TECNOLOGIA BARREIRO

ESCOLA SUPERIOR POLITÉCNICO SETÚBAL

EcoTracker

Licenciatura em Bioinformática

Laboratório de Bioinformática

Docente: Francisco Martins

Data: 27/01/2025

Joana Caetano 202200646

Lucas Terlica 202200239

Índice

Índice de Figuras

1. Introdução

A perda de biodiversidade é um dos desafios ambientais mais urgentes da atualidade, com impactos profundos nos ecossistemas e na sustentabilidade do planeta. O crescimento exponencial de espécies ameaçadas é impulsionado por fatores antropogénicos, como a desflorestação, urbanização e poluição, além das alterações climáticas, que alteram drasticamente os habitats naturais e as interações ecológicas (Butchart et al., 2010). A extinção de espécies compromete serviços ecossistémicos essenciais, como a polinização, o controlo biológico de pragas e a regulação do ciclo da água, afetando diretamente a vida humana (Cardinale et al., 2012).

A monitorização da biodiversidade é fundamental para compreender a distribuição das espécies, avaliar o seu estado de conservação e apoiar a implementação de estratégias eficazes de proteção ambiental. No entanto, a recolha e análise de dados sobre a biodiversidade em escala global representam um desafio significativo devido à complexidade dos ecossistemas e à dispersão das informações disponíveis. A falta de ferramentas acessíveis e integradas para a exploração desses dados dificulta a sua utilização por investigadores, conservacionistas e decisores políticos, comprometendo os esforços de conservação e mitigação do declínio das populações de espécies (Pimm et al., 2014).

O ***EcoTracker*** surge como uma solução para este problema, oferecendo uma aplicação web intuitiva e acessível para facilitar a visualização e análise de dados sobre biodiversidade. Utilizando a API do GBIF (Global Biodiversity Information Facility), uma das maiores bases de dados sobre biodiversidade global (GBIF, 2024), a aplicação permite aos utilizadores pesquisar espécies, visualizar as suas ocorrências geográficas e analisar tendências ao longo do tempo. A disponibilização destes dados de forma interativa tem o potencial de se tornar uma ferramenta essencial na monitorização da biodiversidade, apoiando a tomada de decisões baseadas em dados científicos sólidos.

A aplicação tira partido de tecnologias como *Flask*, para o backend (Grinberg et al., 2018), *Folium*, para a visualização geográfica interativa Jeng et al., 2021), e *Docker*, para garantir a portabilidade e escalabilidade do sistema (Merkel et al., 2014). A disponibilização destes dados de forma interativa pode tornar-se uma ferramenta essencial para apoiar a investigação científica e impulsionar ações de conservação eficazes.

2. Objetivos

O principal objetivo do ***EcoTracker*** é facilitar a monitorização e análise da biodiversidade através da disponibilização de uma ferramenta acessível e interativa que permite visualizar a distribuição geográfica de espécies e identificar padrões de ocorrência ao longo do tempo. A plataforma visa apoiar investigadores, conservacionistas e decisores políticos na compreensão da dinâmica populacional das espécies, contribuindo para a identificação de áreas de risco e para a definição de estratégias de conservação baseadas em dados científicos.

Ao fornecer uma interface intuitiva e baseada em dados de fontes globais confiáveis, como a Global Biodiversity Information Facility (GBIF), o ***EcoTracker*** pretende facilitar o acesso à informação sobre biodiversidade, permitindo uma tomada de decisão mais informada e a implementação de políticas eficazes para a preservação da biodiversidade.

Além disso, a aplicação tem como propósito fomentar o envolvimento da comunidade científica e do público em geral, incentivando uma maior consciência ambiental.

3. Métodos

A implementação do ***EcoTracker*** envolveu uma abordagem estruturada com recurso a diversas ferramentas tecnológicas para garantir uma solução eficiente e escalável. O projeto foi desenvolvido utilizando uma stack tecnologicamente robusta, composta por *Flask*, *Folium*, *Docker*, e integração com a API do GBIF para recolha de dados de biodiversidade.

3.1 Desenvolvimento do Backend com Flask

O backend do ***EcoTracker*** foi desenvolvido com ***Flask***, uma framework minimalista e extensível em Python, amplamente utilizada para a construção de aplicações web devido à sua simplicidade e flexibilidade (Grinberg et al., 2018). Esta framework permite criar rotas para gerir diferentes tipos de pedidos HTTP, facilitando a comunicação entre o utilizador e os serviços da aplicação.

Principais Funcionalidades do Backend

O backend desempenha um papel central no funcionamento do ***EcoTracker***, sendo responsável pela:

- **Gestão de pedidos HTTP:**
  - Processar dados enviados pelos utilizadores através de parâmetros na query string.
  - Responder a esses pedidos devolvendo dados processados, como ficheiros *JSON*, mapas interativos ou páginas *HTML*.
- **Interação com a API GBIF:**
  - O backend conecta-se à API GBIF (Global Biodiversity Information Facility) para recolher dados de biodiversidade (GBIF, 2024). Estes dados incluem informações como a localização geográfica (latitude e longitude), a frequência de avistamentos e a distribuição temporal de espécies.
  - O backend implementa mecanismos para filtrar e organizar os dados de forma eficiente antes de os enviar para o utilizador ou utilizar para visualizações.
- **Processamento e transformação de dados:**
  - Utiliza bibliotecas como **Pandas** para manipulação e análise de dados recolhidos (McKinney, 2011).
  - Converte os dados brutos da API em formatos estruturados que podem ser utilizados para gerar mapas e gráficos.
- **Integração com o frontend:**
  - Fornece os resultados processados (ex.: mapas, gráficos, etc.) para o frontend, garantindo que os utilizadores têm acesso a representações visuais e informativas dos dados de biodiversidade.

3.2 Estrutura do Código

O backend está organizado de forma modular, o que garante clareza, manutenção simplificada e flexibilidade para futuras expansões. A organização principal é a seguinte:

- Rotas: Definidas no ficheiro app.py, gerem pedidos HTTP para funcionalidades como pesquisa de espécies, geração de mapas e visualização de gráficos.
- Funções auxiliares: Incluem chamadas à API GBIF e manipulação de dados, além de integração com bibliotecas como Pandas e Matplotlib (Hunter, 2007).
- Templates: Os ficheiros *HTML* que definem a interface do utilizador estão localizados na pasta templates/.

3.3 Funcionalidades Implementadas

Pesquisa de espécies

A funcionalidade de pesquisa de espécies permite processar o nome científico de uma espécie enviado pelo utilizador, recolher os dados geográficos associados e preparar os resultados para outras partes da aplicação. O backend utiliza uma função auxiliar para comunicar com a API GBIF e obter informações relevantes.

A função que consulta a API processa a resposta recebida, extraindo dados como:

- Coordenadas geográficas (latitude e longitude) das ocorrências.
- Datas de avistamento.
- Informações complementares acerca da espécie.

Detalhes técnicos:

- A função de consulta à API utiliza a biblioteca requests para enviar pedidos HTTP e processar as respostas.
- Os dados recebidos são filtrados para garantir que apenas ocorrências com coordenadas válidas e campos relevantes são mantidas.
- O formato final dos dados é estruturado de forma a facilitar a sua integração com funcionalidades como visualizações gráficas ou mapas interativos.

Geração de mapas interativos

A rota /map permite criar mapas interativos com base nos dados recolhidos da API GBIF. O utilizador fornece o nome da espécie como parâmetro, e o backend gera um mapa em que cada ocorrência é representada por um marcador georreferenciado.

- O mapa é construído dinamicamente utilizando a biblioteca *Folium* (Jeng et al., 2021), sendo centralizado automaticamente com base nas coordenadas fornecidas.
- O ficheiro do mapa gerado é armazenado na pasta templates/ como um ficheiro *HTML*, permitindo que seja facilmente renderizado pelo frontend.

Detalhes técnicos:

- O backend adiciona marcadores no mapa para cada ocorrência, personalizando detalhes como mensagens de pop-up com informações adicionais.
- O mapa é configurado para uma vista inicial genérica, que é ajustada automaticamente com base nos dados geográficos da espécie.

Visualização de gráficos

A rota /visualize é utilizada para criar gráficos baseados nos dados de ocorrência de uma espécie. Estes gráficos podem incluir informações como:

- A distribuição temporal dos avistamentos (ex.: frequência ao longo dos anos).
- A densidade geográfica de ocorrências.

O backend utiliza Matplotlib para gerar gráficos de forma dinâmica e exportá-los como ficheiros de imagem. Estes ficheiros são armazenados na pasta static/ e integrados no frontend como elementos visuais.

Detalhes técnicos:

- Os dados utilizados para os gráficos são filtrados e organizados com Pandas, garantindo que estão no formato necessário para visualização.
- A geração do gráfico inclui personalizações como títulos, eixos rotulados e grelhas, para maior clareza e interpretação.

Comparação de espécies

A rota /compare permite comparar a distribuição de várias espécies num único mapa interativo. O utilizador fornece uma lista de espécies como parâmetro na query string, e o backend processa cada uma separadamente antes de gerar um mapa combinado.

- Cada espécie é representada com marcadores de uma cor distinta, permitindo diferenciar facilmente as distribuições.
- O mapa gerado inclui uma legenda para identificar as cores atribuídas a cada espécie.

Detalhes técnicos:

- O backend gere múltiplos pedidos à API GBIF, um para cada espécie fornecida, e combina os dados recolhidos num único conjunto.
- A biblioteca *Folium* é utilizada para sobrepor os dados de todas as espécies no mesmo mapa, ajustando automaticamente a visualização para incluir todos os pontos de ocorrência.

3.4 Bibliotecas e Ferramentas Utilizadas

O projeto ***EcoTracker*** utiliza diversas bibliotecas e ferramentas para implementar as suas funcionalidades principais de forma eficiente e modular. Entre as principais estão:

- **Flask:** Para gerir as rotas HTTP e estruturar o backend da aplicação (Grinberg, 2018).
- **Pandas:** Para manipulação e organização de dados geográficos e temporais (McKinney, 2011).
- **Matplotlib:** Para a criação de gráficos que representam tendências nos dados (Hunter, 2007).
- **Folium:** Para gerar mapas interativos com base nos dados georreferenciados (Jeng et al., 2021).
- **Requests:** Para comunicação com a API GBIF e obtenção de dados externos (Reitz, 2023).

```
1  import requests
2  import folium
3  from folium.plugins import MarkerCluster
4  from flask import Flask, render_template, request
5  import matplotlib.pyplot as plt
6  import io
7  import base64
8  from datetime import datetime
9  import csv
10 from flask import send_file
11 import os
12 from branca.element import Template, MacroElement
```

Figura 1 - Bibliotecas utilizadas no EcoTracker

```
1  Flask == 3.0.3
2  requests == 2.32.3
3  folium == 0.18.0
4  matplotlib == 3.6.2
5  pandas == 1.5.3
6  gunicorn == 20.1.0
7  pytest == 8.3.4
```

Figura 2 - Lista de dependências do EcoTracker especificadas no ficheiro requirements.txt

3.5 Interface com CSS

O ***EcoTracker*** utiliza ***CSS*** para garantir uma interface intuitiva e responsiva. O ficheiro styles.css, localizado em hosting/static/, contém as regras de estilização que definem a aparência da aplicação, separando a lógica do backend da camada visual.

A implementação foca-se na usabilidade e acessibilidade, assegurando que a aplicação seja compatível com diferentes dispositivos e resoluções. Foram aplicadas técnicas de **responsividade, animações suaves e organização estrutural** para melhorar a experiência do utilizador.

3.6 Infraestrutura e Conteinerização com Docker

A aplicação ***EcoTracker*** foi containerizada utilizando dois ambientes *Docker* distintos, cada um com um propósito específico. Esta abordagem permite uma configuração modular, garantindo ambientes consistentes e fáceis de reproduzir, ao mesmo tempo que assegura a separação entre o backend e o servidor de hospedagem.

Estrutura de Conteinerização

O projeto está organizado em dois diretórios principais, cada um contendo um *Dockerfile* específico para configurar os respetivos serviços:

1. **Diretório WebApp:**
    - Contém o *Dockerfile* que configura o ambiente para a aplicação *Flask*.
    - Inclui a instalação das dependências da aplicação, especificadas no ficheiro requirements.txt.
    - Executa o servidor **Gunicorn**, que serve a aplicação *Flask*, na porta 5000.
2. **Diretório hosting:**
    - Contém o *Dockerfile* utilizado para configurar o servidor **Nginx**.
    - Utiliza um ficheiro de configuração personalizado (nginx.conf), que define o comportamento do proxy reverso, incluindo o redirecionamento de pedidos para o serviço *Flask* e a gestão de ficheiros estáticos.

Descrição dos Serviços no docker-compose.yml

O ficheiro docker-compose.yml orquestra os dois serviços principais:

- **web:**
    - Constrói o container *Flask* a partir do diretório WebApp.
    - Mapeia a porta interna 5000 para a mesma porta na máquina anfitriã.
    - Executa o servidor Gunicorn com 4 workers, garantindo o processamento eficiente dos pedidos.
- **nginx:**
    - Constrói o container Nginx a partir do diretório hosting.
    - Mapeia a porta interna 80 para a porta 80 da máquina anfitriã.

- Carrega a configuração personalizada do ficheiro nginx.conf, que:
  - Redireciona pedidos na rota / para o serviço web.
  - Serve ficheiros estáticos diretamente, utilizando o alias configurado.
- Define uma dependência explícita do serviço web para garantir que a aplicação *Flask* está disponível antes de iniciar o Nginx.

3.7 Instância e Domínio

O ***EcoTracker*** está hospedado numa instância **AWS EC2 t3.micro**, uma solução em cloud que proporciona recursos adequados para suportar a execução contínua e eficiente da aplicação. A escolha desta instância deve-se ao seu equilíbrio entre custo e performance, sendo ideal para cargas de trabalho de baixa a moderada intensidade (Amazon Web Services, 2023). Esta configuração permite ainda a escalabilidade horizontal, facilitando o ajuste da capacidade conforme a necessidade, através da adição de instâncias adicionais.

Configuração da Instância:

- **Tipo:** t3.micro
- **vCPUs:** 2
- **Memória:** 1 GB
- **Armazenamento:** Elastic Block Store (EBS) configurável, garantindo persistência dos dados mesmo após reinicializações.

Domínio e Direcionamento de Tráfego:

- O domínio **ecotracker.me** foi registado através do serviço **dominios.pt**, sendo configurado para apontar diretamente para o IP público da instância EC2.
- Utilizando o serviço **AWS Route 53** para a gestão de DNS, foi configurado um registo A, garantindo a resolução rápida e fiável do domínio, direcionando os utilizadores à aplicação.

3.8 Testes Automatizados

Os testes automatizados foram desenvolvidos utilizando **pytest**, garantindo a qualidade e estabilidade das funcionalidades implementadas (Krekel et al., 2015). Estes testes cobrem a validação da interação com a API GBIF, a correta geração de mapas interativos, a verificação de detalhes de espécies e a criação de ficheiros *HTML*.

Cobertura dos Testes

Os testes implementados verificam aspetos essenciais do funcionamento da aplicação, incluindo:

- **Validação da consulta à API GBIF:**
    - Testes garantem que a API retorna os dados corretos para espécies válidas e responde apropriadamente para nomes inválidos ou inexistentes.
    - São verificadas respostas esperadas para pesquisas feitas por espécie e por país.
- **Geração e verificação de ficheiros HTML para mapas interativos:**
    - Testes asseguram que os mapas gerados são corretamente criados e armazenados na pasta templates/.
    - O conteúdo dos ficheiros *HTML* é verificado para garantir que não estão vazios ou inválidos.
- **Obtenção de detalhes de espécies:**
    - Testes confirmam que a função responsável por buscar informações detalhadas de espécies retorna os campos esperados.
    - Há testes que simulam o comportamento da API GBIF usando unittest.mock.patch para verificar se a aplicação trata corretamente respostas da API (van Rossum & Warsaw, 2001).
- **Criação de mapas comparativos entre espécies:**
    - Testes verificam que a funcionalidade que gera mapas de comparação entre espécies diferentes funciona corretamente.
    - O ficheiro species_comparison_map.html é criado e validado como parte dos testes.
- **Validação da obtenção das espécies mais comuns por país:**
    - Testa-se se a função retorna corretamente uma lista com as espécies mais avistadas num determinado país.
    - Cada item da lista deve ser um **tuplo** contendo o nome da espécie e o número de ocorrências.

3.9 Integração Contínua e Deployment com GitLab CI/CD

Para garantir um fluxo de desenvolvimento eficiente e automatizado, foi implementada uma pipeline de CI/CD utilizando **GitLab CI/CD**. Esta pipeline automatiza os processos de teste, build e deployment, permitindo que cada alteração no repositório seja validada e implantada de forma rápida e confiável (GitLab Inc., 2023).

Estrutura da Pipeline

A pipeline é definida no ficheiro .gitlab-ci.yml e está organizada em três etapas principais:

1. **Testes (Stage: test):**
    - Executa **pytest** para validar o código e garantir que todas as funcionalidades estão a operar conforme o esperado.
    - Instala as dependências do projeto a partir do ficheiro requirements.txt antes de executar os testes.
    - Define a variável de ambiente PYTHONPATH para incluir o diretório WebApp, assegurando que as importações funcionam corretamente durante os testes.
2. **Build (Stage: build):**
    - Cria as imagens *Docker* necessárias para a aplicação.
    - Utiliza o serviço **docker:dind** (*Docker*-in-*Docker*) para garantir que os containers podem ser construídos no ambiente de integração contínua.
    - O comando docker-compose build é utilizado para gerar as imagens definidas no ficheiro docker-compose.yml.
3. **Deployment (Stage: deploy):**
    - Conecta-se à instância AWS via **SSH** utilizando uma chave privada armazenada de forma segura como variável de ambiente no GitLab CI.
    - Atualiza as imagens *Docker* na instância e reinicia os serviços utilizando os comandos: docker-compose pull && docker-compose up -d
    - A etapa de deployment está configurada para ser executada apenas nas branches main e build-hotfix, garantindo que apenas alterações aprovadas sejam implementadas no ambiente de produção.

4. Resultados

4.1 Pesquisa de Espécies

A aplicação permite a busca de espécies através do seu nome científico, retornando informações detalhadas e a localização das ocorrências num mapa interativo. A Figura 3 apresenta a interface principal da aplicação, onde o utilizador pode inserir o nome da espécie para realizar a pesquisa.

A Figura 4 mostra os filtros disponíveis, que possibilitam refinar a busca por datas e país (neste caso não foram utilizados). Após a submissão da pesquisa, os resultados são apresentados num mapa (Figura 5), onde as ocorrências são agrupadas em *clusters* para melhor visualização.

A Figura 7 ilustra um exemplo de ocorrência individual de *Oryctolagus cuniculus* no mapa interativo. Ao clicar num marcador, o utilizador tem acesso a um *popup* com detalhes sobre a espécie, incluindo a sua classificação taxonómica, nome comum, data de registo da observação e uma imagem representativa.

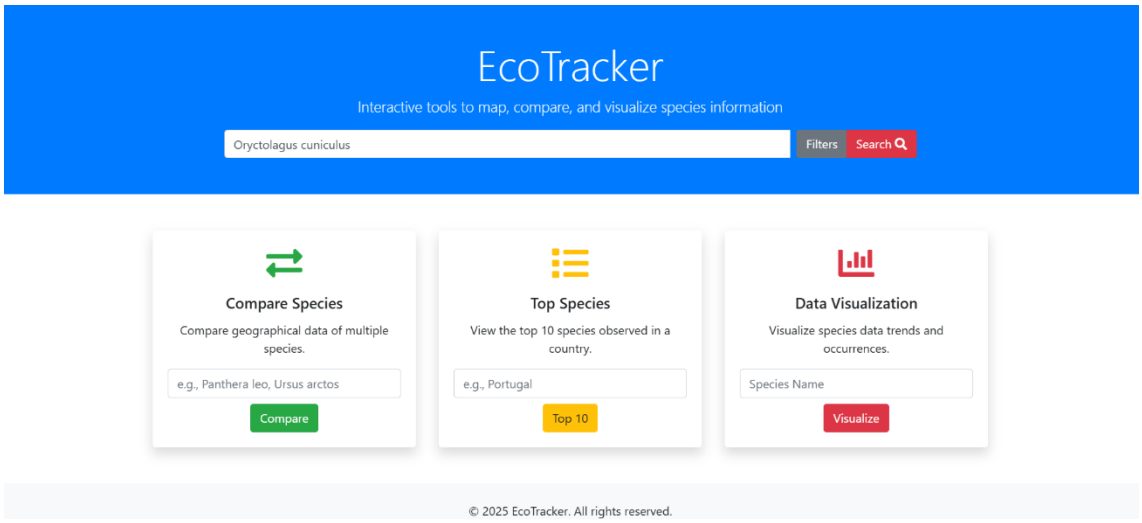

Figura 3 - Interface principal do EcoTracker

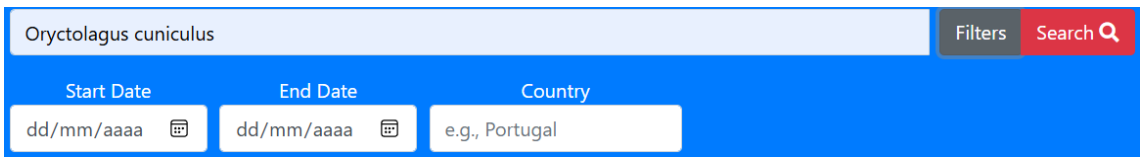

Figura 4 - Filtros de pesquisa avançada

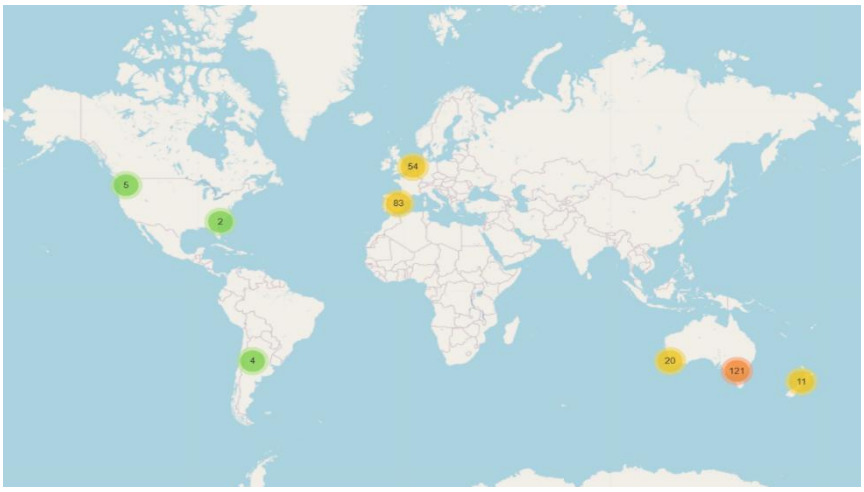

Figura 5 - Mapa de avistamentos de Oryctolagus cuniculus

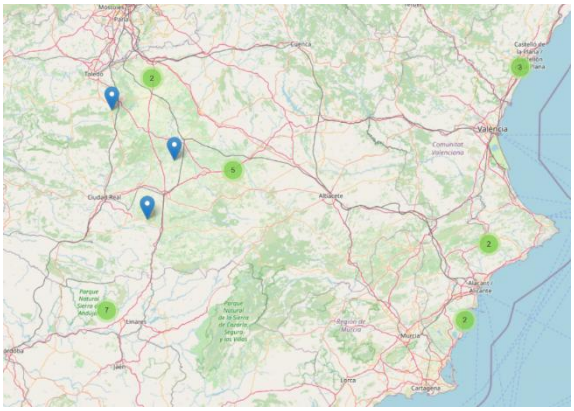

Figura 6 - Mapa de avistamentos de Oryctolagus cuniculus com zoom

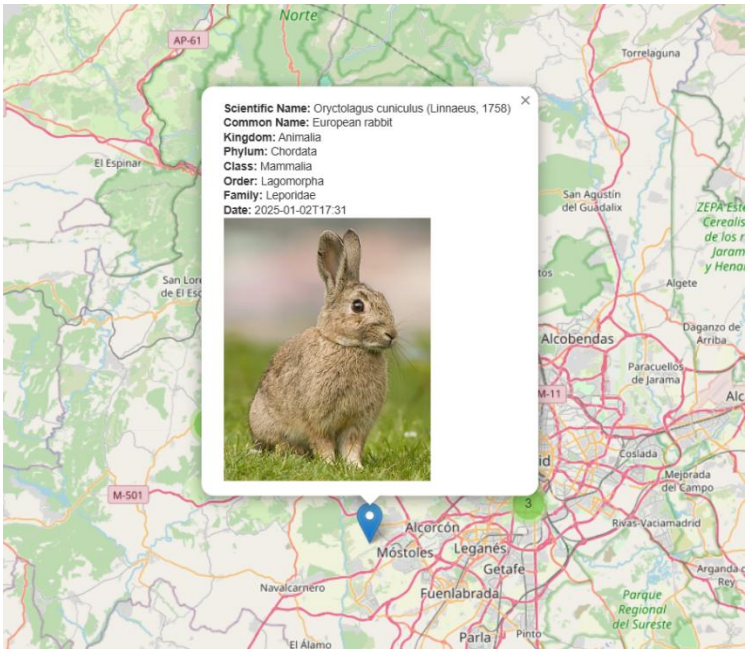

Figura 7 - Janela de informações de Oryctolagus cuniculus

4.2 Comparação de Espécies

A funcionalidade de comparação permite a visualização simultânea da distribuição geográfica de múltiplas espécies. A Figura 8 ilustra a interface de pesquisa para comparação, onde foram selecionadas as espécies *Oryctolagus cuniculus* e *Ursus arctos*.

O resultado da comparação é apresentado na Figura 9, onde as duas espécies são representadas em diferentes cores, evidenciando as regiões onde foram registadas ocorrências. As Figuras 6 e 7 mostram detalhes individuais das espécies, incluindo nome científico, classificação taxonómica, e imagens associadas às observações.

As Figuras 10 e 11 ilustram a comparação entre *Oryctolagus cuniculus* (coelho-bravo) e *Ursus arctos* (urso-pardo). Cada ocorrência individual é representada por um marcador, e ao clicar num deles, um *popup* exibe informações detalhadas sobre a espécie, incluindo a sua classificação taxonómica, data do avistamento e uma imagem representativa.

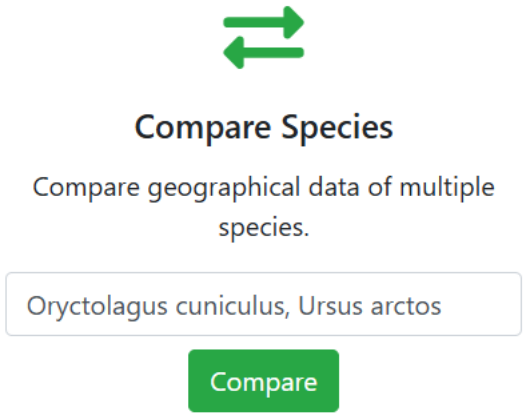

Figura 8 - Funcionalidade de Comparação de Espécies

Figura 9 - Mapa da comparação das espécies Oryctolagus cuniculus e Ursus arctos

Figura 10 - Janela de informações de Oryctolagus cuniculus

Figura 11 - Janela de informações de Ursus arctos

4.3 Top 10 Espécies

A funcionalidade *Top Species* permite listar as 10 espécies mais observadas num determinado país. A Figura 12 apresenta a interface para pesquisa, enquanto a Figura 13 exibe os resultados, mostrando a lista de espécies mais registadas, acompanhadas do número de ocorrências e de imagens representativas.

Figura 12 - Funcionalidade de Top10 Espécies

Top 10 Species in Germany

Explore the most frequent species in this region

Results

Rank	Scientific Name	Occurrences	Image
1	Passer domesticus	12	
2	Parus major	11	
3	Ardea cinerea	9	
4	Pica pica	7	
5	Anas platyrhynchos	7	
6	Cyanistes caeruleus	6	
7	Fringilla coelebs	5	
8	Columba livia	5	
9	Talpa europaea	5	
10	Turdus merula	4	

Figura 13- Listagem das 10 espécies mais observadas na Alemanha

4.4 Visualização de Dados

A aplicação também permite visualizar a distribuição temporal de observações de uma espécie específica. A Figura 14 mostra a interface de pesquisa para esta funcionalidade. A Figura 15 apresenta um gráfico de barras ilustrando a frequência de avistamentos da espécie *Oryctolagus cuniculus* ao longo do tempo.

Os dados utilizados para a construção do gráfico são apresentados na Figura 16, que contém uma tabela com a contagem diária de ocorrências registadas.

Estes resultados demonstram a eficácia do ***EcoTracker*** na organização e apresentação de dados de biodiversidade, proporcionando uma interface interativa para a exploração de informações científicas.

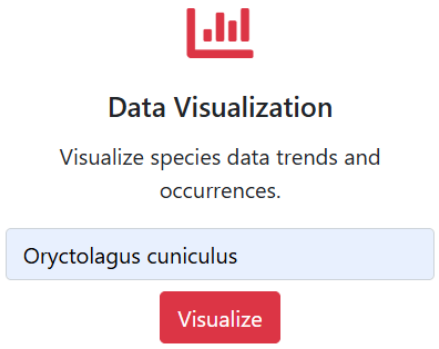

Figura 14 - Funcionalidade de Visualização de Dados

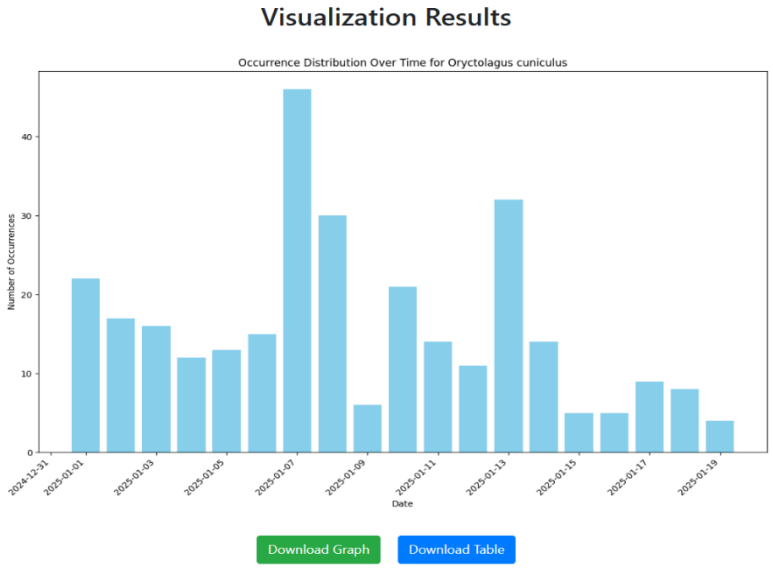

Figura 15 - Gráfico de distribuição de ocorrências ao longo do tempo para Oryctolagus cuniculus

Date	Count
2025-01-01	22
2025-01-02	17
2025-01-03	16
2025-01-04	12
2025-01-05	13
2025-01-06	15
2025-01-07	46
2025-01-08	30
2025-01-09	6
2025-01-10	21
2025-01-11	14
2025-01-12	11
2025-01-13	32
2025-01-14	14
2025-01-15	5
2025-01-16	5
2025-01-17	9
2025-01-18	8
2025-01-19	4

Figura 16 - Tabela de contagem de ocorrências por data para Oryctolagus cuniculus

5. Discussão

Os resultados obtidos através do ***EcoTracker*** demonstram a eficácia da plataforma na recolha e visualização de dados sobre biodiversidade, permitindo uma análise interativa da distribuição geográfica e temporal das espécies. A integração com a API do **Global Biodiversity Information Facility (GBIF)** proporcionou acesso a um vasto conjunto de dados científicos, permitindo aos utilizadores explorar padrões de ocorrência de espécies de forma acessível e detalhada.

Pesquisa de Espécies e Visualização de Dados

Os resultados da funcionalidade de pesquisa de espécies evidenciaram a capacidade da aplicação de recuperar e exibir informações detalhadas sobre a distribuição de espécies individuais. As figuras apresentadas mostraram a distribuição geográfica de **Oryctolagus cuniculus**, com ocorrências registadas em várias regiões, destacando-se a Europa e a Austrália. Os utilizadores puderam visualizar metadados detalhados, incluindo a classificação taxonómica, data de registo da ocorrência e imagens representativas. Esta abordagem interativa reforça a importância de ferramentas de visualização na análise de biodiversidade, alinhando-se com estudos prévios que destacam a necessidade de plataformas acessíveis para integrar e analisar grandes volumes de dados ecológicos (Pimm et al., 2014; GBIF, 2024).

A funcionalidade de visualização temporal permitiu analisar a frequência de avistamentos ao longo do tempo, conforme ilustrado pelo gráfico gerado para **Oryctolagus cuniculus**. O aumento do número de registos em determinados períodos sugere padrões sazonais que podem estar associados a fatores ambientais e comportamentais da espécie, informação relevante para estudos de ecologia populacional (Butchart et al., 2010).

Comparação de Espécies

A funcionalidade de comparação demonstrou ser uma ferramenta útil para a análise conjunta da distribuição de múltiplas espécies. Os resultados mostraram as ocorrências de **Oryctolagus cuniculus** e **Ursus arctos**, permitindo identificar áreas de sobreposição e exclusividade geográfica. A presença do urso-pardo em regiões montanhosas e do coelho-bravo em zonas de menor altitude confirma padrões ecológicos previamente descritos na literatura, evidenciando a utilidade da plataforma na análise de distribuição de habitats (Cardinale et al., 2012).

Top 10 Espécies

A funcionalidade de **Top Species** revelou as espécies mais frequentemente registadas em determinadas regiões, com destaque para **Passer domesticus**, **Parus major** e **Ardea cinerea** na Alemanha. Estes dados corroboram estudos sobre a distribuição de aves em ambientes urbanos e rurais, onde espécies generalistas tendem a apresentar uma maior taxa de ocorrência devido à sua adaptação a diferentes habitats (Jenkins et al., 2020).

Validação e Limitações

Os testes automatizados desenvolvidos com **pytest** confirmaram a funcionalidade das principais operações, incluindo a consulta à API GBIF e a geração de mapas e gráficos. No entanto, a qualidade dos dados recuperados depende da base de dados original do GBIF, o que pode resultar em variações na precisão e cobertura espacial dos registos (GBIF, 2024).

6. Conclusão

Os resultados obtidos demonstram que o ***EcoTracker*** é uma ferramenta eficaz para a visualização e análise de biodiversidade, permitindo aos utilizadores explorar a distribuição de espécies de forma acessível e interativa. A sua integração com a API GBIF e o uso de tecnologias como *Flask, Folium* e *Docker* garantem escalabilidade e flexibilidade na implementação da aplicação (Grinberg et al., 2018; Merkel et al., 2014). Futuras melhorias a configuração de **HTTPS** utilizando **Let's Encrypt** para maior segurança, a melhoria da interface para otimizar a experiência do utilizador, e o aumento da capacidade de processamento de avistamentos sem comprometer a velocidade da aplicação.

7. Referências

Amazon Web Services. (2023). *Amazon EC2 Instance Types*. Retrieved from https://aws.amazon.com/ec2/instance-types/

Amazon Web Services. (2023). *AWS Auto Scaling – Scalable Cloud Resources*. Retrieved from https://aws.amazon.com/autoscaling/

Amazon Web Services. (2023). *Amazon Elastic Block Store (EBS)*. Retrieved from https://aws.amazon.com/ebs/

Butchart, S. H. M., Walpole, M., Collen, B., Van Strien, A., Scharlemann, J. P., Almond, R. E. A., ... & Watson, R. (2010). Global biodiversity: Indicators of recent declines. *Science, 328*(5982), 1164-1168. https://doi.org/10.1126/science.1187512

Canonical. (2022). *Ubuntu Server 22.04 LTS Documentation*. Retrieved from https://ubuntu.com/server/docs

GBIF. (2024). *Global Biodiversity Information Facility*. Retrieved from https://www.gbif.org

GitLab Inc. (2023). *Continuous Integration & Continuous Deployment with GitLab CI/CD*. Retrieved from https://docs.gitlab.com/ee/ci/

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90-95. https://doi.org/10.1109/MCSE.2007.55

Jenkins, R. K., Keane, A., Rakotomboavonjy, V., Rakotondravony, D., Randrianantoandro, J. C., & Raxworthy, C. J. (2020). Distribution and conservation status of the passerine birds of Madagascar. *Biodiversity and Conservation, 29*(4), 997-1015. https://doi.org/10.1007/s10531-020-01936-1

Jeng, Y., Liu, Y., & Lin, C. (2021). Interactive Geographic Visualization with Folium and OpenStreetMap. *International Journal of Geospatial Research, 8*(2), 45-60. https://doi.org/10.4018/IJGR.20210401.oa01

Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, M., & Gal, A. (2015). *pytest: Simple powerful testing with Python*. Retrieved from https://pytest.org

McKinney, W. (2011). Pandas: A foundational Python library for data analysis and statistics. *Python for High-Performance and Scientific Computing, 14*(9), 1-9.

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal, 2014*(239), 2.

Pimm, S. L., Jenkins, C. N., Abell, R., Brooks, T. M., Gittleman, J. L., Joppa, L. N., ... & Sexton, J. O. (2014). The biodiversity of species and their rates of extinction, distribution, and protection. *Science, 344*(6187), 1246752. https://doi.org/10.1126/science.1246752

Reitz, K. (2023). Requests: HTTP for Humans. Retrieved from https://docs.python-requests.org

van Rossum, G., & Warsaw, B. (2001). *PEP 8 – Style Guide for Python Code*. Python Enhancement Proposals.

dominios.pt. (2024). *Serviço de Registo de Domínios em Portugal*. Retrieved from https://www.dominios.pt