda a informação encontrada, no caso de resposta parcial, a estrutura mantém-se, mas o que não tem informação virá com a estrutura vazia. De seguida é apresentada na Tabela 10 as funções contidas no *script __init__.py*:

Tabela 10 – Tabela de funções e seus objetivos contidas no script __ini__.py

Função	Objetivo
`def foundSequence(file, email, program, matrix, alignments, scores,exp,dropoff,match_scores, gapopen,gapext,filter,seqrange, gapalign,compstats,align,stype, database,web)`	Função que tem todos os *Inputs* necessários para chamar todos as ferramentas detalhadas anteriormente pela seguinte ordem: • TranslateTool; • BlastTool; • FindProtein; • DrugBankTool. A resposta é uma estrutura complexa onde consta toda a informação encontrada.
`def read_Blast_Json_Protein(json_file)`	Como a resposta do Blast é em formato JSON, é necessário ter uma função que vai ler esta estrutura e retirar a informação necessária para a estrutura complexa.
`def read_Uniprot_Json(json_file,variants)`	Como a resposta da UniProt é em formato JSON, é necessário ter uma função que vai ler esta estrutura e retirar a informação necessária para a estrutura complexa.

4.5.2. Aplicação *Desktop*

Para validar e demonstrar as capacidades das novas funcionalidades integradas na biblioteca Biopython, e para facilitar a sua utilização por parte de utilizadores finais, procedeu-se ao desenvolvimento de uma aplicação *desktop*. Esta abordagem foi escolhida por oferecer um ambiente de execução robusto e com acesso direto aos recursos do sistema, ideal para processamento de dados bioinformáticos. Foi selecionada a *framework* PyQt6 v6.7.1 para a construção da interface gráfica, devido à sua flexibilidade, ao vasto conjunto de componentes de UI (*widgets*) disponíveis e à sua integração nativa com Python, o que assegura uma comunicação fluida com a biblioteca "Found Sequence".
A aplicação *desktop* foi desenhada para ser intuitiva, guiando o utilizador através do fluxo de análise, desde a submissão dos dados de entrada até à visualização dos resultados. Esta aplicação não só simplifica a interação do utilizador com o módulo "Found Sequence", mas também permite uma visualização clara e prática dos resultados combinados das diferentes ferramentas bioinformáticas.

4.5.2.1 UI Tool

A implementação exigiu um período de familiarização com o ambiente PyQt6 e a instalação das bibliotecas dependentes requeridas. Para a conceção visual dos ecrãs, recorreu-se à ferramenta Qt Designer, parte do ecossistema PyQt, permitindo traduzir os *wireframes* inicialmente desenhados (conforme descrito no ponto 4.4) numa interface funcional. O Qt Designer permitiu criar um protótipo visual da aplicação, posicionando elementos como botões, áreas de texto e secções de resultados de forma precisa. Cada ecrã da aplicação foi guardado como um ficheiro `.ui`, que foi posteriormente convertido para código Python para integração e implementação da lógica de negócio.

4.5.2.2 Configurações/Instalações

Para dar início ao desenvolvimento da aplicação, foi necessário seguir um conjunto de passos essenciais para configurar o ambiente de trabalho:

1. Instalação da ferramenta Qt Designer para conceção dos ecrãs da aplicação;
2. Instalação no Visual Studio Code da biblioteca PyQt6 para poder interagir com o projeto de interface gráfica;
3. No caso, foi utilizado o seguinte comando no terminal do IDE: `pip install PyQt6`

A estrutura do projeto foi organizada de forma modular, separando a lógica da interface gráfica do código de controlo da aplicação. Esta separação facilita a manutenção e futuras atualizações.

4.5.2.3 Ecrãs e Interação com "Found Sequence" – Biopython

De seguida serão detalhados os ecrãs de cada ferramenta e a sua interação com as funções descritas na secção 5.4.1 – Biopython:

Ecrã Inicial

Tal como idealizado nos *wireframes*, existe um ecrã inicial que funciona como um portal, dando acesso a cada uma das ferramentas individuais e à funcionalidade "*All in One*". Cada opção é acompanhada por uma breve explicação do seu propósito, orientando o utilizador sobre as capacidades de cada módulo. Este ecrã inicial ilustrado na Figura 15, foi concebido para ser o ponto de partida central, estabelecendo uma navegação clara e lógica.

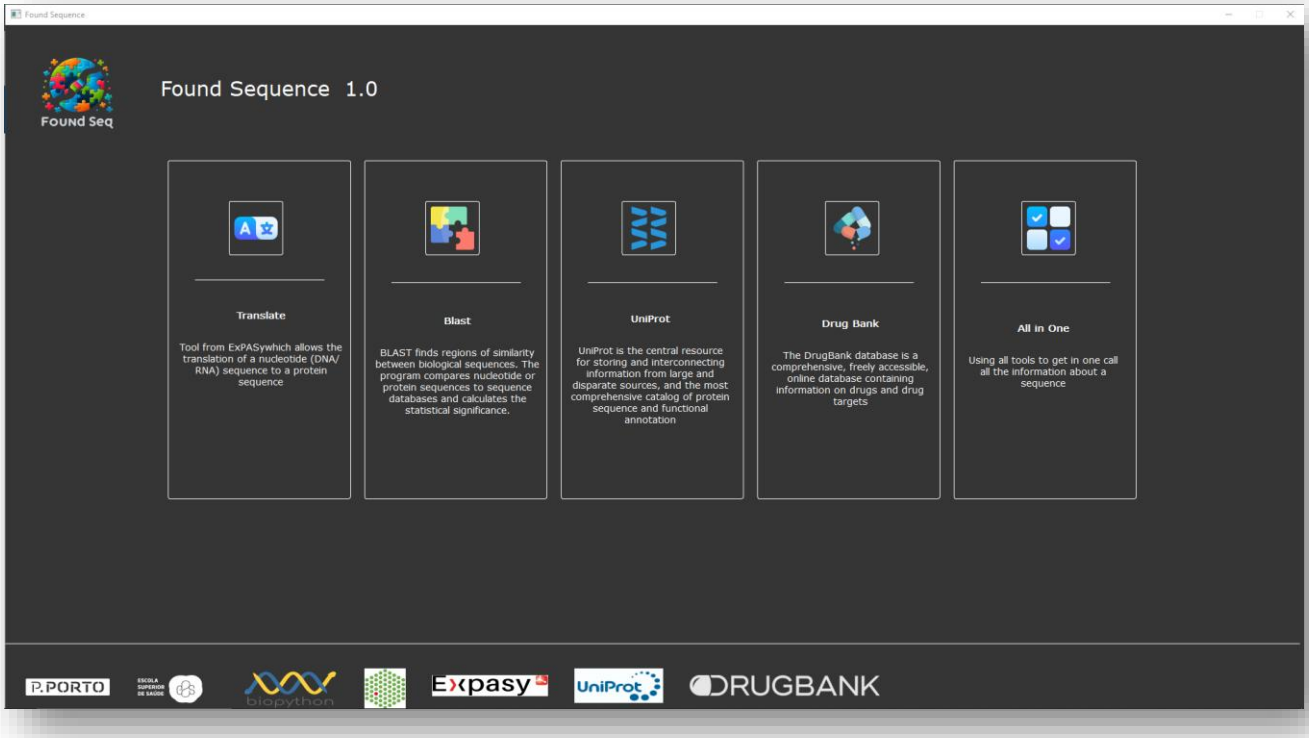


Figura 15 – Ecrã inicial da aplicação Desktop

Ecrã Translate Tool

O ecrã ilustrado na Figura 16 permite ao utilizador fazer o *upload* de um ficheiro em formato FASTA. A interface aciona as funções de validação da biblioteca "Found Sequence" para garantir que o ficheiro tem o formato correto e não está vazio. Após a validação, o ficheiro é enviado para a função expasy_Translate_Tool, que, por sua vez, interage com a API do ExPASy. Os resultados, incluindo as sequências traduzidas e a maior ORF, são exibidos numa área de texto designada. O utilizador tem ainda a opção de gerar um relatório em PDF com os resultados obtidos.

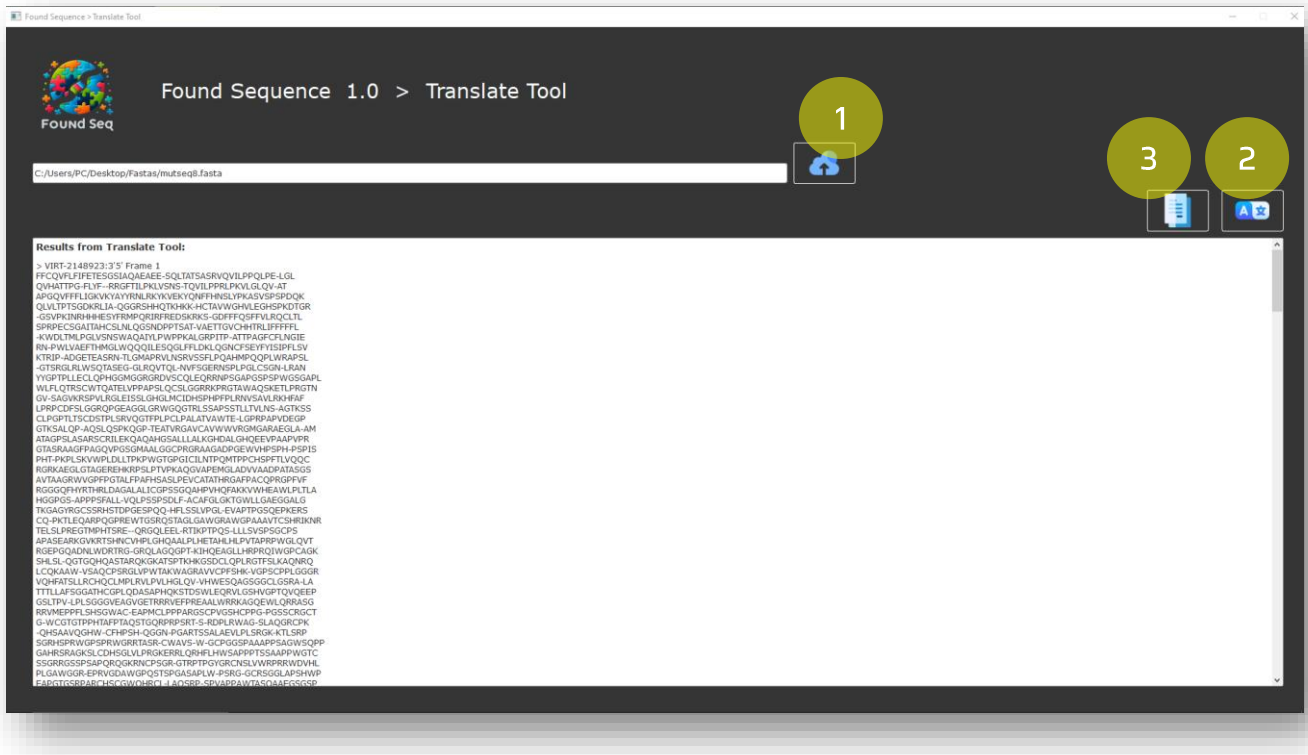


Figura 16 – Ecrã com área de inserção de dados e área de resultados para interação com a Translate Tool

De seguida temos representadas na Tabela 11 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *Translate Tool*:

*Tabela 11 – Tabela com ações do ecrã Translate Tool e interação com as funções da biblioteca "*Found Sequence"

Ação	Interação com "*Found Sequence*" – Translate Tool
1 – *Upload* de ficheiro	`def validate_FileFormat(file):`
	`def validate_FileEmpty(file)`
	`def validate_Nucleotide_Sequence(file,web)`

2 – Enviar ficheiro à API do ExPASY Translate Tool e obter resposta com tradução da sequência em *frames* 5'3' e 3'5' bem como o maior ORF.	`def expasy_Translate_Tool(file,web)` `def get_BigORF(protein)`
3 – Gerar PDF	Aqui não há interação com o "*Found Sequence*", contudo, no projeto aplicacional foi desenvolvida uma função que vai exportar os dados devolvidos para uma página HTML, permitindo fazer a impressão para um ficheiro PDF.

Ecrã Blast Tool

No ecrã ilustrado na Figura 17, o utilizador pode inserir uma sequência de nucleótidos ou de proteína numa caixa de texto. Ao clicar no botão de análise, a aplicação invoca a função blast da biblioteca "Found Sequence", passando a sequência e outros parâmetros necessários.

A biblioteca gere a comunicação com a API do NCBI Blast+, processa a resposta e retorna a informação formatada, que inclui a identificação da sequência, o alinhamento, a presença de *gaps* e as variantes encontradas. Os resultados são apresentados de forma estruturada, facilitando a interpretação pelo utilizador, que pode também exportá-los para PDF.

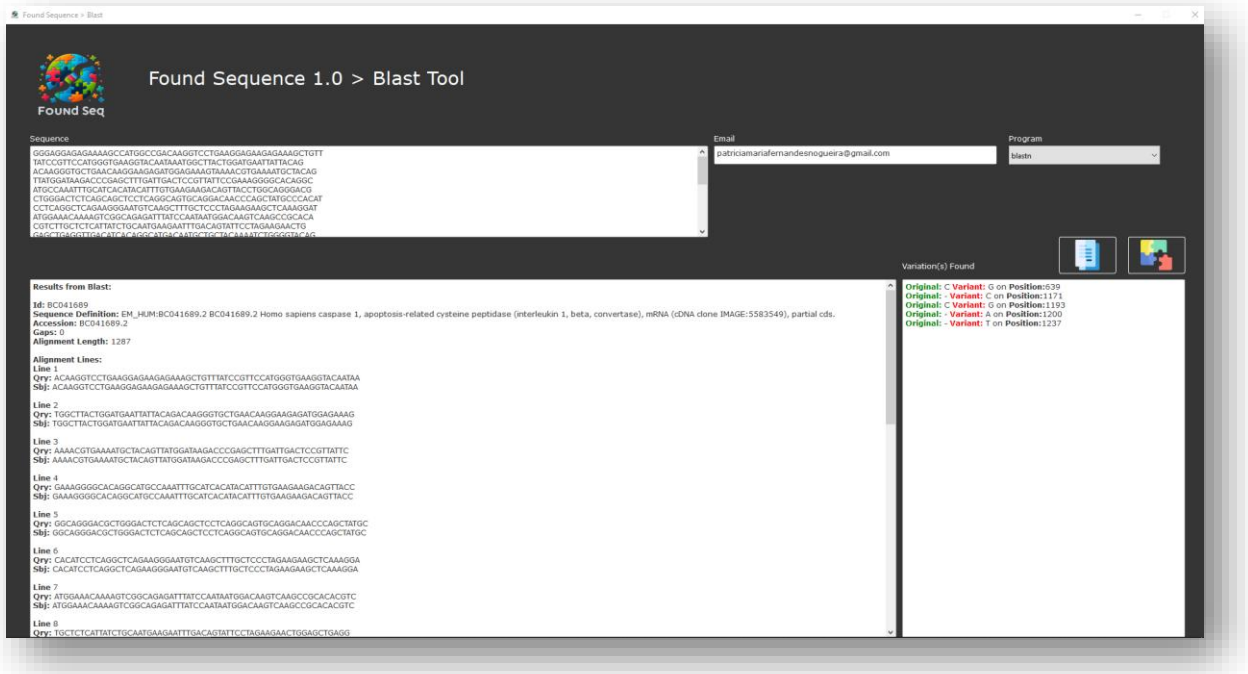


Figura 17 -Ecrã com área de inserção de dados e área de resultados para interação com a Blast Tool

De seguida temos representadas na Tabela 12 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *Blast Tool*:

Tabela 12 – Tabela com ações do ecrã Blast Tool e interação com as funções da biblioteca Found Sequence

Ação	Interação com Found Sequence – Blast Tool
1 – Enviar uma sequência de nucleótidos ou proteína à API do Blast Tool e obter resposta com: • Identificação da sequência • Alinhamento • Gaps • Tamanho da sequência • Variantes encontradas e respetiva posição	`def blast(email, program, matrix, alignments,scores,exp, dropoff, match_scores, gapopen, gapext, filter, seqrange, gapalign, compstats, align, stype, sequence, database)`
2 – Gerar PDF	Aqui não há interação com o "*Found Sequence*", contudo, no projeto aplicacional foi desenvolvida uma função que vai exportar os dados devolvidos para uma página HTML, permitindo fazer a impressão para um ficheiro PDF.

Ecrã UniProt Tool

O ecrã ilustrado na Figura 18 foi projetado para pesquisas diretas na base de dados UniProt. O utilizador insere um identificador único (*hit accession*) da proteína de interesse. A aplicação chama a função found_Uniprot_Protein, que envia o identificador para a API do UniProt. A informação retornada, como a identificação da proteína, a sua função e as variantes associadas a doenças, é exibida na área de resultados. A interface permite uma consulta rápida e focada, com a possibilidade de gerar um relatório detalhado.

Figura 18 – Ecrã com área de inserção de dados e área de resultados para interação com a UniProt Tool

De seguida temos representadas na Tabela 13 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *UniProt:*

*Tabela 13 – Tabela com ações do ecrã UniProt Tool e interação com as funções da biblioteca "*Found Sequence*"*

Ação	**Interação com "*Found Sequence*" – UniProt Tool**
1 – Enviar identificador único (hit accession) à API da UniProt Tool e obter resposta com: • Identificação da proteína • Função • Identificação de todas as variantes possíveis, doenças associadas e descrição.	`def found_Uniprot_Protein(hit_accession)`
2 – Gerar PDF	Aqui não há interação com o "*Found Sequence*", contudo, no projeto aplicacional foi desenvolvida

	uma função que vai exportar os dados devolvidos para uma página HTML, permitindo fazer a impressão para um ficheiro PDF.

Ecrã DrugBank

O ecrã ilustrado na Figura 19 destina-se à consulta de fármacos, o utilizador insere o nome de uma patologia. A aplicação interage com a função found_Drug, que, por sua vez, consulta a base de dados local do DrugBank para encontrar fármacos associados à doença indicada. Os resultados, incluindo os nomes dos fármacos e as suas descrições, são apresentados ao utilizador, que pode exportar a lista para um ficheiro PDF.

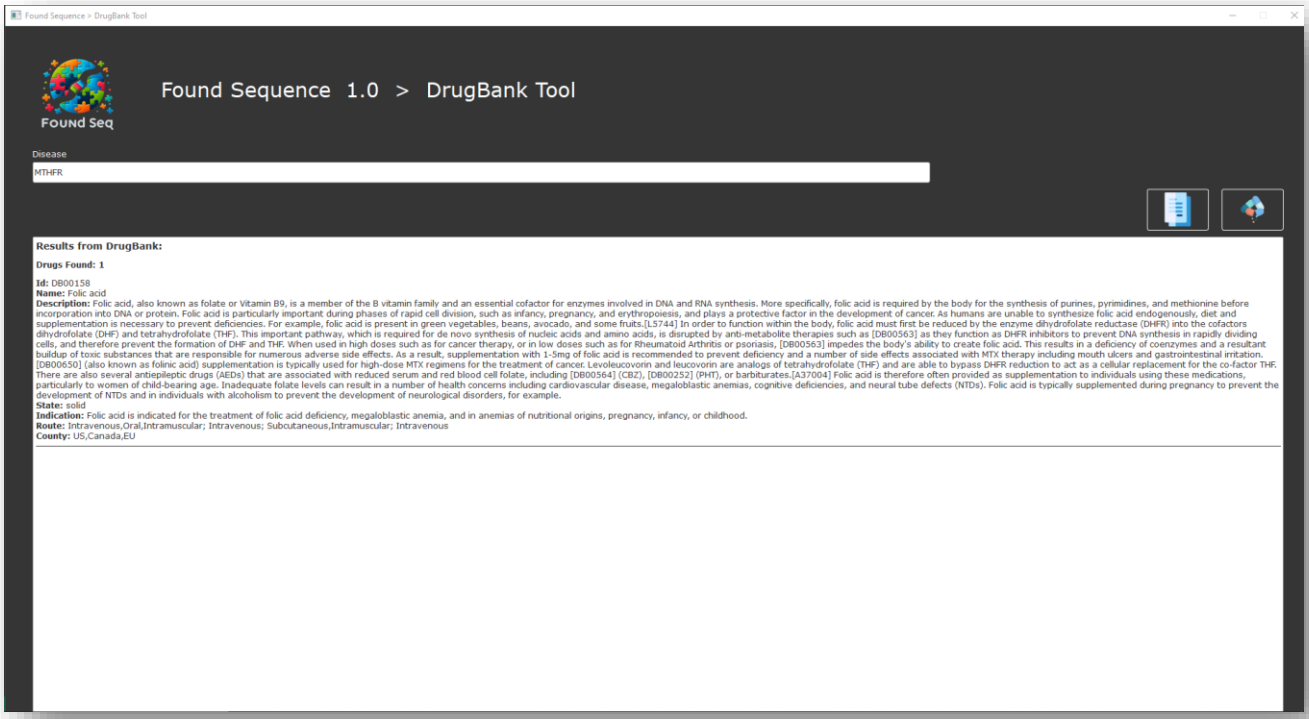


Figura 19 – Ecrã com área de inserção de dados e área de resultados para interação com a DrugBank Tool

De seguida temos representadas na Tabela 14 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *DrugBank Tool*:

*Tabela 14 – Tabela com ações do ecrã DrugBank e interação com as funções da biblioteca "*Found Sequence*"*

Ação	**Interação com "*Found Sequence*" – DrugBank Tool**
1 – Enviar uma patologia à base de dados da DrugBank e tentar encontrar um ou mais fármacos que possam ajudar na minimização/prevenção da mesma.	`def found_Drug(diseases)`
2 – Gerar PDF	Aqui não há interação com o "*Found Sequence*", contudo, no projeto aplicacional foi desenvolvida uma função que vai exportar os dados devolvidos para uma página HTML, permitindo fazer a impressão para um ficheiro PDF.

Ecrã *All in One*

O ecrã ilustrado na Figura 20 representa a funcionalidade central do projeto, integrando todas as ferramentas num único fluxo de trabalho. O utilizador inicia o processo fazendo o *upload* de um ficheiro FASTA. A aplicação chama a função foundSequence, que orquestra sequencialmente as chamadas às APIs do ExPASy, Blast, UniProt e à base de dados do DrugBank, passando os dados relevantes de uma etapa para a outra. Os resultados de cada uma das ferramentas são apresentados em secções distintas dentro da mesma janela, oferecendo uma visão completa e consolidada da análise. A opção de gerar um relatório final em PDF agrega toda a informação num único documento.

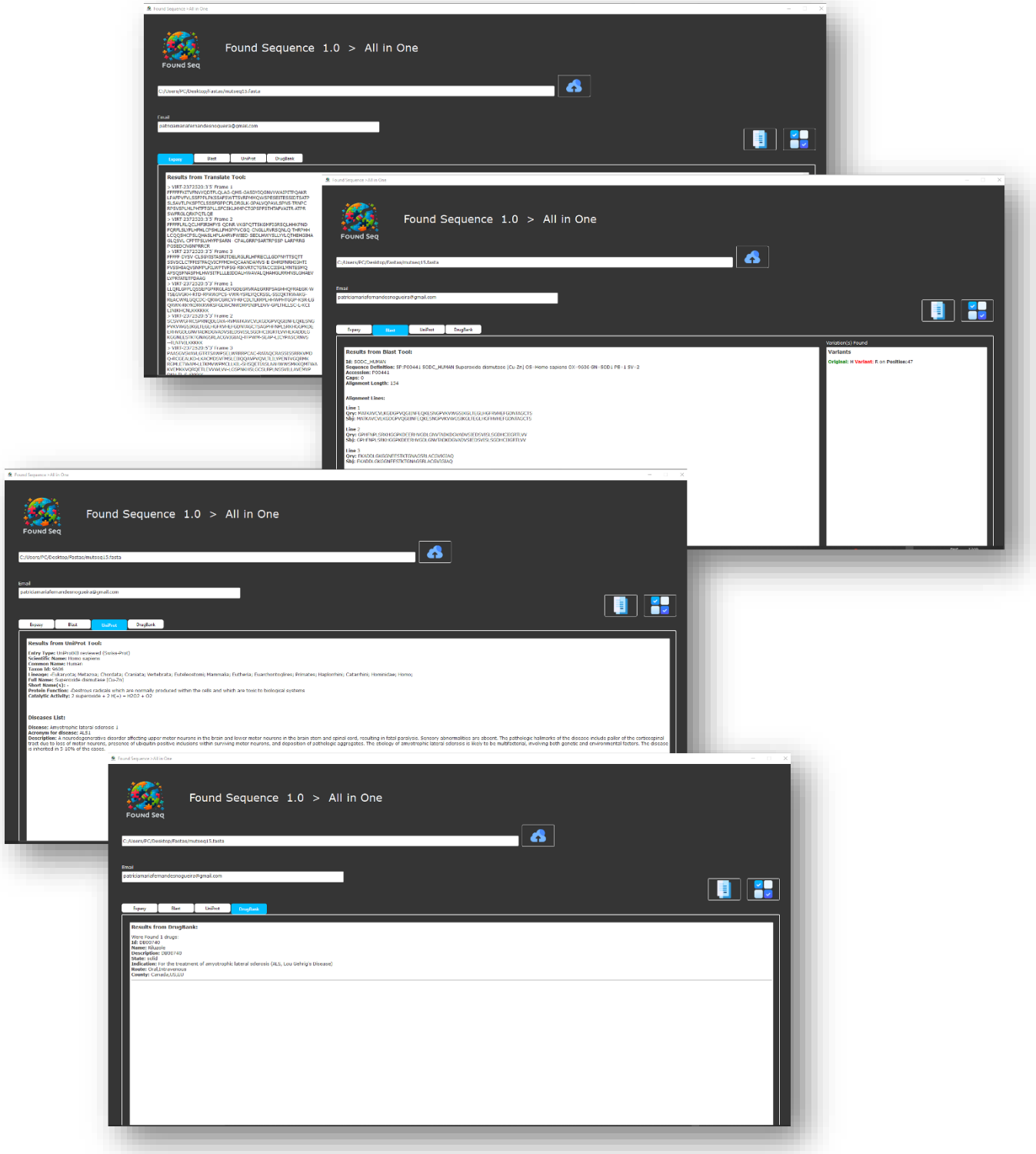


Figura 20 – Ecrã com área de inserção de dados e área de resultados com 4 secções que mostram resultados pata todas as ferramentas, ExPASy, Blast, UniProt e DrugBank

De seguida temos representadas na Tabela 15 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *All in One*:

*Tabela 15 – Tabela com ações do ecrã All in One e interação com as funções da biblioteca "*Found Sequence*"*

Ação	Interação com "*Found Sequence*" – *All in One*
1 – *Upload* de ficheiro	`def validate_FileFormat(file)`
	`def validate_FileEmpty(file)`
	`def validate_Nucleotide_Sequence(file,web)`
2 – Enviar ficheiro ao Found Sequence que, após receber os *Inputs*, vai internamente fazer ordenadamente chamadas às APIs da ExPASy, Blast, UniProt e à base de dados da DrugBank.	`def foundSequence(file, email,program,matrix, alignments,scores,exp, dropoff,match_scores,gapopen, gapext,filter,seqrange, gapalign,compstats,align, stype,database,web)`
3 – Gerar PDF	Aqui não há interação com o "*Found Sequence*", contudo, no projeto aplicacional foi desenvolvida uma função que vai exportar os dados devolvidos de todas as ferramentas para uma página HTML, permitindo fazer a impressão para um ficheiro PDF.

4.5.3. Aplicação *Web*

Tal como a aplicação *desktop*, foi também desenvolvida uma aplicação *web* com o mesmo propósito, isto é, validar e demonstrar as capacidades das novas funcionalidades integradas na biblioteca Biopython. A criação de uma versão *web* foi motivada pela necessidade de oferecer uma solução mais acessível e que não exigisse qualquer tipo de instalação por parte do utilizador final. Para além de simplificar a utilização da nova biblioteca, a aplicação *web* tem benefícios face à aplicação *desktop*, nomeadamente:

- **Acessibilidade Multiplataforma**: As aplicações *web* são acessíveis a partir de qualquer dispositivo com um navegador *web* e ligação à internet (*desktops*, *laptops*, *tablets*, *smartphones*), independentemente do sistema operativo (Windows, macOS, Linux, iOS, Android). Isto elimina a necessidade de desenvolver e manter versões separadas para diferentes plataformas. Os utilizadores não precisam de instalar qualquer *software* específico nos seus dispositivos, tornando o acesso mais fácil e rápido.
- **Manutenção e Atualizações Centralizadas**: As atualizações e correções de *bugs* são feitas no servidor e ficam instantaneamente disponíveis para todos os utilizadores na próxima vez que acedem à aplicação. A gestão da aplicação é centralizada no servidor, o que pode simplificar a manutenção da infraestrutura.
- **Escalabilidade**: É geralmente mais fácil integrar aplicações *web* com outros serviços *online* e APIs, o que abre portas para futuras expansões e a incorporação de novas funcionalidades de forma mais dinâmica.

4.5.3.1 *Web* Framework – Configurações/Instalações

Para o desenvolvimento da aplicação *web*, foi escolhida a *framework* Django, devido à sua robustez, segurança e ao seu paradigma MVT (*Model-View-Template*), que promove um desenvolvimento rápido e organizado. Para dar início à aplicação, foi necessário:

4. **Instalação do Django** no terminal do Visual Studio Code através do comando: `pip install django;`
1. **Criar um projeto Django** ainda no mesmo terminal: `django-admin startproject <nome_do_projeto>;`
2. **Configurações necessárias** após a criação do projeto:
    - Configurar o *debugger* para o Visual Studio Code fazer *debug* de aplicações Django;

- No terminal, executar o servidor de desenvolvimento com o comando: `python manage.py runserver`.

A estrutura do projeto foi dividida em várias aplicações Django, cada uma responsável por uma das ferramentas bioinformáticas, o que resultou num código mais limpo e de fácil manutenção.

4.5.3.2 Ecrãs e Interação com "Found Sequence" – Biopython

A interface da aplicação *web* foi desenhada para ser limpa e funcional, seguindo de perto os *wireframes* originais. A navegação é gerida através de um menu superior, que permite ao utilizador aceder a cada uma das ferramentas individualmente ou à página de resumo. De seguida, serão detalhadas as páginas de cada ferramenta e a sua interação com as funções da biblioteca "Found Sequence":

Página Inicial e Resumo das Ferramentas

A página inicial (*homepage*) ilustrada na Figura 21 serve como portal de entrada, com uma navegação clara para as diferentes secções. Uma página adicional representada na Figura 22, "Tools in a Nutshell", oferece um resumo visual e descritivo de cada ferramenta, semelhante ao ecrã inicial da aplicação *desktop*, ajudando os utilizadores a compreender rapidamente o propósito de cada funcionalidade.

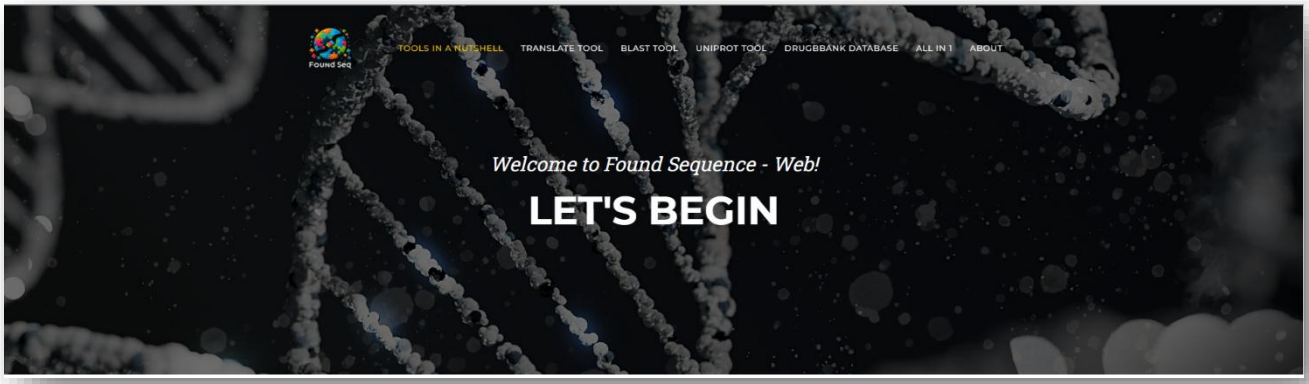


Figura 21 – Ecrã inicial da aplicação Web

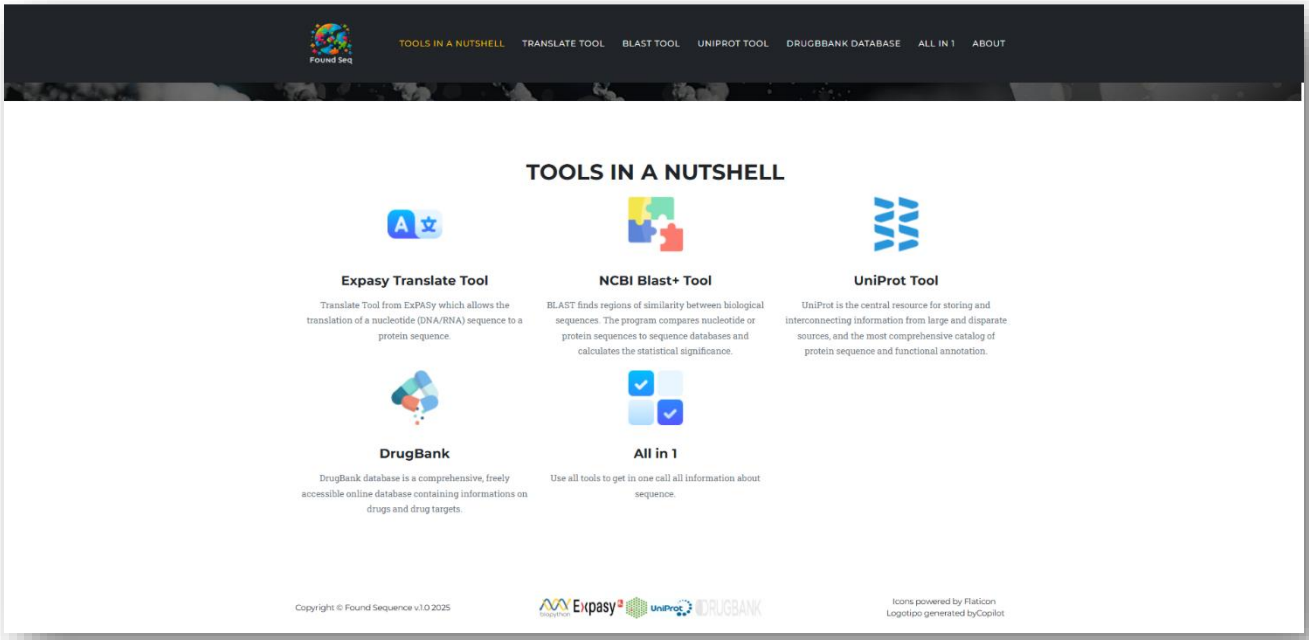


Figura 22 – Página com resumo de cada ferramenta aproximado ao pensado nos wireframes

Página Translate Tool

Na página representada na Figura 23, o utilizador encontra um formulário para o *upload* de ficheiros FASTA. A validação do tipo de ficheiro é gerida diretamente pelos controlos do formulário *web*, que restringem o *upload* a extensões permitidas. Após o envio, o *backend* da aplicação (Django) recebe o ficheiro e invoca a função expasy_Translate_Tool da biblioteca "Found Sequence". Os resultados da tradução e a identificação da maior ORF são então renderizados e apresentados dinamicamente na mesma página, sem necessidade de recarregamento.

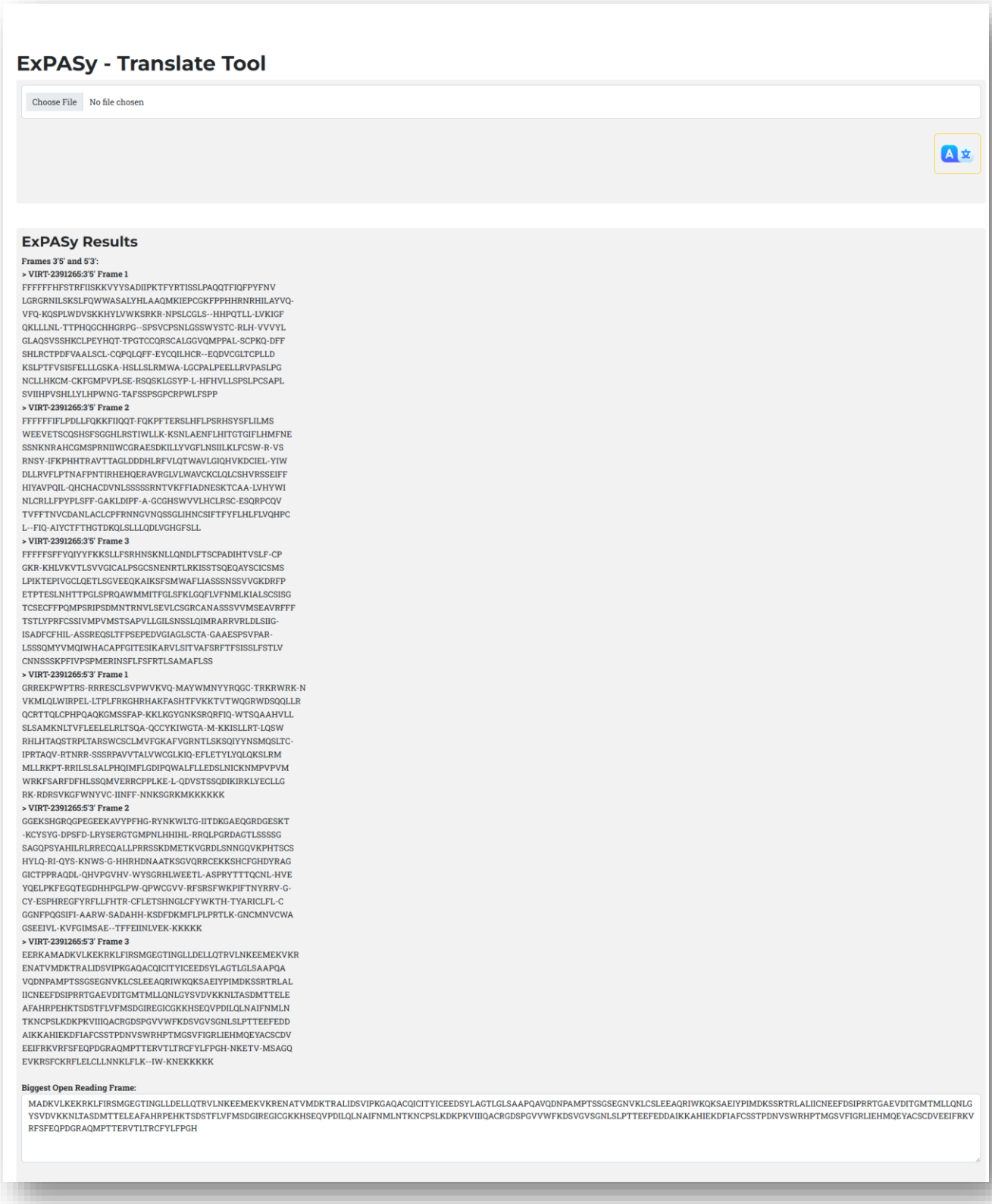


Figura 23 – Página com área de inserção de dados e área de resultados para interação com a Translate Tool

De seguida temos representadas na Tabela 16 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *ExPASy Translate Tool*:

*Tabela 16 – Tabela com ações da página Translate Tool e interação com as funções da biblioteca "*Found Sequence*"*

Ação	**Interação com "*Found Sequence*" – Translate Tool**

1– Enviar ficheiro à API do ExPASY Translate Tool e obter resposta com tradução da sequência em *frames* 5'3' e 3'5' bem como o maior ORF.
Neste caso, a validação existente na aplicação *desktop* é para *upload* de ficheiro obrigatório e validação de extensão FASTA não é necessária porque o componente de *upload* de ficheiro na *web* já contém essas propriedades por defeito.
	`def expasy_Translate_Tool(file,web)`
	`def validate_Nucleotide_Sequence(file,web)`
	`def get_BigORF(protein)`

Página Blast Tool

A página ilustrada na Figura 24 contém uma área de texto onde o utilizador pode colar a sequência a ser analisada. O formulário *web* envia a sequência para o servidor, que chama a função Blast da biblioteca. Os resultados do alinhamento, variantes e outras informações são processados no *backend* e depois apresentados numa secção de resultados na página, formatados para uma leitura clara e organizada.

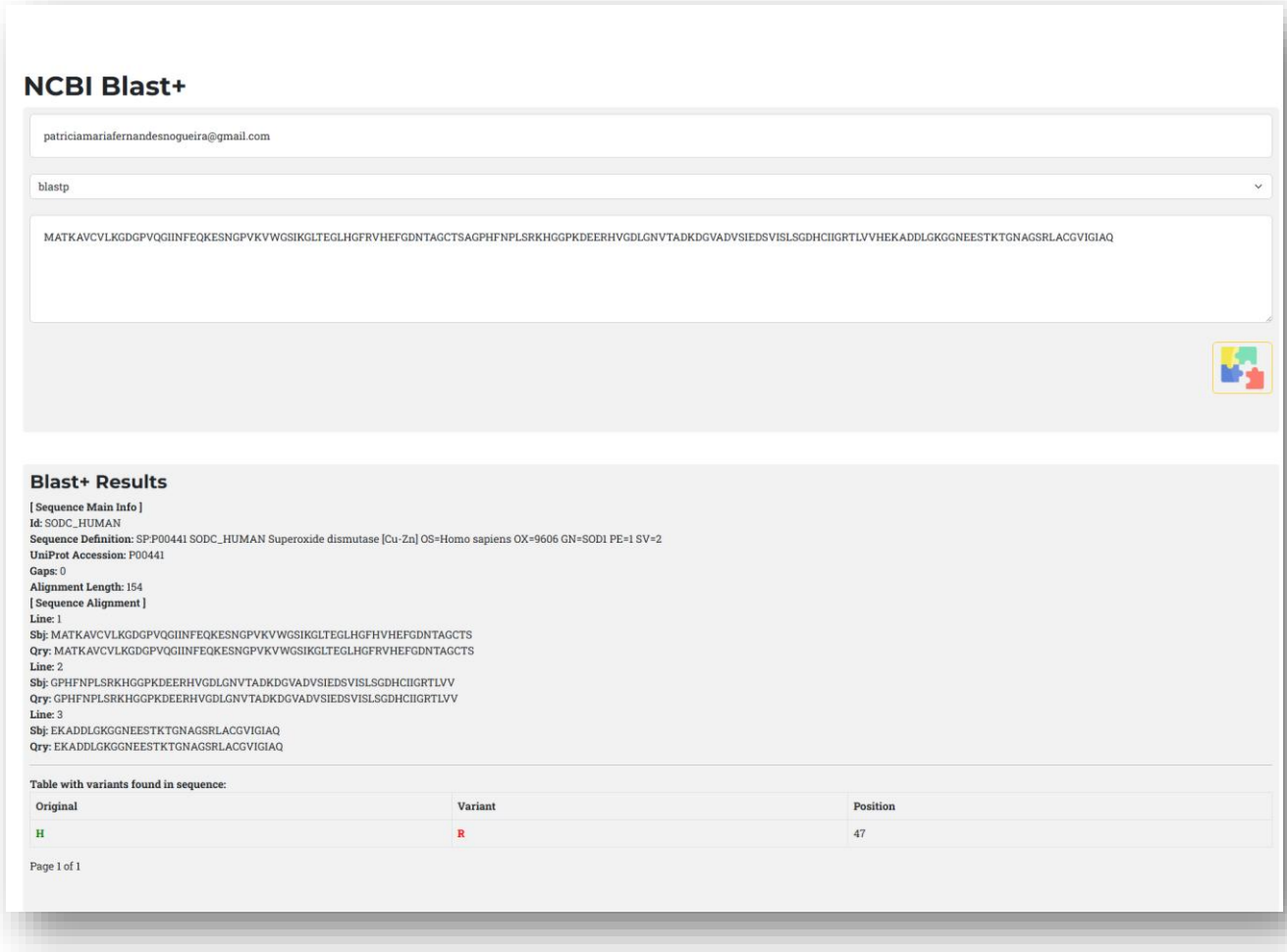


Figura 24 – Página com área de inserção de dados e área de resultados para interação com a Blast Tool

De seguida temos representadas na Tabela 17 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *NCBI Blast + Tool*:

*Tabela 17 – Tabela com ações da página Blast Tool e interação com as funções da biblioteca "*Found Sequence*"*

Ação	Interação com "*Found Sequence*" – Blast Tool

1 – Enviar uma sequência de nucleótidos ou proteína à API do Blast Tool e obter resposta com:
• Identificação da sequência;
• Alinhamento;
• *Gaps;*
• Tamanho da sequência;
• Variantes encontradas e respetiva posição.
	`def blast(email,`
	`program,`
	`matrix,`
	`alignments,`
	`scores,`
	`exp,`
	`dropoff,`
	`match_scores,`
	`gapopen,`
	`gapext,`
	`filter,`
	`seqrange,`
	`gapalign,`
	`compstats,`
	`align,`
	`stype,`
	`sequence,`
	`database)`

Página UniProt Tool

A página ilustrada na Figura 25 permite ao utilizador pesquisar informações sobre uma proteína através do seu *hit accession*. Um campo de texto simples captura o identificador, que é enviado ao servidor para ser processado pela função found_Uniprot_Protein. Os dados da proteína, incluindo a sua função e as variantes associadas a doenças, são retornados e exibidos na página de forma estruturada.

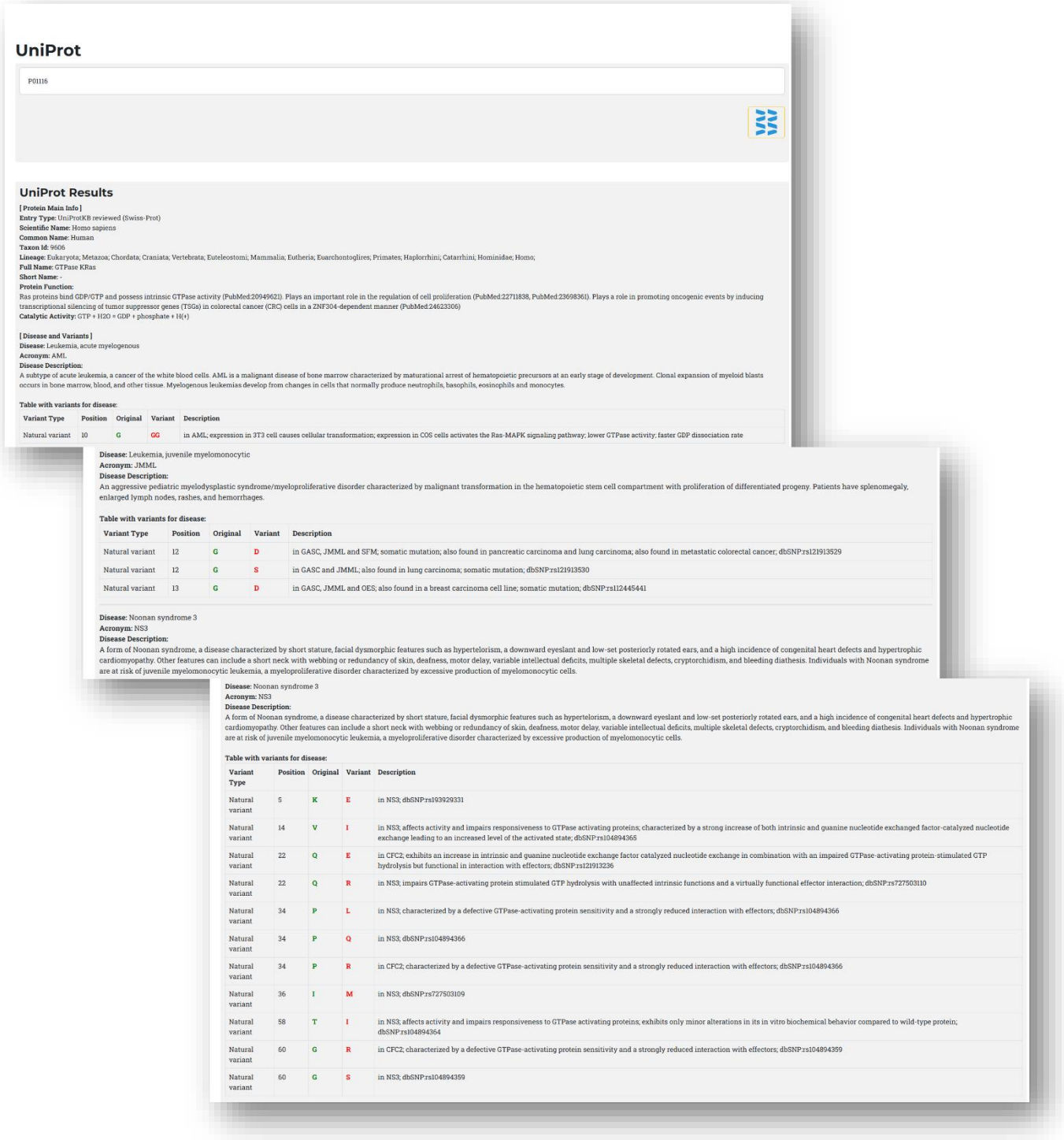


Figura 25 – Página com área de inserção de dados e área de resultados para interação com a UniProt Tool

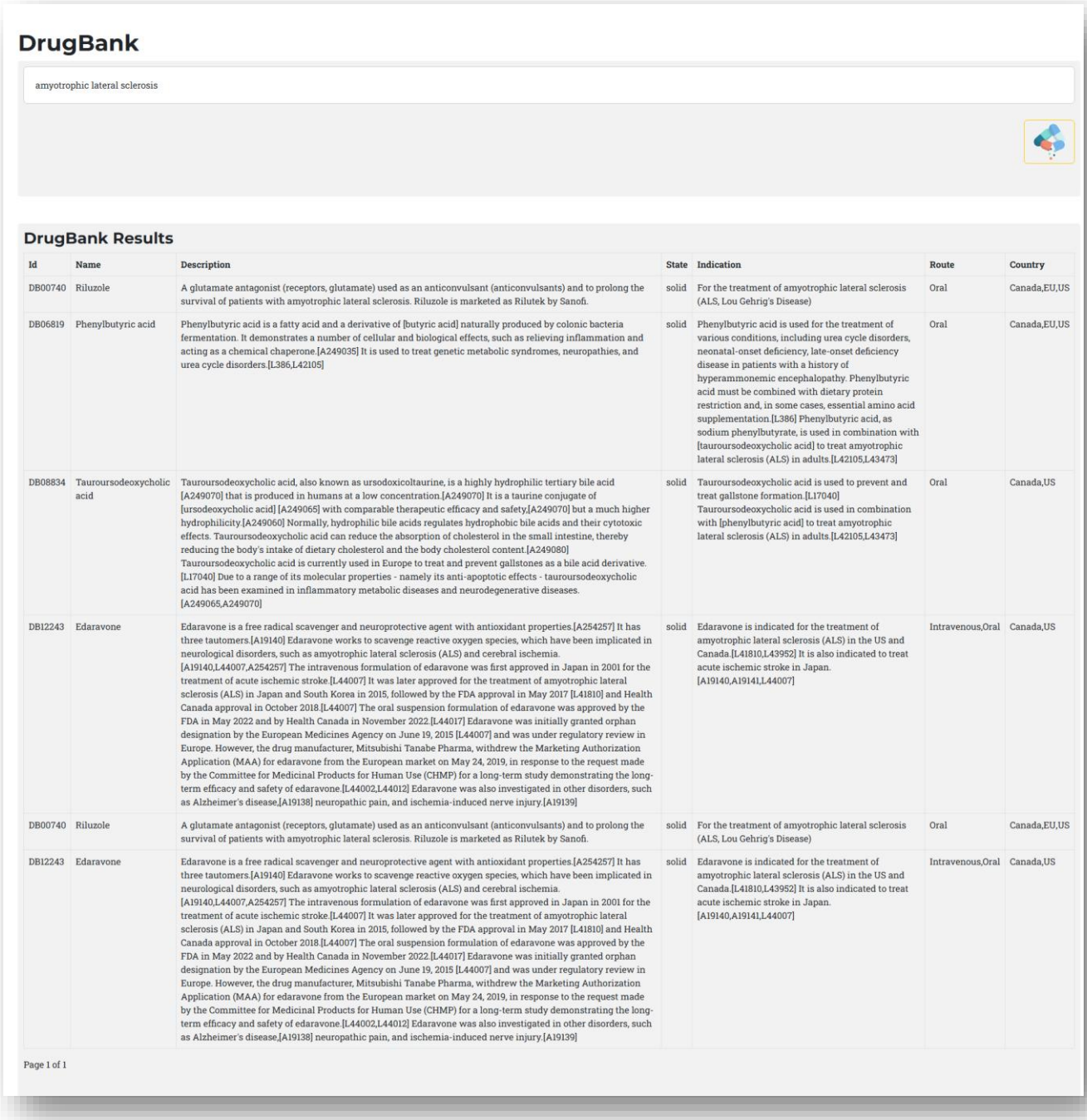
De seguida temos representadas na Tabela 18 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *UniProt – Protein Tool*:

*Tabela 18 – Tabela com ações do ecrã UniProt Tool e interação com as funções da biblioteca "*Found Sequence"

Ação	Interação com "*Found Sequence*" – UniProt Tool
1 – Enviar identificador único (*hit accession*) à API da UniProt Tool e obter a resposta com: • Identificação da proteína; • Função; • Identificação de todas as variantes possíveis, doenças associadas e descrição.	`def found_Uniprot_Protein(hit_accession)`

Página DrugBank

De forma semelhante às outras páginas, a interface do DrugBank ilustrada na Figura 26 apresenta um campo de entrada para o nome de uma patologia. Após a submissão, o servidor executa uma consulta à base de dados local através da função found_Drug e retorna uma lista de fármacos relevantes, que é apresentada ao utilizador numa tabela de resultados.



Id	Name	Description	State	Indication	Route	Country
DB00740	Riluzole	A glutamate antagonist (receptors, glutamate) used as an anticonvulsant (anticonvulsants) and to prolong the survival of patients with amyotrophic lateral sclerosis. Riluzole is marketed as Rilutek by Sanofi.	solid	For the treatment of amyotrophic lateral sclerosis (ALS, Lou Gehrig's Disease)	Oral	Canada,EU,US
DB06819	Phenylbutyric acid	Phenylbutyric acid is a fatty acid and a derivative of [butyric acid] naturally produced by colonic bacteria fermentation. It demonstrates a number of cellular and biological effects, such as relieving inflammation and acting as a chemical chaperone.[A249035] It is used to treat genetic metabolic syndromes, neuropathies, and urea cycle disorders.[L386,L42105]	solid	Phenylbutyric acid is used for the treatment of various conditions, including urea cycle disorders, neonatal-onset deficiency, late-onset deficiency disease in patients with a history of hyperammonemic encephalopathy. Phenylbutyric acid must be combined with dietary protein restriction and, in some cases, essential amino acid supplementation.[L386] Phenylbutyric acid, as sodium phenylbutyrate, is used in combination with [tauroursodeoxycholic acid] to treat amyotrophic lateral sclerosis (ALS) in adults.[L42105,L43473]	Oral	Canada,EU,US
DB08834	Tauroursodeoxycholic acid	Tauroursodeoxycholic acid, also known as ursodoxicoltaurine, is a highly hydrophilic tertiary bile acid [A249070] that is produced in humans at a low concentration.[A249070] It is a taurine conjugate of [ursodeoxycholic acid] [A249065] with comparable therapeutic efficacy and safety,[A249070] but a much higher hydrophilicity.[A249060] Normally, hydrophilic bile acids regulates hydrophobic bile acids and their cytotoxic effects. Tauroursodeoxycholic acid can reduce the absorption of cholesterol in the small intestine, thereby reducing the body's intake of dietary cholesterol and the body cholesterol content.[A249080] Tauroursodeoxycholic acid is currently used in Europe to treat and prevent gallstones as a bile acid derivative. [L17040] Due to a range of its molecular properties - namely its anti-apoptotic effects - tauroursodeoxycholic acid has been examined in inflammatory metabolic diseases and neurodegenerative diseases. [A249065,A249070]	solid	Tauroursodeoxycholic acid is used to prevent and treat gallstone formation.[L17040] Tauroursodeoxycholic acid is used in combination with [phenylbutyric acid] to treat amyotrophic lateral sclerosis (ALS) in adults.[L42105,L43473]	Oral	Canada,US
DB12243	Edaravone	Edaravone is a free radical scavenger and neuroprotective agent with antioxidant properties.[A254257] It has three tautomers.[A19140] Edaravone works to scavenge reactive oxygen species, which have been implicated in neurological disorders, such as amyotrophic lateral sclerosis (ALS) and cerebral ischemia. [A19140,L44007,A254257] The intravenous formulation of edaravone was first approved in Japan in 2001 for the treatment of acute ischemic stroke.[L44007] It was later approved for the treatment of amyotrophic lateral sclerosis (ALS) in Japan and South Korea in 2015, followed by the FDA approval in May 2017 [L41810] and Health Canada approval in October 2018.[L44007] The oral suspension formulation of edaravone was approved by the FDA in May 2022 and by Health Canada in November 2022.[L44017] Edaravone was initially granted orphan designation by the European Medicines Agency on June 19, 2015 [L44007] and was under regulatory review in Europe. However, the drug manufacturer, Mitsubishi Tanabe Pharma, withdrew the Marketing Authorization Application (MAA) for edaravone from the European market on May 24, 2019, in response to the request made by the Committee for Medicinal Products for Human Use (CHMP) for a long-term study demonstrating the long-term efficacy and safety of edaravone.[L44002,L44012] Edaravone was also investigated in other disorders, such as Alzheimer's disease,[A19138] neuropathic pain, and ischemia-induced nerve injury.[A19139]	solid	Edaravone is indicated for the treatment of amyotrophic lateral sclerosis (ALS) in the US and Canada.[L41810,L43952] It is also indicated to treat acute ischemic stroke in Japan. [A19140,A19141,L44007]	Intravenous,Oral	Canada,US
DB00740	Riluzole	A glutamate antagonist (receptors, glutamate) used as an anticonvulsant (anticonvulsants) and to prolong the survival of patients with amyotrophic lateral sclerosis. Riluzole is marketed as Rilutek by Sanofi.	solid	For the treatment of amyotrophic lateral sclerosis (ALS, Lou Gehrig's Disease)	Oral	Canada,EU,US
DB12243	Edaravone	Edaravone is a free radical scavenger and neuroprotective agent with antioxidant properties.[A254257] It has three tautomers.[A19140] Edaravone works to scavenge reactive oxygen species, which have been implicated in neurological disorders, such as amyotrophic lateral sclerosis (ALS) and cerebral ischemia. [A19140,L44007,A254257] The intravenous formulation of edaravone was first approved in Japan in 2001 for the treatment of acute ischemic stroke.[L44007] It was later approved for the treatment of amyotrophic lateral sclerosis (ALS) in Japan and South Korea in 2015, followed by the FDA approval in May 2017 [L41810] and Health Canada approval in October 2018.[L44007] The oral suspension formulation of edaravone was approved by the FDA in May 2022 and by Health Canada in November 2022.[L44017] Edaravone was initially granted orphan designation by the European Medicines Agency on June 19, 2015 [L44007] and was under regulatory review in Europe. However, the drug manufacturer, Mitsubishi Tanabe Pharma, withdrew the Marketing Authorization Application (MAA) for edaravone from the European market on May 24, 2019, in response to the request made by the Committee for Medicinal Products for Human Use (CHMP) for a long-term study demonstrating the long-term efficacy and safety of edaravone.[L44002,L44012] Edaravone was also investigated in other disorders, such as Alzheimer's disease,[A19138] neuropathic pain, and ischemia-induced nerve injury.[A19139]	solid	Edaravone is indicated for the treatment of amyotrophic lateral sclerosis (ALS) in the US and Canada.[L41810,L43952] It is also indicated to treat acute ischemic stroke in Japan. [A19140,A19141,L44007]	Intravenous,Oral	Canada,US

Page 1 of 1

Figura 26 - Página com área de inserção de dados e área de resultados para interação com a DrugBank Tool

De seguida temos representadas na Tabela 19 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *DrugBank Tool*:

*Tabela 19 – Tabela com ações do ecrã DrugBank e interação com as funções da biblioteca "*Found Sequence*"*

Ação	**Interação com "*Found Sequence*" – DrugBank Tool**
1 – Enviar patologia e, com acesso à base de dados da DrugBank, tentar encontrar um ou mais fármacos que possam ajudar na minimização/prevenção da mesma.	`def found_Drug(diseases)`

Página *All in One*

A página "*All in One*" ilustrada na Figura 27 integra todas as funcionalidades numa única interface. O fluxo começa com o *upload* de um ficheiro .FASTA. O servidor Django gere este ficheiro e passa-o para a função
foundSequence, que executa todo o *pipeline* de análise de forma sequencial. À medida que os resultados de cada ferramenta são obtidos, são enviados de volta para a interface do utilizador e exibidos em secções dedicadas, proporcionando uma experiência interativa e consolidada.

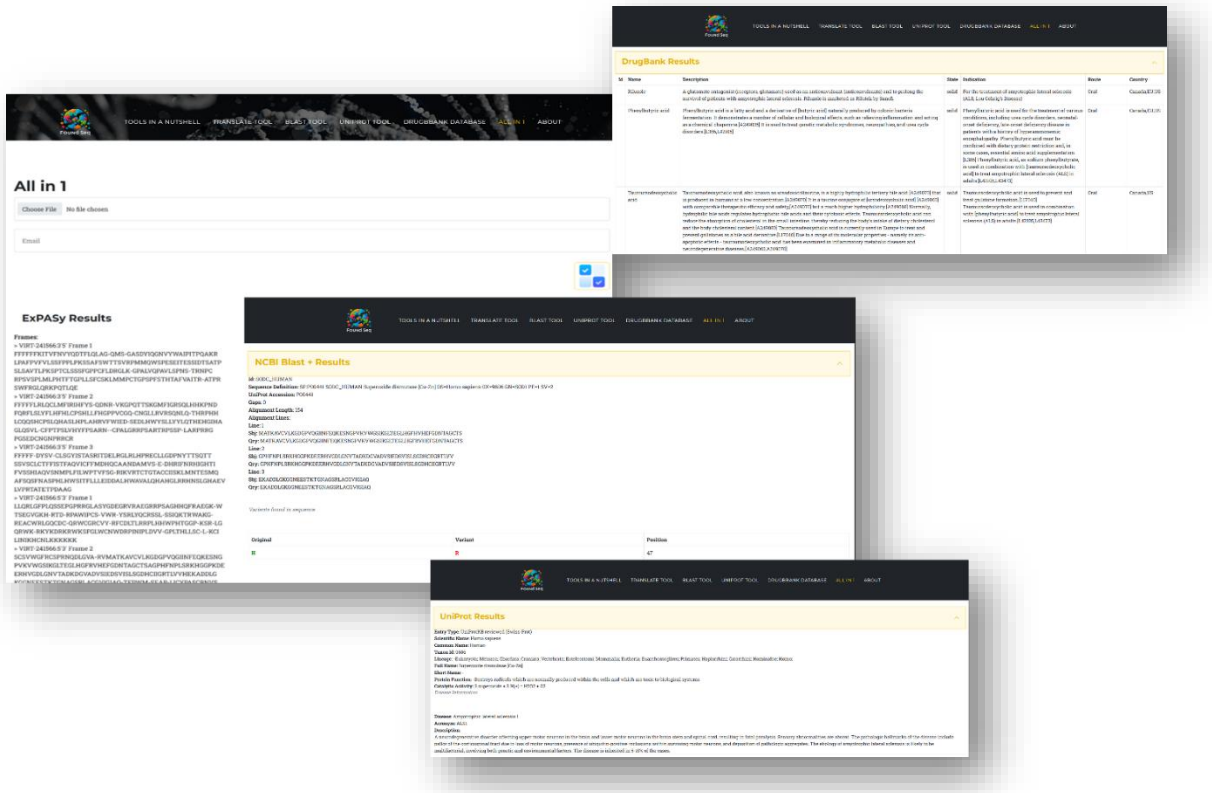


Figura 27 – Página com área de inserção de dados e área de resultados com 4 secções que mostram resultados pata todas as ferramentas, Expasy, Blast, UniProt e DrugBank

De seguida temos representadas na Tabela 20 todas as funções relacionadas com cada uma das ações que fazem parte do ecrã *All in One*:

*Tabela 20 – Tabela com ações da página All in One e interação com as funções da biblioteca "*Found Sequence*"*

Ação	**Interação com "*Found Sequence*" – *All in One***
1 – Enviar ficheiro ao Found Sequence que, após receber os *Inputs*, vai internamente fazer ordenadamente chamadas às APIs do ExPASy, Blast, UniProt e à base de dados da DrugBank.	`def foundSequence(file, email, program, matrix, alignments, scores, exp, dropoff, match_scores, gapopen, gapext, filter, seqrange, gapalign, compstats, align, stype, database, web)`

5. Avaliação da Solução

A avaliação de uma solução de *software*, especialmente no contexto académico e científico, transcende a simples verificação de funcionalidades. É imperativo demonstrar não apenas que o artefato "funciona", mas também que resolve eficazmente o problema para o qual foi concebido, acrescentando valor real em comparação com as alternativas existentes. O presente capítulo detalha o processo de avaliação da biblioteca "*Found Sequence*", focando-se em duas dimensões críticas: a validação da sua funcionalidade integrada e a medição do seu desempenho técnico.

O projeto propôs-se a desenvolver uma nova biblioteca para o Biopython, designada "Found Sequence", com o objetivo de integrar e otimizar o fluxo de trabalho entre diversas ferramentas bioinformáticas essenciais: ExPASy Translate Tool, NCBI Blast+, UniProt e DrugBank. A metodologia de avaliação foi desenhada para validar se a solução cumpre a sua promessa central: transformar um processo manual, fragmentado e moroso numa análise automatizada, coesa e eficiente.

A análise dos resultados, que se segue, sugere o seguinte: a principal constatação é o sucesso na concretização do objetivo central: a criação de uma ferramenta unificada que permite a um utilizador submeter uma sequência genética e obter, num único fluxo e local, informações cruciais que vão desde a tradução da sequência à identificação de proteínas, variantes, patologias associadas e potenciais fármacos. Este desenvolvimento responde diretamente à motivação do projeto, que surgiu da necessidade de interligar estas ferramentas, anteriormente utilizadas de forma isolada e sequencial, seja em contexto académico ou de trabalho. A solução não impede que o utilizador confronte os resultados com as ferramentas originais para obter ainda mais detalhes; contudo, a informação principal fica sintetizada e acessível num só local. Adicionalmente, o sistema foi desenhado para ser robusto, garantindo que, mesmo que uma pesquisa não necessite de percorrer todas as ferramentas (por exemplo, se não forem encontradas variantes), é apresentado um resultado final claro que informa o utilizador sobre o que foi possível obter e o porquê.

5.1. Resultados – Funcionalidade *All in One*

5.1.1. Completo

Este cenário representa o "caso ideal", em que a análise percorre com sucesso todas as etapas do *pipeline*. Na Tabela 21 são representadas cada uma das etapas e o respectivo resultado.

Tabela 21 – Fluxo de resultado completo na pesquisa

ExPASy Translate Tool	NCBI Blast+	UniProt	Drugabank
Tradução da sequência efetuada.	Encontrada semelhança com sequência de referência; Identificação da sequência e identificador no UniProt; Alinhamento e sequência com a sequência de referência; Variantes encontradas.	Identificação e funções da proteína; Patologias associadas a variantes da ferramenta anterior.	Encontrados fármacos para minimização ou controlo de patologia.

5.1.2. Sem Fármaco

Neste cenário representado na Tabela 22, a análise identifica uma patologia associada a uma variante, mas não encontra fármacos correspondentes na base de dados local do DrugBank.

Tabela 22 - Fluxo de resultado sem resultados para fármacos

ExPASy Translate Tool	NCBI Blast+	UniProt	Drugabank
Tradução da sequência efetuada.	Encontrada semelhança com sequência de referência; Identificação da sequência e identificador no UniProt; Alinhamento e sequência com a sequência de referência; Variantes encontradas.	Identificação e funções da proteína; Patologias associadas a variantes da ferramenta anterior.	Sem fármacos encontrados.

5.1.3. Sem patologia para variantes

Neste cenário representado na Tabela 23, o sistema identifica variantes na sequência, mas a consulta ao UniProt não revela nenhuma patologia diretamente associada a essas alterações específicas.

Tabela 23 – Fluxo de resultados sem patologia encontrada para as variantes

ExPASy Translate Tool	NCBI Blast+	UniProt	Drugabank
Tradução da sequência efetuada.	Encontrada semelhança com sequência de referência; Identificação da sequência e identificador no UniProt; Alinhamento e sequência com a sequência de referência; Variantes encontradas.	Identificação e funções da proteína; Uma ou mais patologias, mas não para as variantes encontradas na ferramenta anterior.	Sem resultados.

5.1.4. Sem variantes

Neste caso representado na Tabela 24, a sequência de consulta corresponde perfeitamente à sequência de referência no Blast, terminando o fluxo de análise, uma vez que não há variantes para investigar.

Tabela 24 – Fluxo de resultados sem variantes encontradas

ExPASy Translate Tool	NCBI Blast+	UniProt	Drugabank
Tradução da sequência efetuada.	Encontrada semelhança com sequência de referência; Identificação da sequência e identificador no UniProt; Alinhamento e sequência com a sequência de referência; Sem variantes.	Sem resultados.	Sem resultados.

5.1.5. Sem semelhança com sequência de referência

Este é o cenário representado na Tabela 25, em que a sequência de ADN, após traduzida, não encontra nenhuma correspondência significativa no Blast, interrompendo o processo na fase inicial.

Tabela 25 – Fluxo de resultados sem semelhança com sequência de referência

ExPASy Translate Tool	NCBI Blast+	UniProt	Drugabank
Tradução da sequência efetuada.	Sem semelhança com sequência de referência.	Sem resultados.	Sem resultados.

5.2. Medições de Desempenho

Para além da validação funcional, é crucial avaliar se a biblioteca "Found Sequence" cumpre a sua intenção de otimizar o processo de análise. Esta secção foca-se na avaliação quantitativa do desempenho técnico da solução.

O objetivo principal é medir e comparar a eficiência temporal da abordagem integrada ("*All in One*") com o fluxo de trabalho manual, que envolve a utilização separada de cada ferramenta nativa (ExPASy, NCBI Blast+, etc.). Através de testes com sequências de diferentes tamanhos, pretende-se verificar como a automatização impacta o tempo total de análise, validando a hipótese de que a nova biblioteca, apesar de poder ter um tempo de processamento computacional superior, resulta numa redução significativa do tempo global de trabalho do ponto de vista do utilizador.

Objetivo: Avaliar a eficiência técnica da biblioteca "*Found Sequence*"

Procedimento:

- Foram usadas 5 sequências de teste em formato FASTA de diferentes tamanhos para garantir a robustez da análise.
- Mediu-se o tempo total de execução de cada tipo de resultado em dois cenários:

- Uso manual das ferramentas: Simula o fluxo de trabalho de um investigador que acede a cada site (ExPASy, Blast, UniProt), insere os dados, aguarda os resultados e interpreta-os para passar à fase seguinte.
- Uso da solução integrada: Utilização da funcionalidade "*All in One*" das aplicações desenvolvidas para obter um relatório consolidado.

Resultados Esperados: A solução integrada, apesar de poder ter um tempo de processamento computacional superior devido às chamadas de API sequenciais e ao processamento de dados, resultaria numa redução significativa do tempo de análise global do ponto de vista do utilizador, eliminando as etapas manuais de transição.

5.2.1. ExPASy

A primeira análise de desempenho focou-se na ferramenta ExPASy Translate Tool. A Tabela 26 mostra o resumo de cada um dos tempos de execução para diferentes ficheiros na ferramenta nativa do ExPASy:

Tabela 26 - Tabela de resultados para a pesquisa no Expasy Translate Tool (Tool Nativa)

FASTA			ExPASy – Tool Nativa	
Ficheiro	**Tamanho**	**Sequência**	**Condições de procura de resultado**	**Tempo de tradução e análise (hh:mm:ss)**
mutseq15.FASTA	1Kb	707	• Resultado apresenta *frames* 5'3' e 3'5'; • ORF *frames* assinalados a cor diferente; • Utilizador tem que identificar visualmente qual o maior ORF; • Não lê FASTA, há necessidade de copiar a sequência.	0:00:08
mutseq7.FASTA	8Kb	7204		0:00:13
mutseq14.fasta	6Kb	5544		0:00:11
mutseq4.FASTA	3Kb	2334		0:00:08
mutseq5.FASTA	2Kb	1640		0:00:06

Posteriormente, a mesma análise foi realizada utilizando a biblioteca "Found Sequence", que recorre à API do ExPASy, com os seguintes resultados representados na Tabela 27:

*Tabela 27 – Tabela de resultados para a pesquisa no "*Found Sequence*" com recurso à API do ExPASy*

FASTA			"*Found Sequence*" – ExPASy Translate Tool API	
Ficheiro	**Tamanho**	**Sequência**	**Condições de procura de resultado**	**Tempo de tradução e análise (hh:mm:ss)**
mutseq15.FASTA	1Kb	707	• Resultado apresenta *frames* 5'3' e 3'5'; • ORF *frames* não assinalados a cor diferente; • Possibilidade de *upload* de ficheiro; • Apresenta o maior ORF ao utilizador.	0:00:06
mutseq7.FASTA	8Kb	7204		0:00:09
mutseq14.FASTA	6Kb	5544		0:00:07
mutseq4.FASTA	3Kb	2334		0:00:06
mutseq5.FASTA	2Kb	1640		0:00:05

Tendo por base as duas tabelas anteriores (Tabela 26 e 27) foi extraído um gráfico ilustrado na Figura 28 para melhor comparação dos tempos de análise da ferramenta nativa vs ao recurso à API no '*Found Sequence*':

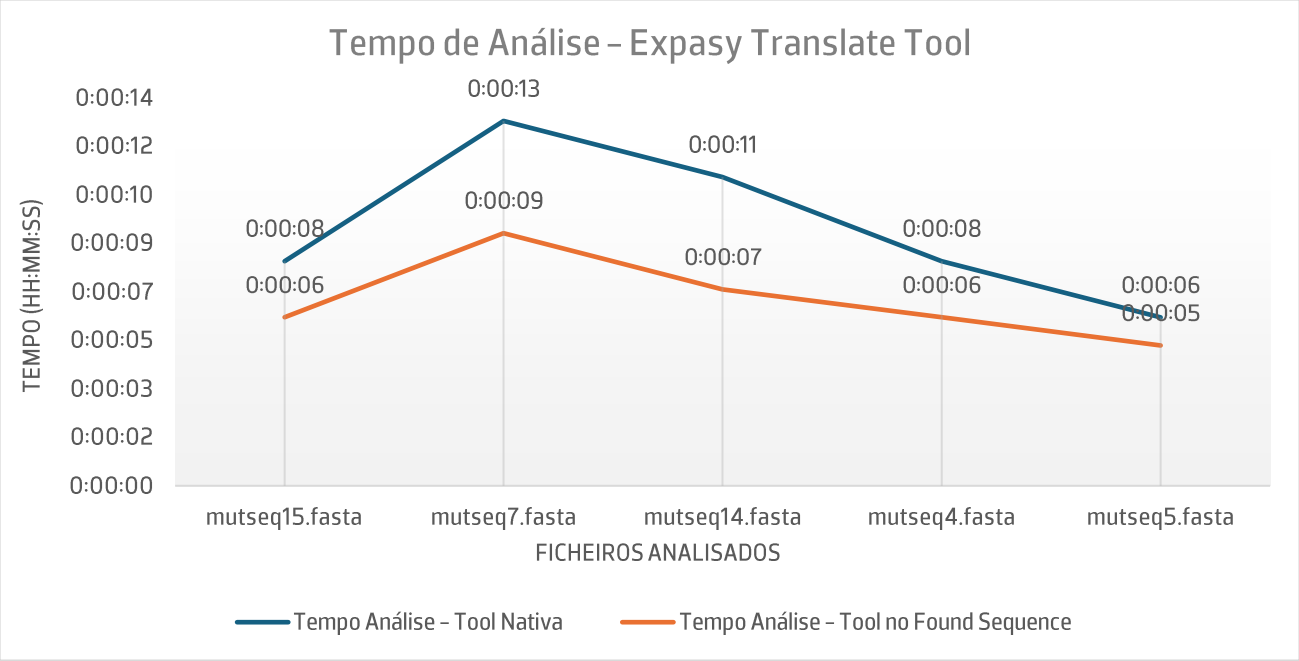


*Figura 28 – Gráfico comparativo de tempos de análise entre a tool nativa do ExPASy e o "*Found Sequence*" com recurso à API*

Conclusão:

Considerando que o objetivo principal nesta fase é a identificação da maior ORF para posterior análise no Blast, os resultados sugerem que a biblioteca "Found Sequence" proporciona uma redução no tempo de análise. Esta vantagem decorre da automatização da identificação da maior ORF, uma tarefa que, na

ferramenta nativa, teria de ser executada visualmente pelo utilizador. A aplicação desenvolvida (*desktop/web*) fornece diretamente esta informação, otimizando o fluxo de trabalho.

5.2.2. NCBI Blast+

Para a ferramenta NCBI Blast+, a análise comparativa de desempenho produziu os seguintes resultados representados na Tabela 28:

Tabela 28 – Tabela de resultados para a pesquisa no NCBI Blast (ferramenta nativa)

FASTA			NCBI Blast+ – Ferramenta Nativa	
Ficheiro	**Tamanho**	**Sequência**	**Condições de procura de resultado**	**Tempo de tradução e análise (hh:mm:ss)**
mutseq15.FASTA	1Kb	707	• Programa: blastp; • Identifica a sequência; • Identifica o tamanho; • Identifica ligação para UniProt ; • Não alinha a sequência com a referência; • Não identifica *gaps;* • Para ter a lista de variantes possíveis, é necessário consultar o UniProt.	0:01:05
mutseq7.FASTA	8Kb	7204		0:00:14
mutseq14.FASTA	6Kb	5544		0:00:12
mutseq4.FASTA	3Kb	2334		0:00:19
mutseq5.FASTA	2Kb	1640		0:00:11

Posteriormente, a mesma análise foi realizada utilizando a biblioteca "Found Sequence", que recorre à API do Blast+, com os seguintes resultados representados na Tabela 29:

*Tabela 29 – Tabela de resultados para a pesquisa no "*Found Sequence*" com recurso à API do Blast*

FASTA			"*Found Sequence*" – NCBI Blast+ API	
Ficheiro	**Tamanho**	**Sequência**	**Condições de procura de resultado**	**Tempo de tradução e análise (hh:mm:ss)**
mutseq15.FASTA	1Kb	707	• Programa: blastp; • Identifica a sequência; • Identifica o tamanho;	0:04:13
mutseq7.FASTA	8Kb	7204		0:06:51
mutseq14.FASTA	6Kb	5544		0:03:15

mutseq4.FASTA	3Kb	2334	• Identifica ligação para UniProt ; • Alinha a sequência com a referência; • Identifica *gaps;* • Identifica variantes e respetiva posição.	0:06:49
mutseq5.FASTA	2Kb	1640		0:04:56

Tendo por base as duas tabelas anteriores (Tabela 28 e 29) foi extraído um gráfico ilustrado na Figura 29 para melhor comparação dos tempos de análise da ferramenta nativa vs ao recurso à API no '*Found Sequence*':

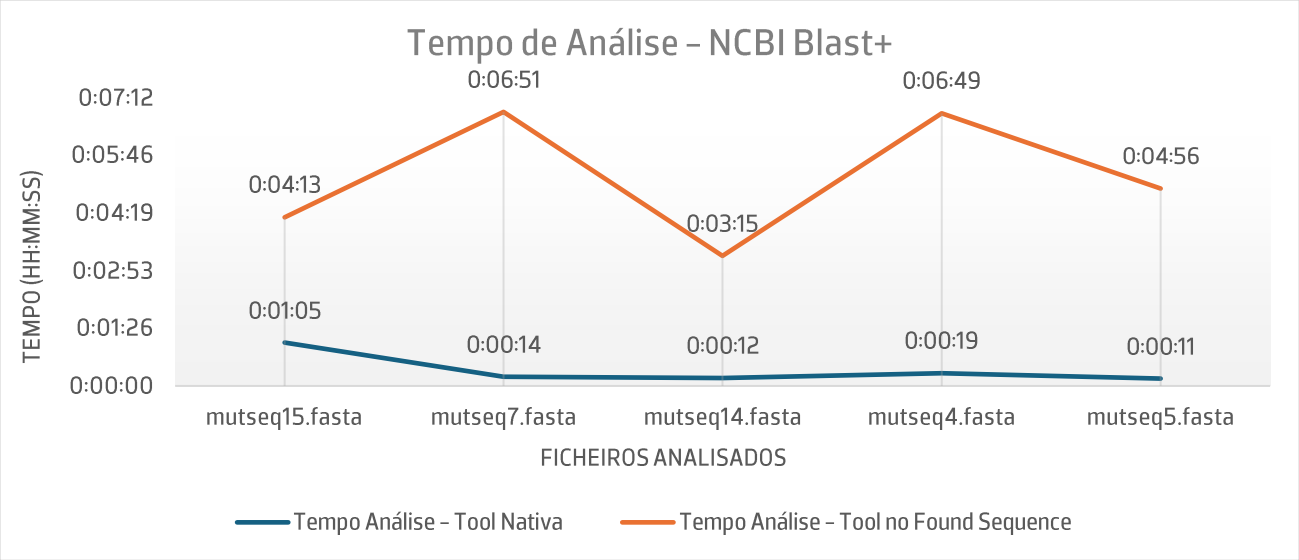


*Figura 29 – Gráfico comparativo de tempos de análise entre a tool nativa do NCBI Blast+ e o "*Found Sequence*" com recurso à API*

Conclusão:

O objetivo desta análise é identificar a sequência, alinhá-la com a sua referência e determinar as suas variantes e respetivas posições. A ferramenta nativa executa esta tarefa num tempo significativamente inferior. No entanto, não apresenta o alinhamento, as variantes ou os *gaps* nos resultados, exigindo o recurso a ferramentas adicionais para uma análise completa.

Em contrapartida, a ferramenta "*Found Sequence*", embora mais lenta, compila toda a informação necessária numa única resposta. É crucial considerar que a demora na resposta do "*Found Sequence*" está diretamente associada aos mecanismos de *throttling* implementados pela API do NCBI Blast+, que limitam a velocidade das requisições.

5.2.3. UniProt

A análise de desempenho para a ferramenta UniProt é apresentada de seguida na Tabela 30:

Tabela 30 – Tabela de resultados para a pesquisa no UniProt (ferramenta nativa)

UniProt – Ferramenta Nativa	
Condições de procura de resultado	Tempo de tradução e análise (hh:mm:ss)
• Inserir Hit Accession; • Resultados com toda a informação da proteína; • Explorar a página e a informação da proteína. No caso de ser uma pesquisa sequencial, perceber na área de "Diseases and Variants" se algo se aplica ao encontrado no NCBI Blast+.	0:00:02
	0:00:01
	0:00:02
	0:00:02
	0:00:02

Posteriormente, a mesma análise foi realizada utilizando a biblioteca "Found Sequence", que recorre à API do UniProt com os seguintes resultados representados na Tabela 31:

Tabela 31 – Tabela de resultados para a pesquisa no "Found Sequence" com recurso à API do UniProt

"*Found Sequence*" – UniProt API	
Condições de procura de resultado	Tempo de tradução e análise (hh:mm:ss)
• Inserir Hit Accession; • Se for uma pesquisa isolada, devolve informação da proteína e a lista de variantes e doenças associadas; • Se for uma pesquisa sequencial, devolve a informação da proteína e a patologia associada à variante encontrada no serviço NCBI Blast+, caso exista.	0:00:02
	0:00:01
	0:00:02
	0:00:02
	0:00:02

Tendo por base as duas tabelas anteriores (Tabela 30 e 31) foi extraído um gráfico ilustrado na Figura 30 para melhor comparação dos tempos de análise da ferramenta nativa vs ao recurso à API no '*Found Sequence*':

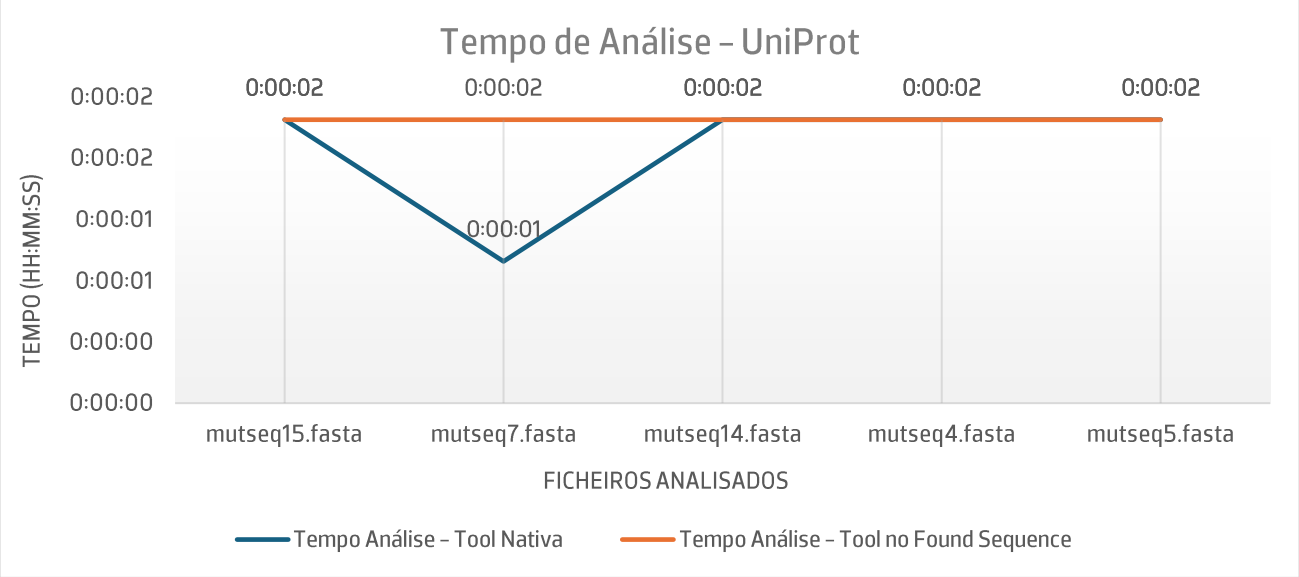


Figura 30 – Gráfico comparativo de tempos de análise entre a tool nativa do UniProt e o "Found Sequence" com recurso à API

Conclusão:

Esta análise comparou a eficiência de duas abordagens para obter dados de proteínas da base de dados UniProt através do seu número de acesso: a utilização de uma aplicação nativa e o recurso à API através do "Found Sequence". Concluiu-se que os tempos de resposta de ambos os métodos são equivalentes, não havendo ganhos ou perdas de performance significativos entre eles. A principal vantagem da abordagem via API reside, portanto, na sua integração com a biblioteca Biopython, uma funcionalidade anteriormente inexistente. Este desenvolvimento representa um avanço, facilitando a automação de pesquisas e a incorporação da recuperação de dados do UniProt em fluxos de trabalho de bioinformática de forma mais direta e programática.

5.2.4. DrugBank

A análise comparativa de tempos de resposta não foi estendida à fonte de dados DrugBank devido à metodologia de integração fundamentalmente distinta adotada.

Enquanto as outras fontes foram acedidas em tempo real através das suas respetivas APIs, a integração com a DrugBank foi implementada através de uma solução de desenvolvimento personalizada. Esta abordagem foi a alternativa escolhida para evitar os custos associados à utilização da API oficial. O

processo envolveu a análise estrutural de um ficheiro XML, a criação de uma base de dados dedicada e o desenvolvimento de uma aplicação para processar e migrar os dados para essa base de dados.
Visto que o acesso aos dados da DrugBank se dá através de uma consulta a esta base de dados local previamente processada — e não através de um pedido direto a um serviço externo —, uma comparação de tempos de resposta seria metodologicamente inconsistente e não produziria um resultado equitativo ou fidedigno.

5.2.5. *All in One*

A Tabela 32 consolida os tempos de resposta para a funcionalidade "*All in One*", que integra todas as ferramentas, em comparação com a soma dos tempos das ferramentas nativas.

Tabela 32 – Tabela de resultados para a funcionalidade All in One e comparação com as ferramentas nativas

FASTA			Ferramentas Nativas Σ (ExPASy, NCBI Blast+, DrugBank)*	*"Found Sequence"*	Tipo de Resultado
Ficheiro	Tamanho	Sequência	Tempo de resposta (hh:mm:ss)	Tempo de resposta (hh:mm:ss)	
mutseq15.FASTA	1Kb	707	0:01:15	0:02:46	Completo
mutseq7.FASTA	8Kb	7204	0:00:27	0:06:54	ExPASy+Blast
mutseq14.FASTA	6Kb	5544	0:00:25	0:02:54	ExPASy+Blast+UniProt
mutseq4.FASTA	3Kb	2334	0:00:29	0:06:48	ExPASy+Blast+UniProt
mutseq5.FASTA	2Kb	1640	0:00:17	0:04:47	ExPASy+Blast

Tendo por base a tabela anterior foi extraído um gráfico ilustrado na Figura 31 para melhor comparação dos tempos de análise do somatório das ferramentas nativas para diferentes tipos de resultados vs a funcionalidade *All in One* do '*Found Sequence*':

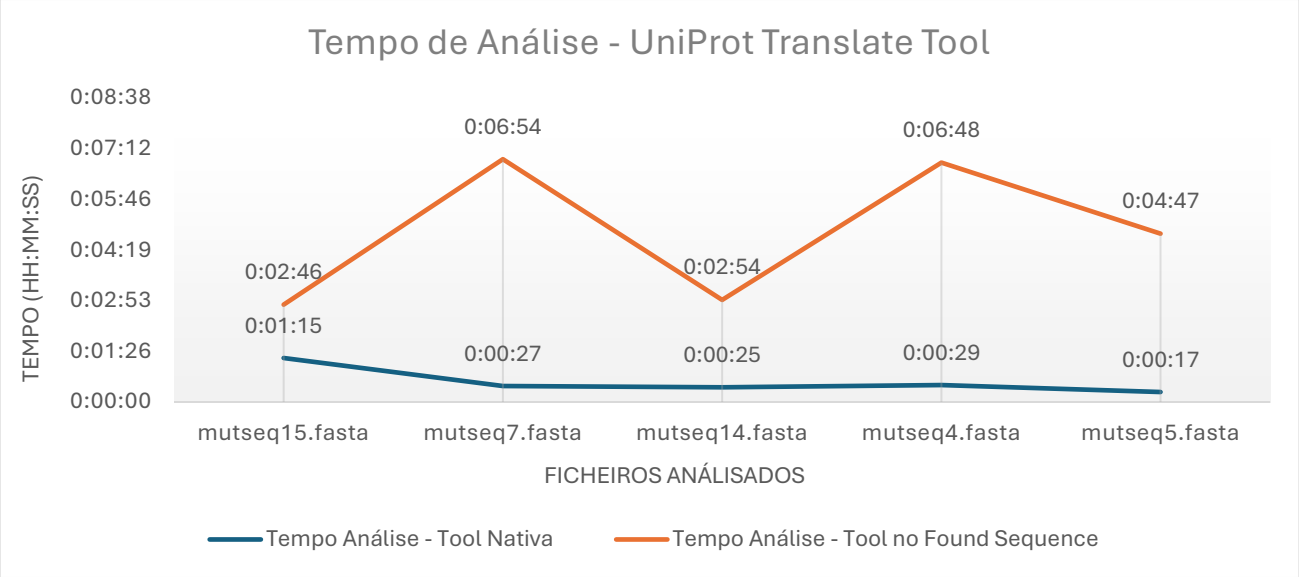


Figura 31 – Gráfico comparativo de tempos de análise entre somaório das tools nativas e a funcionalidade All in One do "Found Sequence"

Conclusão:

A análise comparativa do tempo de resposta revela que as ferramentas nativas são, individualmente, mais rápidas do que a ferramenta "*Found Sequence*". Contudo, uma avaliação focada apenas neste critério seria incompleta e levaria a uma conclusão equivocada sobre a eficiência global.

O principal diferencial reside na integralidade dos resultados:

1. Ferramentas Nativas (ex.: NCBI Blast+): Oferecem uma resposta rápida, mas fragmentada, que carece de informações essenciais como o alinhamento, as variantes e os *gaps*. A obtenção destes dados implica etapas de trabalho adicionais e o uso de outras ferramentas.
2. Ferramenta "*Found Sequence*": Apresenta um resultado consolidado, estruturado e completo, que pode ser diretamente utilizado para a geração de relatórios, eliminando a necessidade de processamento manual posterior.

Olhando aos resultados, estes sugerem que o tempo adicional exigido pelo "*Found Sequence*" é justificado pela entrega de uma análise completa. O tempo que se pouparia com as ferramentas nativas seria inevitavelmente gasto em tarefas subsequentes de pesquisa e compilação de dados, tornando o processo global menos eficiente e assim conclui-se que existe uma redução significativa do tempo de análise.

6. Discussão

6.1. Contribuições Científicas e Práticas

A principal contribuição deste projeto é a criação da "*Found Sequence*", uma solução de *software* de código aberto que automatiza e integra um fluxo de trabalho bioinformático complexo. A biblioteca agiliza a análise ao eliminar a necessidade de transcrever dados manualmente entre diferentes serviços, reduzindo o tempo e o potencial de erro humano. A agregação de resultados num relatório único oferece uma visão contextualizada que não é diretamente fornecida pelas ferramentas isoladas. Adicionalmente, o desenvolvimento de interações "de raiz" com as APIs do UniProt, ExPASy e a criação de uma base de dados local para o DrugBank enriquecem o ecossistema Biopython, preenchendo lacunas funcionais identificadas na análise inicial.

6.2. "Found Sequence" vs Ferramentas em uso Isolado

A comparação entre a utilização da biblioteca "*Found Sequence*" e o uso isolado das ferramentas bioinformáticas revela vantagens claras da abordagem integrada:

1. Agilidade no Fluxo de Trabalho: A principal mais-valia reside na automatização da passagem de dados entre as diferentes ferramentas. No uso isolado, o investigador necessitaria de manualmente copiar e colar sequências, identificadores ou nomes de patologias entre as interfaces de cada serviço (ExPASy, Blast, UniProt, DrugBank). A "*Found Sequence*", especialmente através da sua funcionalidade "*All in One*", elimina esta necessidade, reduzindo significativamente o tempo despendido e o potencial de erro humano na transcrição de dados.
2. Informação Agregada: A capacidade de gerar um relatório final que agrega os resultados de todas as ferramentas é um avanço considerável. Em vez de gerir múltiplos *outputs* de diferentes fontes, o utilizador obtém uma visão integrada e contextualizada da informação relativa à sequência mistério.
3. Fácil de usar: As interfaces gráficas *desktop* (PyQt6) e *web* (Django) foram desenvolvidas para simplificar a interação do utilizador com este conjunto de ferramentas bioinformáticas. Isto torna a análise mais acessível, especialmente para utilizadores que poderiam sentir-se intimidados pela necessidade de aprender a usar múltiplas ferramentas complexas individualmente.
4. Lógica Integrada: A biblioteca não se limita a agregar chamadas; ela incorpora lógica de processamento, como a identificação da maior ORF após a tradução pelo ExPASy, o

alinhamento e identificação de variantes nos resultados do Blast+, e a correlação de variantes com patologias no UniProt e destas com fármacos no DrugBank. Este processamento inteligente acrescenta valor para além da simples automação.

A análise inicial da biblioteca Biopython revelou que, embora existissem módulos para algumas das ferramentas (ExPASy, UniProt, Blast), estes ou não se adequavam completamente às necessidades do projeto (e.g., dados retornados ou funcionalidades específicas), encontravam-se descontinuados (Blast *online*), ou implicavam custos (ElasticBlast), justificando o desenvolvimento de novas interações via APIs públicas ou, no caso do DrugBank, através de um ficheiro XML e uma base de dados local. O desenvolvimento "de raiz" para os serviços necessários do UniProt, ExPASy e DrugBank preencheu as lacunas identificadas.

O desenvolvimento das aplicações *desktop* e *web* não só valida a funcionalidade da biblioteca "*Found Sequence*", como também demonstra a sua versatilidade e potencial de aplicação em diferentes ambientes de trabalho. A aplicação *web*, em particular, oferece vantagens em termos de acessibilidade multiplataforma e manutenção centralizada.

6.3. Desafios e Limitações

Apesar dos resultados positivos, é importante reconhecer algumas limitações. A principal limitação técnica está na interação com o DrugBank, que, devido a restrições de acesso à API, necessitou da utilização de um ficheiro XML e da criação de uma base de dados local. Embora funcional para o âmbito do projeto e através de uma licença de estudante, esta abordagem implica que os dados do DrugBank não são atualizados em tempo real, dependendo da periodicidade de atualização do ficheiro XML.

Outra consideração é a dependência das APIs públicas das ferramentas ExPASy, Blast+ e UniProt.

Este modelo de funcionamento apresenta riscos, uma vez que quaisquer alterações nestas APIs (*endpoints*, formatos de dados, limites de utilização) poderão exigir atualizações na biblioteca " Found Sequence " para manter a sua funcionalidade.

6.4. Considerações Éticas e de Sustentabilidade

O desenvolvimento e a utilização da biblioteca "*Found Sequence*" foram pautados por considerações éticas. A ferramenta foi projetada para interagir exclusivamente com bases de dados públicas e anónimas (NCBI, UniProt, ExPASy), não envolvendo o processamento de dados genéticos sensíveis ou identificáveis de pacientes. O seu propósito é académico e de investigação, funcionando como uma ferramenta de auxílio à análise e não como um sistema de diagnóstico clínico. Qualquer utilização futura em contexto clínico exigiria validações rigorosas e o cumprimento de todas as normativas de proteção de dados de saúde.

Este projeto incorpora também princípios fundamentais da sustentabilidade informática. Na vertente ambiental, a automação e otimização do fluxo de trabalho reduzem o tempo de processamento computacional, resultando num menor consumo energético. Economicamente, a escolha deliberada por APIs gratuitas em detrimento de soluções pagas como o Elastic Blast e a disponibilização da ferramenta como código aberto, posicionam a "*Found Sequence*" como uma solução de baixo custo. Finalmente, na vertente **social**, o projeto destaca-se ao democratizar o acesso a ferramentas avançadas de bioinformática. Ao ser *open-source* e acompanhado de interfaces gráficas intuitivas , promove a inclusão e capacitação na comunidade científica, desde estudantes a investigadores, alinhando-se com a filosofia colaborativa do próprio Biopython.

7. Conclusão

Este projeto de mestrado culminou com o desenvolvimento da "Found Sequence", uma nova biblioteca para Biopython que responde diretamente a um problema central na análise bioinformática: a fragmentação de ferramentas essenciais. A "Found Sequence" integra com sucesso o ExPASy Translate Tool, o NCBI Blast+, o UniProt e o DrugBank, transformando um fluxo de trabalho anteriormente manual, demorado e suscetível a erros num *pipeline* automatizado, coeso e eficiente.

O impacto deste projeto manifesta-se na sua capacidade de otimizar drasticamente o tempo de análise, desde a submissão de uma sequência genética até à identificação de potenciais fármacos. Ao automatizar a transferência de dados e agregar os resultados num relatório consolidado e de fácil interpretação, a solução não só aumenta a eficiência, como também enriquece o ecossistema Biopython, preenchendo lacunas funcionais identificadas, nomeadamente com a integração inovadora da base de dados DrugBank e a criação de interações modernas com as APIs do UniProt e ExPASy. A viabilidade e a aplicabilidade da biblioteca foram inequivocamente demonstradas através do desenvolvimento de aplicações *desktop* e *web* totalmente funcionais, que tornam estas análises complexas acessíveis a um público mais vasto, incluindo estudantes e investigadores com menos conhecimentos de programação.

O culminar deste trabalho não representa um ponto final, mas sim a base para contribuições futuras de maior alcance.

Os próximos passos estratégicos incluem:

- A expansão da biblioteca para incluir outras ferramentas bioinformáticas relevantes;
- A exploração de mecanismos para atualização mais dinâmica dos dados do DrugBank, caso o acesso à API se torne mais flexível ou mediante a utilização de outras bases de dados de fármacos com APIs abertas;
- A otimização dos algoritmos de *parsing* e processamento de dados para lidar com volumes de informação ainda maiores ou formatos de dados mais complexos;
- Submissão ao Biopython Oficial: Após a estabilidade da biblioteca "Found Sequence" que está ainda numa fase beta, o passo mais lógico e de maior impacto será preparar e submeter formalmente o módulo para inclusão no projeto principal do Biopython. Esta ação disponibilizaria a ferramenta a toda a comunidade bioinformática global, maximizando a sua utilidade e promovendo o desenvolvimento colaborativo;

- Publicação Científica: Os resultados, a metodologia de desenvolvimento e a validação de desempenho, já apresentados no congresso VII XoveTIC, constituem uma base sólida para a redação de um artigo científico a ser submetido a uma revista da especialidade. A publicação validaria o trabalho perante a comunidade científica e asseguraria a sua disseminação;
- Colaboração com Laboratórios: Procurar ativamente a colaboração com laboratórios de investigação em biologia molecular ou farmacogenómica para aplicar a "Found Sequence" a projetos de investigação reais. Esta iniciativa permitiria validar a ferramenta em cenários de uso prático e recolher *feedback* valioso para futuras iterações e melhorias.

Em suma, o '*Found Sequence*' é uma contribuição real para a ciência. É uma ferramenta de código aberto com grande potencial para ajudar os investigadores em biologia molecular a alcançar resultados de forma mais eficiente.

Referências Bibliográficas

Armstrong, J., Fiddes, I. T., Diekhans, M., Erazo, Y., Storer, J. M., Granat, A., Heitmann, M., Clawson, H., Nassar, J., Fischer, C., Haeussler, M., Trani, V. G., Zweig, A. S., Hinrichs, A. S., Spektor, R., Raney, B. J., Paten, B., Kent, W. J., & Haussler, D. (2024). The UCSC Genome Browser, 2024 update. *Nucleic Acids Research*, *52*(D1), D1189–D1196. https://doi.org/10.1093/nar/gkad1089

Babo, P., Pinho, A. J., & Alves, A. C. (2012). A Bioinformática na Biologia e nas Ciências da Saúde. *Millenium: Journal of Education, Technologies, and Health*, *43*, 87–101.

Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S., Prjibelski, A. D., Pyshkin, A. V., Sirotkin, A. V., Vyahhi, N., Tesler, G., Alekseyev, M. A., & Pevzner, P. A. (2012). SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*, *19*(5), 455–477. https://doi.org/10.1089/cmb.2012.0021

Bienert, S., Waterhouse, A., de Beer, T. A. P., Tauriello, G., Studer, G., Bordoli, L., & Schwede, T. (2017). The SWISS-MODEL repository: New features and functionality. *Nucleic Acids Research*, *45*(D1), D313–D319. https://doi.org/10.1093/nar/gkw1032

Camacho, C., Boratyn, G. M., Joukov, V., Vera Alvarez, R., & Madden, T. L. (2023). ElasticBLAST: Accelerating sequence search via cloud computing. *BMC Bioinformatics*, *24*(1), 103. https://doi.org/10.1186/s12859-023-05206-6

Chapman, B., & Chang, J. (2000). Biopython: Python tools for computational biology. *ACM SIGBIO Newsletter*, *20*(2), 15–19. https://doi.org/10.1145/360262.360268

Cingolani, P., Platts, A., Wang, L. L., Coon, M., Nguyen, T., Wang, L., Land, S. J., Lu, X., & Ruden, D. M. (2012). A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff. *Fly*, *6*(2), 80–92. https://doi.org/10.4161/fly.19695

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & de Hoon, M. J. (2009). Biopython: freely available Python tools for computational

molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163

Cunningham, F., Allen, J. E., Allen, J., Alvarez-Jarreta, J., Amode, M. R., Armean, I. M., Austine-Orimoloye, O., Azov, A. G., Bendix, F., Bennett, E., Bhai, J., Billis, K., Boddu, S., Bosc, N., Bouı́, L., Brooks, L., Cummins, C., Davidson, C., Deschamps, J., … Martin, F. J. (2024). Ensembl 2024. *Nucleic Acids Research*, *52*(D1), D984–D991. https://doi.org/10.1093/nar/gkad1049

Diniz, W. J. S., & Canduri, F. (2017). Bioinformatics: An overview and its applications. *Genetics and Molecular Research*, *16*(1). https://doi.org/10.4238/gmr16019553

Django Software Foundation. (n.d.). *Getting started with Django*. Retrieved October 5, 2024, from https://www.djangoproject.com/start/

DrugBank. (n.d.). *Clinical drug data API*. Retrieved October 26, 2023, from https://www.drugbank.com/clinical

DrugBank. (2020). *API reference*. DrugBank Help Center. Retrieved October 26, 2023, from https://docs.drugbank.com/v1/

Garrison, E., & Marth, G. (2012). *Haplotype-based variant detection from short-read sequencing* [Preprint]. arXiv. https://arxiv.org/abs/1207.3907

Harrison, P. W., Mbonea, M., & Gurdasani, D. (2024). A guide to navigating study design in human genomics. *Nature Reviews Genetics*, *25*(1), 18–34. https://doi.org/10.1038/s41576-023-00632-4

Knox, C., et al. (2024). DrugBank 6.0: The DrugBank Knowledgebase for 2024. *Nucleic Acids Research*.

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., & Phillippy, A. M. (2017). Canu: Scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Research*, *27*(5), 722–736. https://doi.org/10.1101/gr.215087.116

Langmead, B., & Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. *Nature Methods*, *9*(4), 357–359. https://doi.org/10.1038/nmeth.1923

Lehugeur, T. P., & Melo, H. C. S. (2018). Bioinformática aplicada ao desenvolvimento de novos fármacos. *Psicologia e Saúde em Debate*, *4*(Suppl. 1), 55–55.

Li, D., Liu, C.-M., Luo, R., Sadakane, K., & Lam, T.-W. (2015). MEGAHIT: An ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, *31*(10), 1674–1676. https://doi.org/10.1093/bioinformatics/btv033

Li, H. (2018). Minimap2: Pairwise alignment for nucleotide sequences. *Bioinformatics*, *34*(18), 3094–3100. https://doi.org/10.1093/bioinformatics/bty191

Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, *25*(14), 1754–1760. https://doi.org/10.1093/bioinformatics/btp324

Lukka, K. (2003). The constructive research approach. Case study research in logistics. Publications of the Turku School of Economics and Business Administration, Series B, 1(2003), 83-101

McKusick-Nathans Department of Genetic Medicine, Johns Hopkins University School of Medicine. (n.d.). *OMIM API help*. OMIM. Retrieved October 26, 2023, from https://www.omim.org/help/api

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., & DePristo, M. A. (2010). The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, *20*(9), 1297–1303. https://doi.org/10.1101/gr.107524.110

National Center for Biotechnology Information. (n.d.). *BLAST: Basic Local Alignment Search Tool*. Retrieved October 26, 2023, from https://blast.ncbi.nlm.nih.gov/Blast.cgi

Nightingale, A., Antunes, R., Alpi, E., Bursteinas, B., & Teixeira, D. (2017). The Proteins API: Accessing key integrated protein and genome information. *Nucleic Acids Research*, *45*(W1), W539–W544. https://doi.org/10.1093/nar/gkx237

Open Bioinformatics Foundation. (n.d.). *Welcome to the OBF*. Retrieved April 2, 2023, from https://www.open-bio.org/

Paysan-Lafosse, T., et al. (2023). InterPro in 2023. *Nucleic Acids Research*, *51*(D1), D418–D427. https://doi.org/10.1093/nar/gkac997

Paysan-Lafosse, T., Blum, M., Chuguransky, S., Grego, T., Pinto, B. L., Salazar, G. A., Bileschi, M. L., Bork, P., Bridge, A., Colwell, L., Gough, J., Haft, D. H., Letunić, I., Marchler-Bauer, A., Mi, H., Natale, D. A., Necci, M., Orengo, C. A., Pandurangan, A. P., … Mitchell, A. L. (2023). InterPro in 2022. *Nucleic Acids Research*, *51*(D1), D418–D427. https://doi.org/10.1093/nar/gkac993

Raney, B. J., Rosenbloom, K. R., Hnědá, A. K., Lee, B. T., Nassar, L. R., Clawson, H., Zweig, A. S., Haeussler, M., & Kent, W. J. (2024). The UCSC Genome Browser database: 2024 update. *Nucleic Acids Research*, *52*(D1), D1239–D1246. https://doi.org/10.1093/nar/gkad1084

Sayers, E. W., Bolton, E. E., Brister, J. R., Canese, K., Chan, J., Comeau, D. C., Connor, R., Funk, K., Kelly, C., Kim, S., Madej, T., Marchler-Bauer, A., Lanczy, A., Lathrop, S., Lu, Z., O'Leary, N., Phan, L., Skripvarsky, I., Karsch-Mizrachi, I., & Ostell, J. (2025). Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, *53*(D1), D17–D25. https://doi.org/10.1093/nar/gkae1057

Seemann, T. (2014). Prokka: Rapid prokaryotic genome annotation. *Bioinformatics*, *30*(14), 2068–2069. https://doi.org/10.1093/bioinformatics/btu153

Subhash, S. (2023). *Basic Local Alignment Search Tool (BLAST) of isolated Sars-Cov-2 genome using Biopython scripts at windows os* (Version 2). Zenodo. https://doi.org/10.5281/zenodo.7734807

Swiss Institute of Bioinformatics. (n.d.). *Expasy - Translate tool*. Retrieved October 26, 2023, from https://*web*.expasy.org/translate/

UniProt Consortium. (n.d.). *UniProt*. Retrieved October 26, 2023, from https://www.uniprot.org/

Van der Auwera, G. A., & O'Connor, B. D. (2020). *Genomics in the Cloud: Using Docker, GATK, and WDL in Terra*. O'Reilly Media.

Zhang, W., Chen, J., Yang, Y., Tang, Y., Shang, J., & Shen, B. (2011). A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. *PLoS ONE*, *6*(3), e17915. https://doi.org/10.1371/journal.pone.0017915

Anexos

Repositórios de código – Github

Os links abaixo dão acesso aos repositórios de código no GitHub. Estes projetos são de código aberto e estão disponíveis publicamente para consulta, estudo de código-fonte e fazer um "fork" para criação de versões próprias e experimentar. De momento, não são aceites contribuições diretas (*pull requests*) nestes repositórios.

Projeto forked do original Biopython onde foram feitas alas aterações à biblioteca

- https://github.com/patanyska/biopython_foundseq
- Clone: https://github.com/patanyska/biopython_foundseq.git

Projeto da aplicação *desktop*

- https://github.com/patanyska/ui_foundseq
- Clone: https://github.com/patanyska/ui_foundseq.git

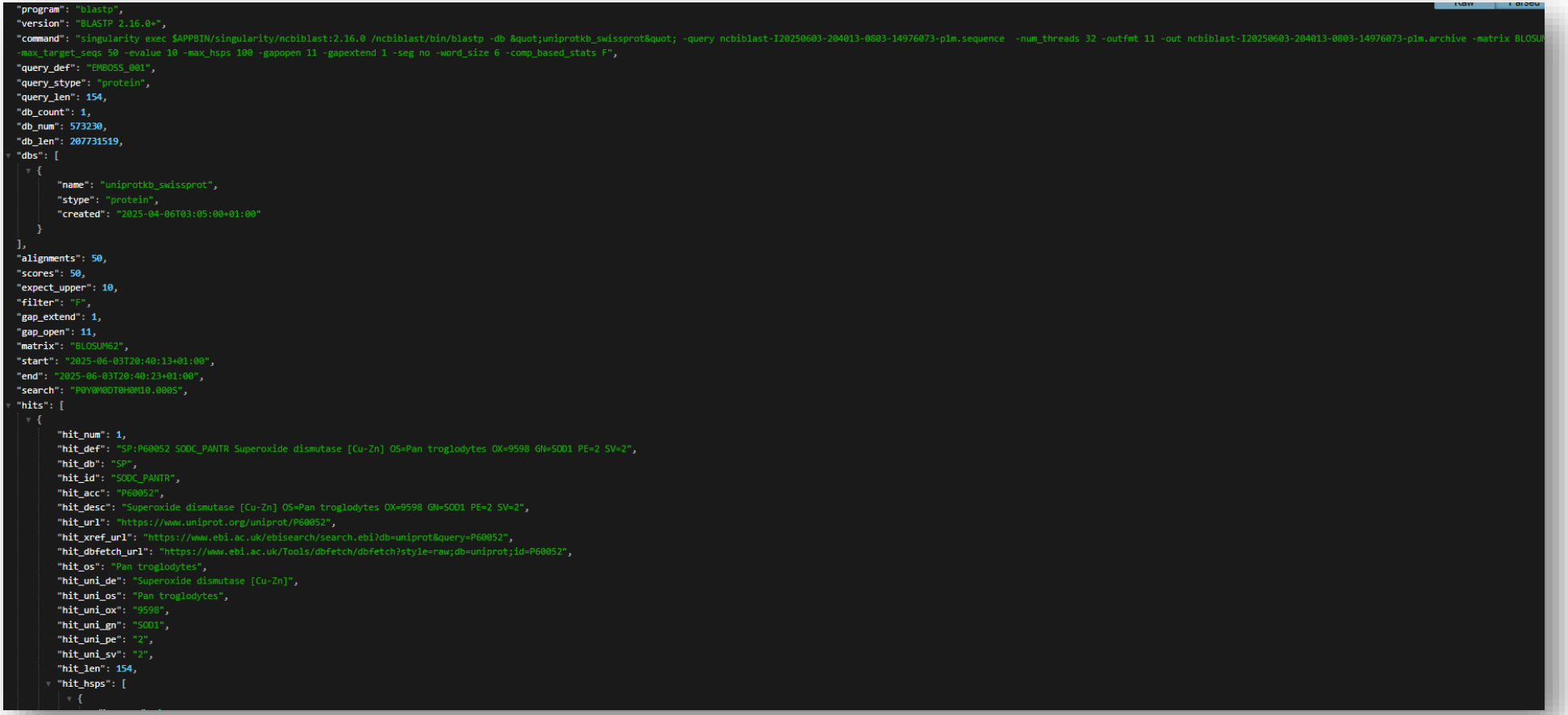
Projeto da aplicação *web*

- https://github.com/patanyska/*web*_foundseq
- Clone: https://github.com/patanyska/*web*_foundseq.git

Trecho de Ficheiro FASTA enviado pela API ExPASy Translate Tool

```
# DNA_sequence
cccactggccggtataaagcaccgtgaccctcaggtgacgcaccagggccggctgccgtcggggacagggctttccatagccatggcccagcagtggagcctccaaaggctcgcaggccgccatccgcaggacagctatgaggacagcacccagtccagcatcttcacctacaccaacagcaactccaccagaggccccttcgaaggcccgaattaccacatcgctcccagatggg
tgtaccacctcaccagtgtctggatgatctttgtggtcattgcatccgtcttcacaaatgggcttgtgctggcggcccaccatgaagttcaagaagctgcgccacccgctgaactggatcctggtgaacctggcggtcgctgacctggcagagaccgtcatcgccagcactatcagcgttgtgaaccaggtctatggctacttcgtgctgggccaccctatgtgtgtcctggagggc
tacaccgtctccctgtgtgtgggatcacaggtctctggtctctggccatcatttcctgggagagatggatggtggtctgcaagccctttggcaatgtgagatttgatgccaagctggccatcgtgggcattgccttctcctggatctgggctgctgtggacagccccgcccatctttggttggagcaggtactggccccacggcctgaagacttcatgcggcccagacgtgttcag
cggcagctcgtacccc gggggtgcagtcttacatgattgtcctcatggtcacctgctgcatcaccccactcagcatcatcgtgctctgctacctccaagtgtggctggccatccgagcggtggcaaagcagcagaaagagtctgaatccacccagaaggcagagaaggaagtgacgcgcatggtggtggtgatggtcctggcattctgcttctgctggggaccatacgccttcttcg
catgctttgctgctgccaaccctggctacccctccaccctttgatggctgccctgccggccttctttgccaaaagtgccactatctacaaccccgttatctatgtctttatgaaccggcagtttcgaaactgcatcttgcagcttttcgggaagaaggttgacgatggctctgaactctccagcgcctccaaaacggaggtctcatctgtgtcctcggtatcgcctgcatgaggt
ctgcctcctaccatcccgcccaccggggctttggccacctctccttttccccctccttctctccatccctgtaaaataaatgtaatttatctttgccaaaaccaacaaagtcacagaggctttcactgcagtgtgggaccacctgagcctctgcgtgtgcaggcactgggtctcgagagggtgcaaggggggataaagaggagagagcgcttcatagactttaagttttcccgagcctc
atgtctaccgatggcgtgaaaggatcctggcaaaacagaagtgtgaggcaggtgggcgtctatatccatttcaccaggctggtggttacataatcggcaagcaagagctgtggaggggcttgctggatgccctcagcacccaggaggagggaggggagctagcaagctaaggcaggtggccctcctggccccttaaggtccatctgctggaggcccagagtccttggagtacagtct
acacctggaggggacccattcctgccagtctgtggcagggatggcgcgccacctctgccaggccaggaccccaagcccgatcagcatcagcatggtgcaggtgcacaggcgtgagctgatcagtgacgaggggcaggcacacaaggtggagacaaagaccaagaggacggttgccagtgagaggcgcggactcaggaacttgaacaacatctgcgggggacggctttggaggtgct
ccgctgcctccagttgggtgacttgctgtagcatctccagcttggatattcggctcttgaaggtctccgtgatctcctgcaggagacgaaaatgcacgcaccagaagtca
#
                        5'3' Frame 1


                        PTGRYKAP-PSGDAPGPAAVGDRAFHSHGPAVEPPKARRPPSAGQL-GQHPVQHLHLHQQQLHQRPLRRPELPHRSQMGVPPHQCLDDLCGHCIRLHKWACAGGHHEVQEAAPPAELDPGEPGGR-PGRDRHRQHYQRCEPGLWLLRAGPPYVCPGGLHRLPVWDHRSLVSGHHFLGEMDGGLQALWQCEI-
CQAGHRGHCLLLDLGCCVDSPAHLWLEQVLAPRPEDFMRPRRVQRQLVPRGAVLHDCPHGHLLHHPTQHHRALLPPSVAGHPSGGKAAERV-IHPEGREGSDAHGGGDGPGILLLLGTIRLLRMLCCCQPWLPLPPFDGCPAGLLCQKCHYLQPRYLCLYEPAVSKLHLAAFREEG-RWL-TLQRLQNGGLICVLGIACMRSASYPSRPPGLWPPLLSPSFSIPVK-M-
FIFAKTNKVTEAFTAVWDHLSLCVCRHWVSRGCKGDKEERALHRL-VFPSLMSTDGVKGSWQNRSVRQVGVYIHFTRLVVT-SASKSCGGACWMPSAPRRREGASKLRQVALLAP-
GPSAGGPESLEYSLHLEGTHSCQSVAGMARHLCQARTPSPISISMVQVHRRELISDEGQAHKVETKTKRTVASERRGLRNLNNICGGRLWRCSAASSWVTCCSISSLDIRLLKVSVISCRRRKCTHQKS

#
                        5'3' Frame 2



PLAGIKHRDPQVTHQGRLPSGTGLSIAMAQQWSLQRLAGRHPQDSYEDSTQSSIFTYTNSNSTRGPFEGPNYHIAPRWVYHLTSVWMIFVVIASVFTNGLVLAATMKFKKLRHPLNWILVNLAVADLAETVIASTISVVNQVYGYFVLGHPMCVLEGYTVSLCGITGLWSLAIISWERWMVVCKPFGNVRFDAKLAIVGIAFSWIWAAVWTAPPIFGWSRYWPHGLKTSCGPDVFS
GSSYPGVQSYMIVLMVTCCITPLSIIVLCYLQVWLAIRAVAKQQKESESTQKAEKEVTRMVVVMVLAFCFCWGPYAFFACFAAANPGYPFHPLMAALPAFFAKSATIYNPVIYVFMNRQFRNCILQLFGKKVDDGSELSSASKTEVSSVSSVSPA-GLPPTHPAHRGFGHLSFPPPSPSL-NKCNLSLPKPTKSQRLSLQCGTT-
ASACAGTGSREGARGIKRRERFIDFKFSRASCLPMA-KDPGKTEV-GRWASISISPGWWLHNRQARAVEGLAGCPQHPGGGRELAS-GRWPSWPLKVHLLEAQSPWSTVYTWRGPIPASLWQGWRATSARPGPQARSASAWCRCTGVS-SVTRGRHTRWRQRPRGRLPVRGADSGT-TTSAGDGFGGAPLPPVG-LAVASPAWIFGS-RSP-SPAGDENARTRS

#
                        5'3' Frame 3


                        HWPV-STVTLR-RTRAGCRRGQGFP-PWPSSGASKGSQAAIRRTAMRTAPSPASSPTPTATPPEAPSKARITTSLPDGCTTSPVSG-SLWSLHPSSQMGLCWRPP-SSRSCATR-TGSW-TWRSLTWQRPSSPALSAL-TRSMATSCWATLCVSWRATPSPCVGSQVSGLWPSFPGRDGWWSASPLAM-
DLMPSWPSWALPSPGSGLLCGQPRPSLVGAGTGPTA-RLHAAQTCSAAARTPGCSLT-LSSWSPAASPHSASSCSATSKCGWPSERWQSSRKSLNPPRRQRRK-RAWWW-WSWHSASAGDHTPSSHALLLPTLATPSTL-WLPCRPSLPKVPLSTTPLSMSL-
TGSFETASCSFSGRRLTMALNSPAPPKRRSHLCPRYRLHEVCLLPIPPTGALATSPFPLLLHPCKINVIYLCQNQQSHRGFHCSVGPPEPLRVQALGLERVQGG-RGESAS-TLSFPEPHVYRWRERILAKQKCEAGGRLYPFHQAGGYIIGKQELWRGLLDALSTQEEGGS-QAKAGGPPGPLRSICWRPRVLGVQSTPGGDPFLPVCGRDGAPPLPGQDPKPDQHQHGAGAQA-
ADQ-RGAGTQGGDKDQEDGCQ-EARTQELEQHLRGTALEVLRCLQLGDLL-HLQLGYSALEGLRDLLQETKMHAPEV
```

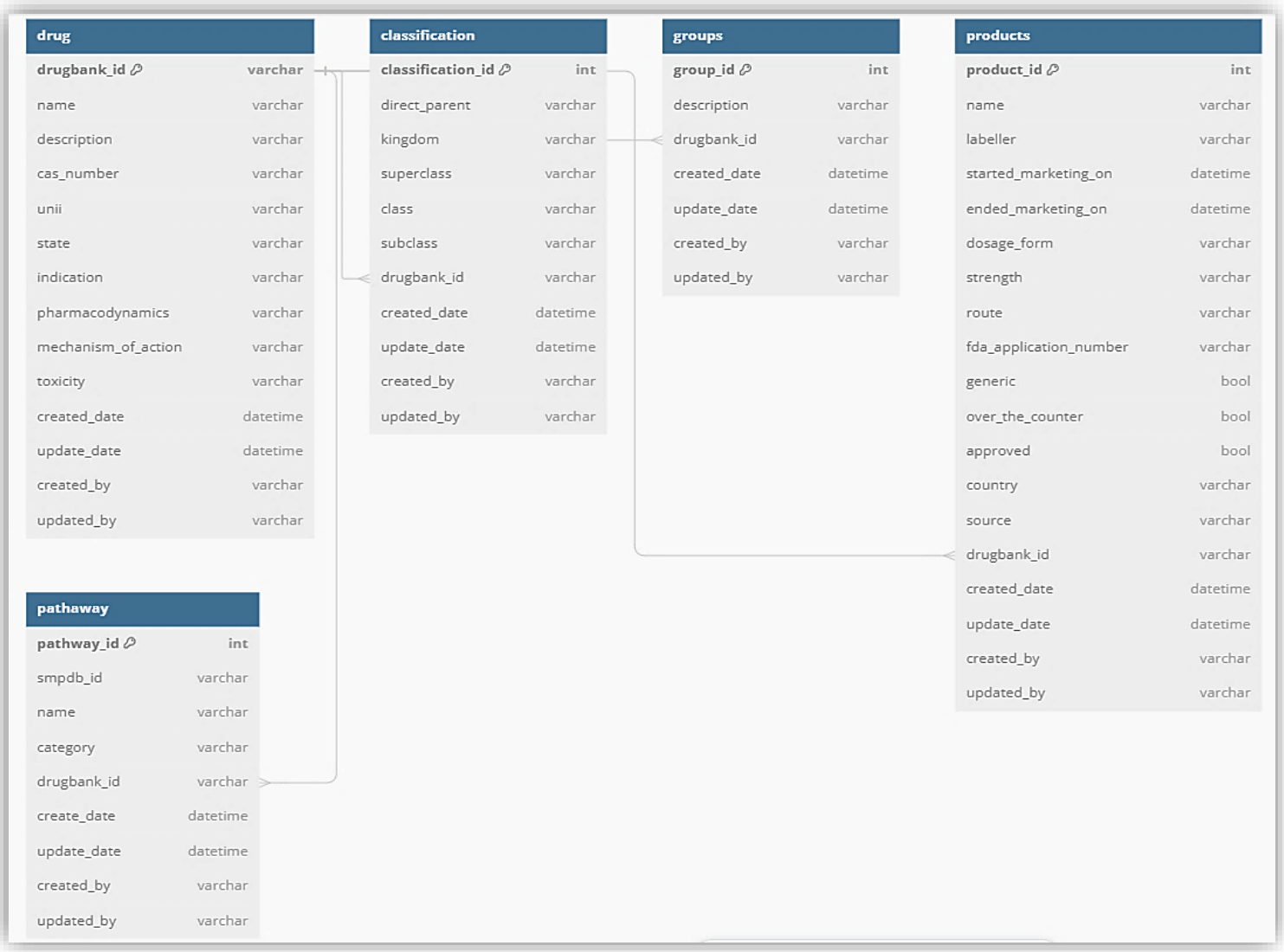
Trecho de JSON enviado pela API NCBI Blast+



Trecho de JSON enviado pela API UniProt – Proteína


```
"program": "blastp",
"version": "BLASTP 2.16.0+",
"command": "singularity exec $APPBIN/singularity/ncbiblast:2.16.0 /ncbiblast/bin/blastp -db &quot;uniprotkb_swissprot&quot; -query ncbiblast-I20250414-163051-0230-30
"query_def": "EMBOSS_001",
"query_stype": "protein",
"query_len": 364,
"db_count": 1,
"db_num": 572970,
"db_len": 207621817,
"dbs": [
    {
        "name": "uniprotkb_swissprot",
        "stype": "protein",
        "created": "2025-02-13T02:15:00+00:00"
    }
],
"alignments": 50,
"scores": 50,
"expect_upper": 10.0,
"filter": "F",
"gap_extend": 1.0,
"gap_open": 11.0,
"matrix": "BLOSUM62",
"start": "2025-04-14T16:30:51+01:00",
"end": "2025-04-14T16:31:00+01:00",
"search": "P0Y0M0DT0H0M9.000S",
"hits": [
    {
        "hit_num": 1,
        "hit_def": "SP:P04001 OPSG_HUMAN Medium-wave-sensitive opsin 1 OS=Homo sapiens OX=9606 GN=OPN1MW PE=1 SV=1",
        "hit_db": "SP",
        "hit_id": "OPSG_HUMAN",
        "hit_acc": "P04001",
        "hit_desc": "Medium-wave-sensitive opsin 1 OS=Homo sapiens OX=9606 GN=OPN1MW PE=1 SV=1",
        "hit_url": "https://www.uniprot.org/uniprot/P04001",
```

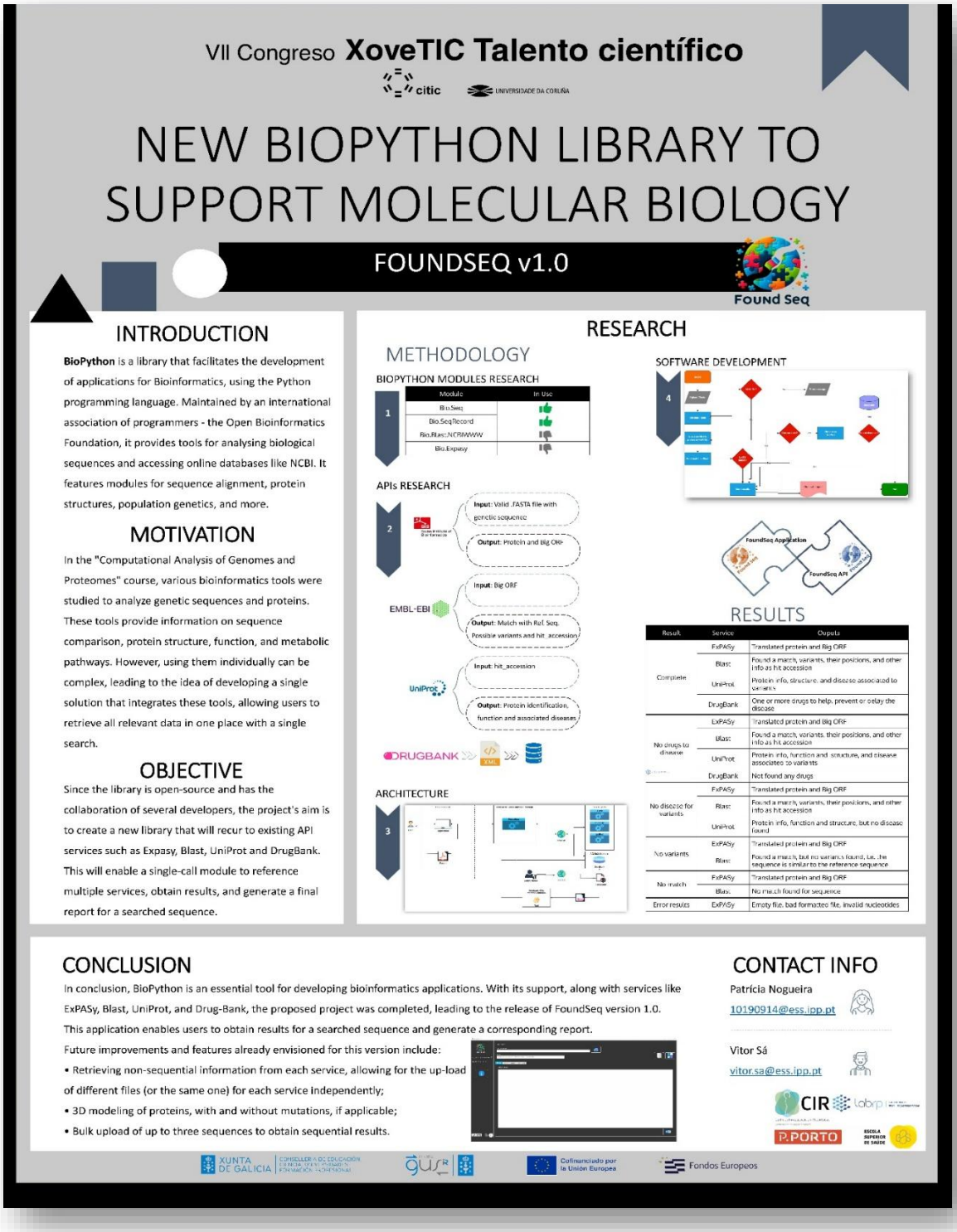
Diagrama Relacional – Base de Dados DrugBank



Poster de apresentação no congresso XOVETIC

No passado outubro de 2024 , foi realizada a conferência VII outubro de 2024, e foi apresentado o projeto desenvolvido, e foi elaborado um poster para ser desenvolvido durante a apresentação.

Mais detalhes acerca da conferência aqui.



Techos de *reports* exportados da aplicação *web*

Expasy

Frames 3'5' and 5'3':

> VIRT-1252797:3'5' Frame 1
FFFFFFKITVFNVYQDTFLQLAG-QMS-GASDYIQGNVYWAIPITPQAKR
LPAFPVFVLSSFPPLPKSSAFSWTTSVRPMMQWSPESEITESSIDTSATP
SLSAVTLPKSPTCLSSSFGPPCFLDRGLK-GPALVQPAVLSPNS-TRNPC
RPSVSPLMLPHTFTGPLLSFCSKLMMPCTGPSPFSTHTAFVAITR-ATPR
SWFRGLQRKPQTLQE
> VIRT-1252797:3'5' Frame 2
FFFFFLRLQCLMFIRIHFYS-QDNR-VKGPQTTSKGMFIGRSQLHHKPND
FQRFLSLYFLHFHLCPSHLLFHGPPVCGQ-CNGLLRVRSQNLQ-THRPHH
LCQQSHCPSLQHASLHPLAHRVFWIED-SEDLHWYSLLYYLQTHEHGIHA
GLQSVL-CFPTPSLVHYFPSARN--CPALGRRPSARTRPSSP-LARPRRG
PGSEDCNGNPRRCR
> VIRT-1252797:3'5' Frame 3
FFFFF-DYSV-CLSGYISTASRITDELRGLRLHPRECLLGDPNYTTSQTT
SSVSCLCTFFISTFAQVICFFMDHQCAANDAMVS-E-DHRIFNRHIGHTI
FVSSHIAQVSNMPLFILWPTVFSG-RIKVRTCTGTACCIISKLMNTESMQ
AFSQSFNASPHLHWSITFLLLEIDDALHWAVALQHAHGLRRHNSLGHAEV
LVPRTATETPDAAG
> VIRT-1252797:5'3' Frame 1
LLQRLGFPLQSSEPGPRRGLASYGDEGRVRAEGRRPSAGHHQFRAEGK-W
TSEGVGKH-RTD-RPAWIPCS-VWR-YSRLYQCRSSL-SSIQKTRWAKG-
REACWRLGQCDC-QRWCGRCVY-RFCDLTLRRPLHHWPHTGGP-KSR-LG
QRWK-RKYKDRKRWKSFGLWCNWDRPINIPLDVV-GPLTHLLSC-L-KCI
LINIKHCNLKKKKKK
> VIRT-1252797:5'3' Frame 2
SCSVWGFRCSPRNQDLGVA-RVMATKAVCVLKGDGPVQGIINFEQKESNG
PVKVWGSIKGLTEGLHGFRVHEFGDNTAGCTSAGPHFNPLSRKHGGPKDE
ERHVGDLGNVTADKDGVADVSIEDSVISLSGDHCIIGRTLVVHEKADDLG
KGGNEESTKTGNAGSRLACGVIGIAQ-TFPWM-SEAP-LICYPASCRNVS
--TLNTVILKKKKK
> VIRT-1252797:5'3' Frame 3
PAASGVSVAVLGTRTSAWPSELWRRRPCAC-RATAQCRASSISSRRKVMD
Q-RCGEALKD-LKACMDSVFMSLEIIQQAVPVQVLTLILYPENTVGQRMK
RGMLETWAM-LLTKMVWPMCLLKIL-SHSQETIASLAAHWWSMKKQMTWA
KVEMKKVQRQETLEVVWLVV-LGSPNKHSLGCSLRPLNSSVILLAVEMYP
DKH-TL-S-KKKKK

Biggest Open Reading Frame:

MATKAVCVLKGDGPVQGIINFEQKESNGPVKVWGSIKGLTEGLHGFRVHEFGDNTAGCTSAGPHFNPLSRKHGGPK
DEERHVGDLGNVTADKDGVADVSI
EDSVISLSGDHCIIGRTLVVHEKADDLGKGGNEESTKTGNAGSRLACGVIGIAQ

NCBI Blast +

[Sequence Main Info]

ID: SODC_HUMAN
Definition: SP:P00441 SODC_HUMAN Superoxide dismutase [Cu-Zn] OS=Homo sapiens OX=9606 GN=SOD1 PE=1 SV=2
Accession: P00441
Gaps: 0
Alignment Length: 154

[Sequence Alignment]

```
Line 1:
Sbj:  MATKAVCVLKGDGPVQGIINFEQKESNGPVKVWGSIKGLTEGLHGFHVHEFGDNTAGCTS
Qry:  MATKAVCVLKGDGPVQGIINFEQKESNGPVKVWGSIKGLTEGLHGFRVHEFGDNTAGCTS

Line 2:
Sbj:  GPHFNPLSRKHGGPKDEERHVGDLGNVTADKDGVADVSIEDSVISLSGDHCIIGRTLVV
Qry:  GPHFNPLSRKHGGPKDEERHVGDLGNVTADKDGVADVSIEDSVISLSGDHCIIGRTLVV

Line 3:
Sbj:  EKADDLGKGGNEESTKTGNAGSRLACGVIGIAQ
Qry:  EKADDLGKGGNEESTKTGNAGSRLACGVIGIAQ
```

[Variants Table]

Original	Variant	Position
H	R	47

UniProt

[Protein Main Info]

Entry Type: UniProtKB reviewed (Swiss-Prot)
Scientific Name: Homo sapiens
Common Name: Human
Taxon Id: 9606
Lineage: Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo;
Full Name: Superoxide dismutase [Cu-Zn]
Short Name: -
Protein Function: Destroys radicals which are normally produced within the cells and which are toxic to biological
systems
Catalytic Activity: 2 superoxide + 2 H(+) = H2O2 + O2

[Diseases and Variants]

Disease: Amyotrophic lateral sclerosis 1

Description: A neurodegenerative disorder affecting upper motor neurons in the brain and lower motor neurons in
the brain stem and spinal cord, resulting in fatal paralysis. Sensory abnormalities are absent. The pathologic hallmarks of the disease include pallor of the corticospinal tract due to loss of motor neurons, presence of ubiquitin-positive inclusions within surviving motor neurons, and deposition of pathologic aggregates. The etiology of amyotrophic lateral sclerosis is likely to be multifactorial, involving both genetic and environmental factors. The disease is inherited in 5-10% of the cases.

Type	Position	Original	Variant
Natural variant	5	A	S
Natural variant	5	A	T
Natural variant	5	A	V
Natural variant	7	C	F
Natural variant	8	V	E
Natural variant	9	L	Q
Natural variant	9	L	V
Natural variant	13	G	R
Natural variant	15	V	G
Natural variant	15	V	M
Natural variant	17	G	S
Natural variant	21	F	C
Natural variant	22	E	G
Natural variant	22	E	K

Drugbank

Name	State	Route	Country
Riluzole	solid	Oral	Canada,EU,US
Description			
A glutamate antagonist (receptors, glutamate) used as an anticonvulsant (anticonvulsants) and to prolong the survival of patients with amyotrophic lateral sclerosis. Riluzole is marketed as Rilutek by Sanofi.			
Indication			
For the treatment of amyotrophic lateral sclerosis (ALS, Lou Gehrig's Disease)			

Name	State	Route	Country
Phenylbutyric acid	solid	Oral	Canada,EU,US
Description			
Phenylbutyric acid is a fatty acid and a derivative of [butyric acid] naturally produced by colonic bacteria fermentation. It demonstrates a number of cellular and biological effects, such as relieving inflammation and acting as a chemical chaperone.[A249035] It is used to treat genetic metabolic syndromes, neuropathies, and urea cycle disorders.[L386,L42105]			
Indication			
Phenylbutyric acid is used for the treatment of various conditions, including urea cycle disorders, neonatal-onset deficiency, late-onset deficiency disease in patients with a history of hyperammonemic encephalopathy. Phenylbutyric acid must be combined with dietary protein restriction and, in some cases, essential amino acid supplementation.[L386] Phenylbutyric acid, as sodium phenylbutyrate, is used in combination with [tauroursodeoxycholic acid] to treat amyotrophic lateral sclerosis (ALS) in adults.[L42105,L43473]			

Name	State	Route	Country
Tauroursodeoxycholic acid	solid	Oral	Canada,US
Description			

All in One

Frames 3'5' and 5'3':

```
> VIRT-1252797:3'5' Frame 1
FFFFFFKITVFNVYQDTFLQLAG-QMS-GASDYIQGNVYWAIPITPQAKR
LPAFPVFVLSSFPPLPKSSAFSWTTSVRPMMQWSPSEITESSIDTSATP
SLSAVTLPKSPTCLSSSFGPPCFLDRGLK-GPALVQPAVLSPNS-TRNPC
RPSVSPLMLPHTFTGPLLSFCSKLMMPCTGPSPFSTHTAFVAITR-ATPR
SWFRGLQRKPQTLQE
> VIRT-1252797:3'5' Frame 2
FFFFFLRLQCLMFIRIHFYS-QDNR-VKGPQTTSKGMFIGRSQLHHKPND
FQRFLSLYFLHFHLCPSHLLFHGPPVCGQ-CNGLLRVRSQNLQ-THRPHH
LCQQSHCPSLQHASLHPLAHRVFWIED-SEDLHWYSLLYYLQTHEHGIHA
GLQSVL-CFPTPSLVHYFPSARN--CPALGRRPSARTRPSSP-LARPRRG
PGSEDCNGNPRRCR
> VIRT-1252797:3'5' Frame 3
FFFFF-DYSV-CLSGYISTASRITDELRGLRLHPRECLLGDPNYTTSQTT
SSVSCLCTFPISTFAQVICFFMDHQCAANDAMVS-E-DHRIFNRHIGHTI
FVSSHIAQVSNMPLFILWPTVFSG-RIKVRTCTGTACCISKLMNTESMQ
AFSQSFNASPHLHWSITFLLLEIDDALHWAVALQHAHGLRRHNSLGHAEV
LVPRTATETPDAAG
> VIRT-1252797:5'3' Frame 1
LLQRLGFPLQSSEPGPRRGLASYGDEGRVRAEGRRPSAGHHQFRAEGK-W
TSEGVGKH-RTD-RPAWIPCS-VWR-YSRLYQCRSSL-SSIQKTRWAKG-
REACWRLGQCDC-QRWCGRCVY-RFCDLTLRRPLHHWPHTGGP-KSR-LG
QRWK-RKYKDRKRWKSFGLWCNWDRPINIPLDVV-GPLTHLLSC-L-KCI
LINIKHCNLKKKKKK
> VIRT-1252797:5'3' Frame 2
SCSVWGFRCSPRNQDLGVA-RVMATKAVCVLKGDGPVQGIINFEQKESNG
PVKVWGSIKGLTEGLHGFRVHEFGDNTAGCTSAGPHFNPLSRKHGGPKDE
ERHVGDLGNVTADKDGVADVSIEDSVISLSGDHCIIGRTLVVHEKADDLG
KGGNEESTKTGNAGSRLACGVIGIAQ-TFPWM-SEAP-LICYPASCRNVS
--TLNTVILKKKKK
> VIRT-1252797:5'3' Frame 3
PAASGVSVAVLGTRTSAWPSELWRRRPCAC-RATAQCRASSISSRRKVMD
Q-RCGEALKD-LKACMDSVFMSLEIIQQAVPVQVLTLILYPENTVGQRMK
RGMLETWAM-LLTKMVWPMCLLKIL-SHSQETIASLAAHWWSMKKQMTWA
KVEMKKVQRQETLEVVWLVV-LGSPNKHSLGCSLRPLNSSVILLAVEMYP
DKH-TL-S-KKKKK
```

Biggest Open Reading Frame:

```
MATKAVCVLKGDGPVQGIINFEQKESNGPVKVWGSIKGLTEGLHGFRVHEFGDNTAGCTSAGPHFNPLSRKHGGPK
DEERHVGDLGNVTADKDGVADVSI
EDSVISLSGDHCIIGRTLVVHEKADDLGKGGNEESTKTGNAGSRLACGVIGIAQ
```

BLAST+ Results

[Sequence Main Info]

Id: SODC_HUMAN
Sequence Definition: SP:P00441 SODC_HUMAN Superoxide dismutase [Cu-Zn] OS=Homo sapiens OX=9606 GN=SOD1 PE=1 SV=2
UniProt Accession: P00441
Gaps: 0
Alignment Length: 154

[Sequence Alignment]

```
Line 1:
Sbj:  MATKAVCVLKGDGPVQGIINFEQKESNGPVKVWGSIKGLTEGLHGFHVHEFGDNTAGCTS
Qry:  MATKAVCVLKGDGPVQGIINFEQKESNGPVKVWGSIKGLTEGLHGFRVHEFGDNTAGCTS

Line 2:
Sbj:  GPHFNPLSRKHGGPKDEERHVGDLGNVTADKDGVADVSIEDSVISLSGDHCIIGRTLVV
Qry:  GPHFNPLSRKHGGPKDEERHVGDLGNVTADKDGVADVSIEDSVISLSGDHCIIGRTLVV

Line 3:
Sbj:  EKADDLGKGGNEESTKTGNAGSRLACGVIGIAQ
Qry:  EKADDLGKGGNEESTKTGNAGSRLACGVIGIAQ
```

[Variants Table]

Original	Variant	Position
H	R	47

UniProt Results

[Protein Main Info]

Entry Type: UniProtKB reviewed (Swiss-Prot)
Scientific Name: Homo sapiens
Common Name: Human
Taxon Id: 9606
Lineage: -Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo;
Full Name: Superoxide dismutase [Cu-Zn]
Short Name: -
Protein Function: -Destroys radicals which are normally produced within the cells and which are toxic to biological systems
Catalytic Activity: 2 superoxide + 2 H(+) = H2O2 + O2

[Diseases & Variants]

Disease: Amyotrophic lateral sclerosis 1
Description: A neurodegenerative disorder affecting upper motor neurons in the brain and lower motor neurons in the brain stem and spinal cord, resulting in fatal paralysis. Sensory abnormalities are absent. The pathologic hallmarks of the disease include pallor of the corticospinal tract due to loss of motor neurons, presence of ubiquitin-positive inclusions within surviving motor neurons, and deposition of pathologic aggregates. The etiology of amyotrophic lateral sclerosis is likely to be multifactorial, involving both genetic and environmental factors. The disease is inherited in 5-10% of the cases.

DrugBank Results

Name	State	Route	Country
Riluzole	solid	Oral	Canada,EU,US
Description			
A glutamate antagonist (receptors, glutamate) used as an anticonvulsant (anticonvulsants) and to prolong the survival of patients with amyotrophic lateral sclerosis. Riluzole is marketed as Rilutek by Sanofi.			
Indication			
For the treatment of amyotrophic lateral sclerosis (ALS, Lou Gehrig's Disease)			

Name	State	Route	Country
Phenylbutyric acid	solid	Oral	Canada,EU,US
Description			
Phenylbutyric acid is a fatty acid and a derivative of [butyric acid] naturally produced by colonic bacteria fermentation. It demonstrates a number of cellular and biological effects, such as relieving inflammation and acting as a chemical chaperone.[A249035] It is used to treat genetic metabolic syndromes, neuropathies, and urea cycle disorders.[L386,L42105]			
Indication			
Phenylbutyric acid is used for the treatment of various conditions, including urea cycle disorders, neonatal-onset deficiency, late-onset deficiency disease in patients with a history of hyperammonemic encephalopathy. Phenylbutyric acid must be combined with dietary protein restriction and, in some cases, essential amino acid supplementation.[L386] Phenylbutyric acid, as sodium phenylbutyrate, is used in combination with [tauroursodeoxycholic acid] to treat amyotrophic lateral sclerosis (ALS) in adults.[L42105,L43473]			

P.PORTO

ESCOLA SUPERIOR DE SAÚDE

M

MESTRADO

BIOESTATÍSTICA E BIOINFORMÁTICA APLICADAS À SAÚDE