

Projeto DESOFS

Secure Software Development Life Cycle

Paulo Manuel Baltarejo De Sousa

Cláudio Roberto Ribeiro Maia

Autores

Bernardo Lima – 1221977

Diogo Vieira - 1240448

João Monteiro – 1240469

João Taveira - 1220258

Índice

Abstract.....	1
Introdução.....	2
Descrição do Projeto.....	3
Processo SSDLC Adotado.....	4
Requisitos do Sistema.....	4
Modelo de Domínio.....	4
Requisitos Funcionais.....	6
Requisitos Não Funcionais.....	6
Requisitos de Segurança.....	6
Casos de Uso.....	7
Threat Modeling.....	8
Modelação aplicada ao sistema.....	10
Diagrama de Fluxo de Dados (DFD).....	10
DFD nível 0.....	10
DFD nível 1.....	11
DFD nível 2.....	12
Arquitetura do Sistema.....	13
Ambientes de Execução e Deploy.....	14
OWASP ASVS Nível 2.....	14
Metodologia adotada.....	15
Considerações finais.....	16
Ameaças.....	16
Categorização.....	17
Árvores de ameaças.....	17
Atacante tenta-se passar por funcionário legítimo.....	18
Atacante tenta alterar os dados de registo de atividades.....	18
Um funcionário nega ter registado determinada atividade.....	18
Dados sensíveis dos utilizadores expostos a utilizadores não autorizados.....	19
Aplicação fica indisponível devido a um ataque.....	19
Funcionário tenta aceder a funcionalidades administrativas sem permissão.....	19
Categorização das ameaças identificadas.....	20
Atacante tenta-se passar por funcionário legítimo.....	20
Atacante tenta alterar os dados de registo de atividades.....	20
Um funcionário nega ter registado determinada atividade.....	21
Dados sensíveis dos utilizadores expostos a utilizadores não autorizados.....	21
Aplicação fica indisponível devido a um ataque.....	22
Funcionário tenta aceder a funcionalidades administrativas sem permissão.....	22
Priorização das Ameaças.....	22
Misuse Cases.....	23
Estrutura de Código, Testes e Integração Contínua.....	24
Organização da Solução.....	24
Testes Automatizados.....	24
Integração Contínua com GitHub Actions.....	25
Alinhamento com SSDLC.....	27
Testes e Validação.....	27
Resultados Obtidos.....	27
Validação Manual.....	28
Considerações Finais.....	28
Conclusões.....	28
Trabalho Futuro.....	28
Bibliografia e referências.....	29
Anexos.....	30
A – Estrutura de Código da Solução.....	30
B – Evidências de Testes.....	30
C – Artefactos de Segurança.....	31

Índice de imagens

Figura 1 - https://codesigningstore.com/wp-content/uploads/2023/02/secure-software-development-life-cycle-process.png	2
Figura 2 - fases do projeto	3
Figura 3 - diagrama metodologia SSDLC	4
Figura 4 - Modelo de Domínio	5
Figura 5 - modelo UML de Domínio	6
Figura 6 - diagrama de casos de uso	7
Figura 7 - diagrama de casos de uso	8
Figura 8 - Threat Modeling Process	9
Figura 9 - DFD nível 0	11
Figura 11 - DFD nível 1	12
Figura 13 - DFD nível 2	13
Figura 15 - Arquitetura Lógica da Aplicação	14
Figura 16 - Misuse case 1	23
Figura 17 - Misuse case 2	24
Figura 18 - Misuse case 3	24
Figura 19 - Pipeline CI/CD	26

Abstract

Este relatório descreve o desenvolvimento, implementação e validação de uma aplicação web modular denominada *EcoImpact*, concebida no âmbito da unidade curricular de Desenvolvimento de Software Seguro (DESOFS). A solução tem como objetivo promover a consciência ecológica dos utilizadores, permitindo calcular a pegada ambiental com base em hábitos registados e oferecendo sugestões de melhoria sustentáveis.

A aplicação foi desenvolvida com base numa arquitetura distribuída, composta por três componentes principais: *frontend* em Blazor WebAssembly, *backend* em ASP.NET Core e base de dados PostgreSQL, suportados por ambientes de desenvolvimento e produção geridos através de Docker e Render.com. A segurança da aplicação foi uma prioridade desde o início, com a adoção de boas práticas como autenticação via JWT, proteção de variáveis sensíveis através de ficheiros .env, e integração de workflows CI/CD seguros com GitHub Actions.

O relatório documenta ainda a conformidade com o OWASP ASVS Nível 2, a implementação de testes automatizados (unitários e DAST), bem como a análise dos resultados obtidos durante o processo de integração contínua e deploy controlado. A componente prática é complementada com evidências técnicas que validam a segurança e eficácia da solução proposta, demonstrando a sua maturidade em contextos de produção reais.

Introdução

O presente documento foi desenvolvido no âmbito do projeto da unidade curricular de **Desenvolvimento de Software Seguro (DESOFS)**, cujo objetivo é desenvolver uma aplicação seguindo o processo de **Secure Software Development Life Cycle (SSDLC)**.

A implementação de metodologias de desenvolvimento de software seguro, é uma prática fundamental para o desenvolvimento de aplicações robustas e confiáveis. A adopção da Framework 'Secure Software Development Life Cycle' (SSDLC) integra requisitos de Segurança em cada fase do processo de desenvolvimento de software.

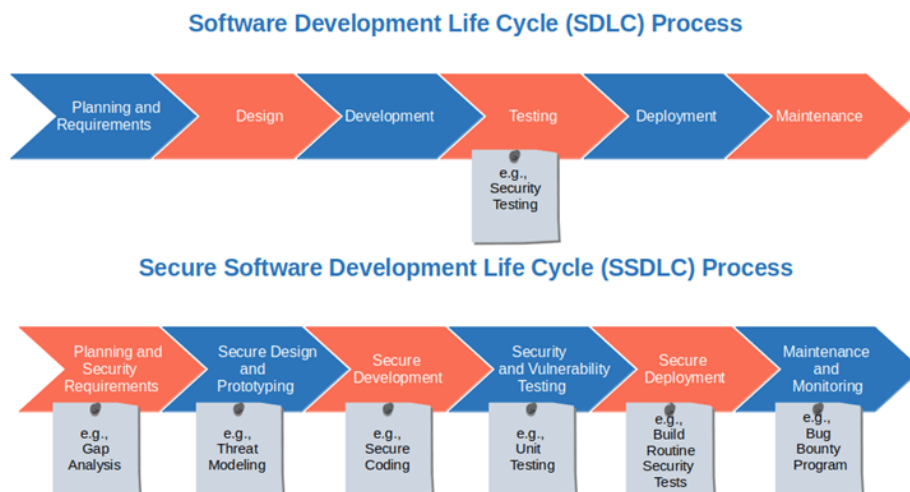


Figura 1 - <https://codesigningstore.com/wp-content/uploads/2023/02/secure-software-development-life-cycle-process.png>

Com esta metodologia procura-se garantir que os riscos da segurança sejam identificados e mitigados o mais cedo possível. Assim sendo, reduz-se significativamente o custo e impactos resultantes de correções posteriores e simultaneamente garante-se uma maior confiança no produto. Ao realizar-se análises de segurança e revisões contínuas durante todo o ciclo, é possível detetar e intervir sobre potenciais incidentes de segurança.

Este processo é fundamental para o desenvolvimento de software atual, uma vez que, ao contemplar todas as suas fases integrantes (análise, design, desenvolvimento, testes, build e deployment), visa tornar a aplicação o mais segura possível. A sua adoção permite alcançar vários benefícios importantes:

- **Redução de custos:** através da identificação antecipada de possíveis vulnerabilidades no sistema, que reduz 30 vezes menos o custo do que se fossem tratadas essas vulnerabilidades durante a fase produção;
- **Security-first:** promove uma cultura orientada à segurança, tornando as equipas mais conscientes e preparadas para desenvolver software com medidas de proteção desde o início;
- **Estratégia de desenvolvimento bem definida:** ao estabelecer desde o início os critérios de segurança da aplicação, decisões como a escolha de tecnologias ou arquitetura tornam-se mais informadas e fundamentadas.

A aplicação desenvolvida, designada por '**EcolImpact Simulator**', visa promover a consciencialização ambiental através da monitorização e avaliação dos hábitos individuais com impacto ecológico, que visa incentivar hábitos de vida mais sustentáveis. No entanto, mais do que o produto final, o foco deste relatório incide sobre o processo seguido, com ênfase especial na identificação de ameaças, definição de requisitos de segurança, implementação de controlos adequados e validação da conformidade com a norma *OWASP Application Security Verification Standard (ASVS) v4.0.3 – Nível 2*.

Foi sido assumido o cenário da sua aplicação num âmbito exclusivamente empresarial, e que serviria de forma de avaliação se as campanhas e eventos de consciencialização que a empresa desenvolve nesta temática estão a surtir numa alteração e consciencialização nas opções de dia-a-dia dos seus

colaboradores. Desta forma, resolvem-se questões relacionadas com a proteção de dados pessoais, ainda que o objetivo seja a não inclusão de dados privados e o seu preenchimento é voluntário e de dados genéricos (e não comprováveis) de comportamentos.

O trabalho apresentado integra ainda os conhecimentos adquiridos nas aulas teóricas da unidade curricular, estando estruturado de forma a evidenciar a progressão lógica desde o planeamento até à verificação final da segurança da aplicação.

Descrição do Projeto

Os objetivos gerais do projeto são implementar uma aplicação funcional com preocupações de segurança, transversais ao ciclo de vida de desenvolvimento. Sendo o objetivo da cadeira a implementação de frameworks de desenvolvimento de software seguro, o design e funcionalidades não vitais foram descurados tendo o enfoque direcionado às passíveis de validação da aplicação correta da framework.

Um dos princípios basilares foi promover a análise e mitigação de riscos desde as fases iniciais do projeto.

De acordo com o projeto, as funcionalidades a implementar incluem:

Categoria	Funcionalidade
Autenticação e Gestão de Sessões	Registo e login de utilizadores com gestão segura de sessões
Interação com o Utilizador	Visualização de hábitos, submissão de escolhas, apresentação de resultados
Gestão de Conteúdos	Inserção, edição e eliminação de tipos de hábitos por utilizadores autorizados
Registo de Atividades	Logging de ações relevantes para efeitos de auditoria e deteção de abuso

Os utilizadores são diferenciados pelas permissões que lhe estão atribuídas:

Tipo de Utilizador	Descrição
Utilizador Registado	Pode submeter escolhas e visualizar resultados
Moderador	Pode adicionar, editar e eliminar tipos de hábitos
Administrador	Permissões de moderador + gestão de utilizadores e análise de logs

E a execução do projeto, respeitou 3 fases de entregas, conforme definido no enunciado:

Fase	Conteúdo Principal
Phase 1	Análise de Requisitos, Modelação de Ameaças, Arquitectura e Design Inicial
Phase 2: Sprint 1	Prototipagem, Design Detalhado, Implementação Inicial, Casos de Teste
Phase 2: Sprint 2	Implementação Final, Testes Funcionais e de Segurança, Validação e Entrega Final

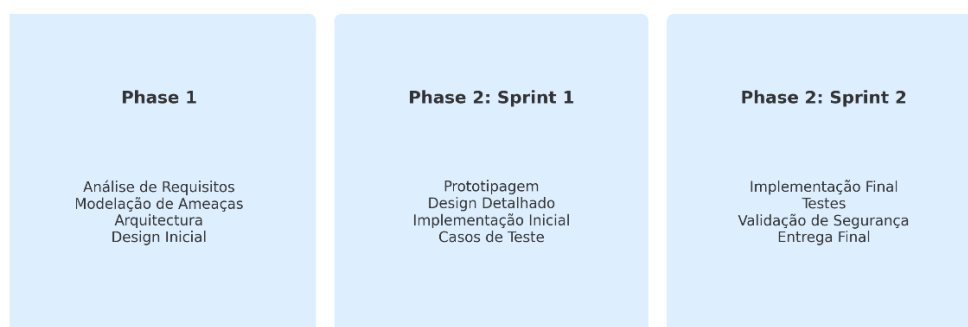


Figura 2 - fases do projeto

Processo SSDLC Adotado

A aplicação do processo SSDLC, foi adotado como metodologia que procura garantir que as decisões relacionadas com a segurança fossem tomadas desde a fase inicial de conceção até à validação final do sistema.

A estrutura do processo SSDLC adotado, resume-se no diagrama:

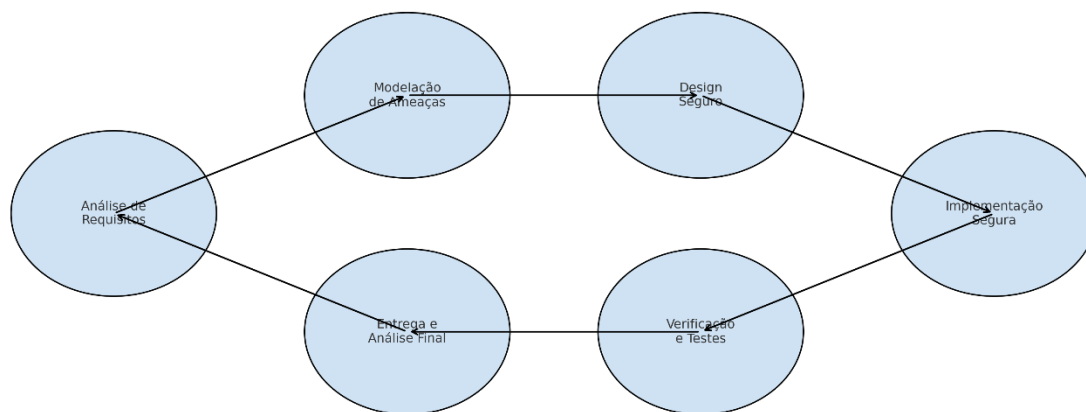


Figura 3 - diagrama metodologia SSDLC

Tendo-se dividido o processo em 6 fases:

Fase	Atividades Principais
1. Análise de Requisitos	Identificação de requisitos funcionais e de segurança. Recolha de dados através do enunciado e modelação de perfis de utilizador.
2. Modelação de Ameaças	Aplicação da metodologia STRIDE para identificação e mitigação de riscos. Análise de abusos e fluxos alternativos.
3. Design Seguro	Definição da arquitetura lógica e física do sistema. Integração de práticas como RBAC, DTOs e validação de inputs.
4. Implementação Segura	Desenvolvimento com boas práticas de programação segura, utilização de autenticação JWT, logging e tratamento de erros.
5. Verificação e Testes	Testes funcionais e de segurança. Validação da conformidade com os requisitos do ASVS Nível 2.
6. Entrega e Análise Final	Consolidação dos testes, avaliação de riscos remanescentes e geração do relatório final.

Requisitos do Sistema

A definição de requisitos do sistema, permite identificar de forma clara as funcionalidades esperadas e os aspetos críticos de segurança da aplicação.

Modelo de Domínio

O modelo de domínio representa as entidades principais do sistema, as suas propriedades e as relações entre elas. Este modelo foi elaborado com base nos requisitos funcionais e na estrutura de dados existente no projeto, refletindo a lógica de negócio da aplicação

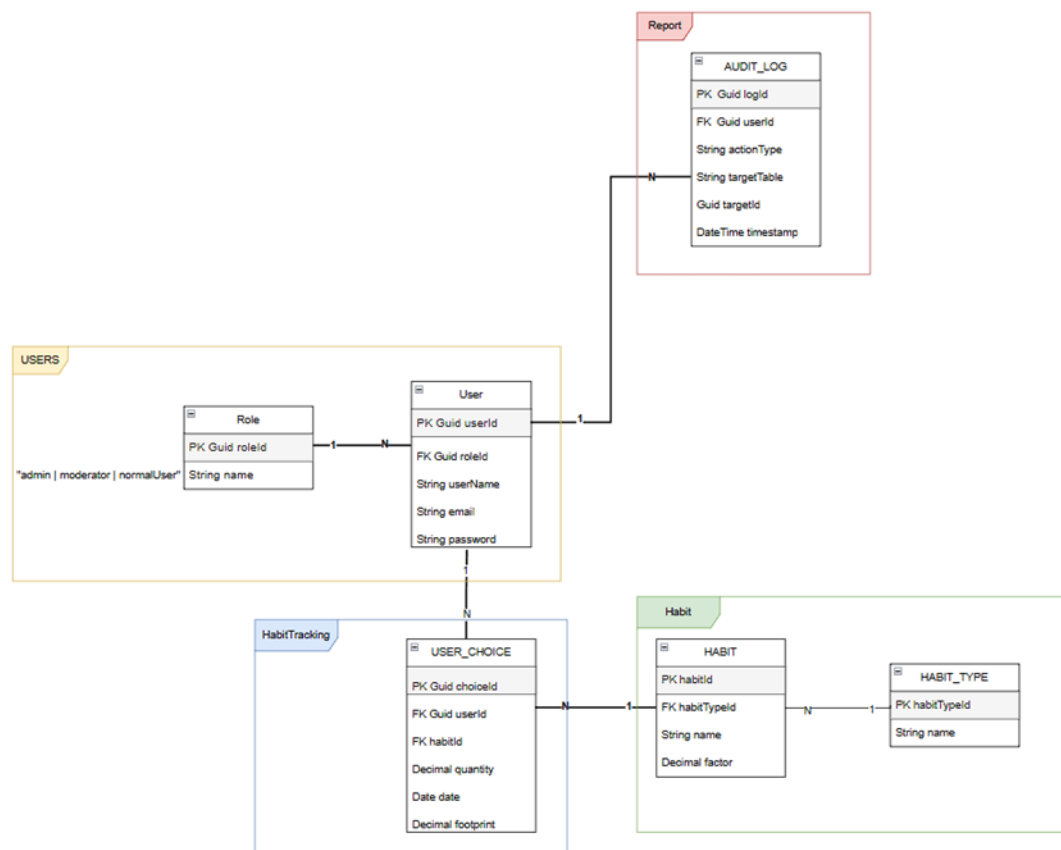


Figura 4 - Modelo de Domínio

No presente projeto, o domínio encontra-se organizado em quatro agregados fundamentais, cada um responsável por um conjunto de funcionalidades bem definido:

Users: Este agregado encarrega-se de toda a lógica de autenticação e autorização, bem como do armazenamento seguro dos dados pessoais dos utilizadores.

A classe **Role** define os diferentes perfis de acesso do sistema (Admin, Moderator, NormalUser), enquanto a classe **User** representa cada conta individual com os respetivos dados de login (username, email, password) e o perfil de acesso associado (roleId). Cada utilizador pode estar associado a várias ações, tanto em termos de registos de atividade (USER_CHOICE) como de operações auditadas (AUDIT_LOG).

HabitTracking: Este agregado é responsável pelo registo das escolhas ambientais feitas diariamente pelos utilizadores. A classe **USER_CHOICE** representa cada ação realizada, contendo como chave primária o identificador da escolha (choiceId) e referenciando diretamente o utilizador (userId) e o hábito selecionado (habitId). Para cada escolha são ainda registados a quantidade (quantity) associada à ação, a data em que foi realizada (date) e o valor calculado da pegada de carbono (footprint). Este valor resulta da multiplicação da quantidade pelo fator de emissão do hábito, permitindo assim avaliar o impacto ambiental individual de cada ação.

Habit: Este agregado organiza os hábitos ambientais monitorizáveis disponíveis no sistema. A classe **HABIT** representa uma ação específica (como “andar de carro” ou “comer carne”), identificada pelo campo habitId, e contém um fator de emissão (factor) que permite calcular a pegada de carbono associada à ação. Cada hábito está associado a uma categoria geral representada pela classe **HABIT_TYPE**, através da chave estrangeira habitTypeId. Esta categorização permite agrupar os hábitos por áreas de impacto ambiental, como “Transporte” ou “Alimentação”, facilitando a análise e visualização dos dados recolhidos.

Report: Este agregado é responsável por garantir a rastreabilidade das operações sensíveis realizadas no sistema. A classe **AUDIT_LOG** regista cada evento com um identificador único (logId) e armazena informações essenciais como o tipo de ação (actionType), a tabela visada (targetTable), o identificador do alvo (targetId), o utilizador que realizou a operação (userId) e o carimbo temporal da

ocorrência (timestamp). Estes registos permitem acompanhar e auditar atividades críticas, sendo acessíveis apenas por utilizadores com privilégios elevados. O campo `userId` liga cada entrada de log ao utilizador que a gerou.

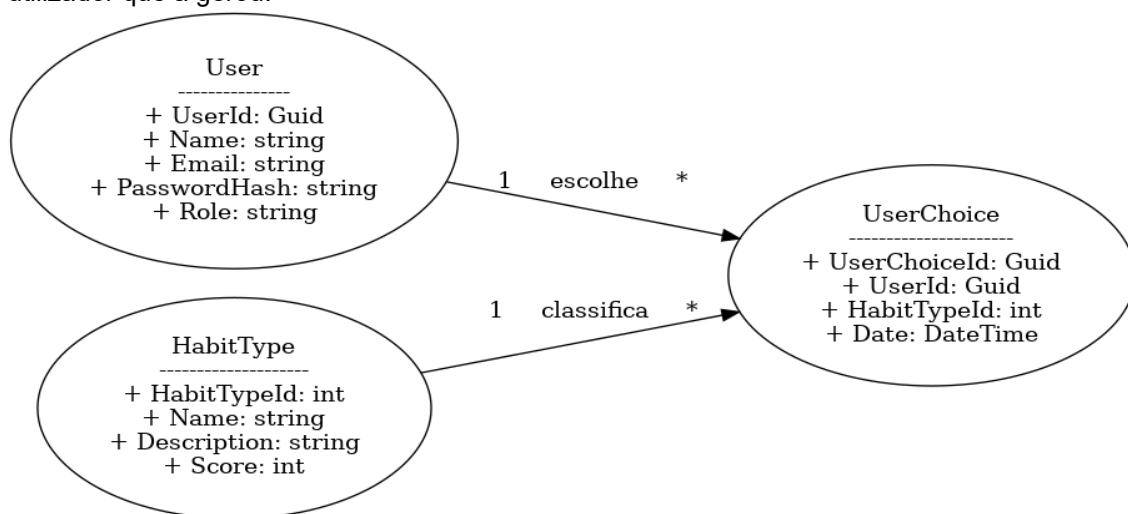


Figura 5 - modelo UML de Domínio

Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que a aplicação deve oferecer aos diferentes tipos de utilizador. Foram organizados por módulo ou área de atuação:

Código	Requisito Funcional
RF01	O sistema deve permitir o registo e autenticação de utilizadores.
RF02	O utilizador autenticado deve poder visualizar os tipos de hábitos existentes.
RF03	O utilizador autenticado deve poder submeter escolhas de hábitos.
RF04	O sistema deve gerar um relatório de impacto com base nas escolhas submetidas.
RF05	Utilizadores autorizados (moderador/admin) devem poder criar, editar e eliminar tipos de hábitos.
RF06	O administrador deve ter acesso a registos de auditoria e controlo de acessos.

Requisitos Não Funcionais

Os Requisitos Não-Funcionais (RNFs) garantem que a aplicação cumpre critérios de qualidade como segurança, desempenho, usabilidade e conformidade.

Código	Requisito Não Funcional
RNF01	O sistema deve cumprir os requisitos definidos pelo OWASP ASVS v4.0.3 – Nível 2.
RNF02	A aplicação deve responder às interações do utilizador com tempo de resposta inferior a 2 segundos.
RNF03	O sistema deve escalar horizontalmente para suportar aumento de carga sem degradação significativa.
RNF04	A interface deve ser intuitiva e adaptada a diferentes dispositivos (mobile e desktop).
RNF05	O sistema deve manter a integridade dos dados mesmo em caso de falhas inesperadas.
RNF06	As ações sensíveis devem ser registadas (log) para efeitos de auditoria e deteção de comportamentos abusivos.
RNF07	A aplicação deve estar em conformidade com o RGPD, incluindo direitos como o esquecimento e exportação de dados.

Requisitos de Segurança

Os requisitos de segurança foram definidos tendo por base em boas práticas de engenharia segura e mapeamento com a norma OWASP ASVS.

Código	Requisito de Segurança
RS01	O sistema deve utilizar autenticação segura (tokens JWT) com expiração e revogação.
RS02	As permissões devem ser geridas por níveis de acesso (RBAC).

RS03	Todos os inputs do utilizador devem ser validados e/ou sanitizados para evitar injeções (SQLi, XSS).
RS04	As passwords devem ser armazenadas de forma cifrada e com salt (ex: bcrypt).
RS05	Os acessos e alterações sensíveis devem ser registados (logs de auditoria).
RS06	O sistema deve utilizar HTTPS para toda a comunicação.
RS07	A importação de ficheiros deve verificar tipo MIME e tamanho permitido.

Com base no risco e na exposição aos utilizadores externos, os requisitos de segurança foram priorizados:

Prioridade	Requisitos Associados
Alta	RS01, RS02, RS03, RS04
Média	RS05, RS06
Baixa	RS07

Casos de Uso

A modelação de casos de uso tem como objetivo identificar as interações principais entre os utilizadores e o sistema, bem como descrever os serviços que a aplicação disponibiliza. Estes casos foram definidos com base nos requisitos funcionais anteriormente identificados.

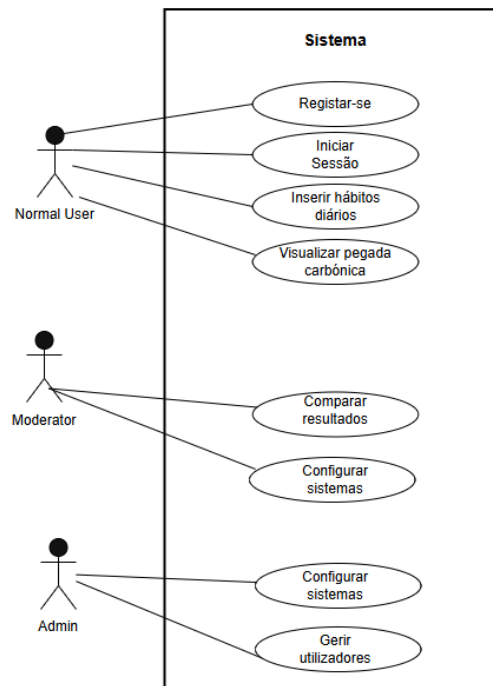


Figura 6 - diagrama de casos de uso

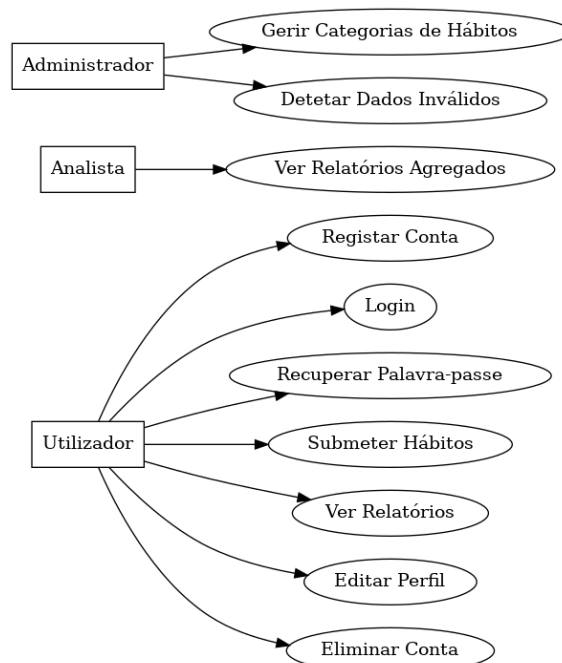


Figura 7 - diagrama de casos de uso

Nas figuras 6 e 7 representa-se os casos de uso global da aplicação, onde se destacam os diferentes atores e as funcionalidades disponíveis consoante os seus privilégios de acesso.

Numa tabela, descreve-se sumariamente os principais casos de uso do sistema:

Código	Caso de Uso	Ator(es)	Descrição resumida
UC01	Registar Conta	Utilizador	Permite ao utilizador criar uma conta com dados pessoais e palavra-passe.
UC02	Autenticar-se (Login)	Utilizador	Permite o login no sistema com verificação JWT.
UC03	Recuperar Palavra-passe	Utilizador	Inicia processo de recuperação via e-mail seguro.
UC04	Submeter Hábitos	Utilizador	Submete dados diários sobre transporte, alimentação, energia e consumo.
UC05	Consultar Relatórios	Utilizador	Visualiza relatórios da pegada de carbono, com histórico.
UC06	Editar Perfil	Utilizador	Permite alterar nome, e-mail e palavra-passe.
UC07	Eliminar Conta	Utilizador	Solicita remoção total da conta e dados associados.
UC08	Gerir Categorias de Hábitos	Administrador	Adiciona, edita ou remove categorias de hábitos.
UC09	Ver Relatórios Agregados	Analista	Consulta estatísticas da comunidade sem acesso a dados identificáveis.
UC10	Detetar Dados Inválidos	Sistema / Administrador	Identifica e alerta para padrões de dados incoerentes ou anómalos.

Threat Modeling

A identificação e análise de ameaças é um dos pilares fundamentais do desenvolvimento seguro de software. O principal objetivo do processo de Threat Modeling é, de forma sistemática, identificar potenciais ameaças e vulnerabilidades no sistema, garantindo que estas sejam tratadas de forma proativa antes que possam ser exploradas.

Para realizar esta análise, é essencial ter um conhecimento profundo da arquitetura do sistema, dos componentes, dos fluxos de dados e das relações de confiança entre as diferentes partes que compõem a aplicação.

A modelação de ameaças foi realizada com base na metodologia **STRIDE**, desenvolvida pela Microsoft, e aplicada aos principais componentes do sistema. Este modelo permite identificar potenciais ameaças de segurança agrupadas em seis categorias: *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service* e *Elevation of Privilege*.

Além da identificação, é também realizada a análise de vulnerabilidades que possam ser exploradas, como falhas de implementação, más configurações ou decisões de design inseguras. O Threat Modeling não é tratado como uma atividade isolada, mas sim como parte integrante do Secure Software Development Life Cycle (SSDLC). Este processo é contínuo e iterativo, sendo reaplicado sempre que ocorrem alterações significativas na aplicação, como:

- Desenvolvimento de novas funcionalidades
- Alterações na arquitetura
- Integração de novas dependências



Figura 8 - Threat Modeling Process

Decompondo o modelo em fases:

1. Identify Potential Threats and Vulnerabilities

Na primeira fase, são mapeadas as potenciais ameaças e vulnerabilidades através da análise:

- Da arquitetura do sistema;
- Dos fluxos de dados;
- Dos limites de confiança;
- Dos componentes e suas interações.

Aplica-se o modelo STRIDE para garantir uma categorização ampla das ameaças, cobrindo diferentes vetores de ataque e cenários de exploração.

2. Assess Impact and Likelihood

Cada ameaça identificada é avaliada quanto ao seu impacto e à probabilidade de ocorrência. Para essa avaliação, foi utilizado o modelo DREAD, que permite uma quantificação objetiva baseada nos seguintes critérios:

- Damage Potential (Potencial de dano)
- Reproducibility (Reprodutibilidade)
- Exploitability (Explorabilidade)
- Affected Users (Utilizadores afetados)
- Discoverability (Descoberta da vulnerabilidade)

Esta análise permite priorizar as ameaças e focar os esforços de mitigação nas mais críticas.

3. Develop Mitigation Strategies

Com base nos riscos identificados e priorizados, são definidas estratégias de mitigação adequadas. Estas podem incluir:

- Fortalecimento dos mecanismos de autenticação e controlo de acessos;
- Encriptação de dados sensíveis em repouso e em trânsito;
- Limitação de taxas de acesso (rate limiting) e aplicação de throttling;

- Implementação de sistemas de detecção e prevenção de intrusões (IDS/IPS);
- Aplicação de políticas de configuração segura e atualização contínua de componentes.

4. Implement and Monitor

Após a definição das estratégias, as mesmas são implementadas na aplicação. A eficácia destas medidas é garantida através de:

- Monitorização contínua;
- Centralização e análise de logs de segurança;
- Auditorias regulares;
- Detecção ativa de novas ameaças e vulnerabilidades.

Sempre que são detetadas alterações no ambiente ou novas ameaças, o processo é reiniciado, reforçando a abordagem dinâmica e adaptativa do Threat Modeling.

Este ciclo iterativo e contínuo, aliado à utilização de modelos como STRIDE e DREAD, permite garantir que o projeto se mantém seguro ao longo de todo o ciclo de vida de desenvolvimento.

Modelação aplicada ao sistema

A modelação foi aplicada aos principais componentes do sistema:

- Autenticação
- API de Submissão de Hábitos
- Base de Dados
- Gestão de Utilizadores
- Geração de Relatórios

Ameaças identificadas

Componente	Categoria STRIDE	Ameaça Identificada
Autenticação	<i>Spoofing</i>	Possibilidade de uso de credenciais roubadas ou JWT falsificado
Autenticação	<i>Repudiation</i>	Utilizador negar ter submetido ações críticas sem logging adequado
API Submissão Hábitos	<i>Tampering</i>	Manipulação maliciosa dos dados em trânsito se TLS não for forçado
API Submissão Hábitos	<i>DoS</i>	Envio massivo de submissões para esgotar recursos do servidor
Base de Dados	<i>Information Disclosure</i>	Exposição indevida de dados sensíveis por má configuração ou injeção SQL
Gestão de Utilizadores	<i>Elevation of Privilege</i>	Utilizador não autorizado aceder a funcionalidades administrativas
Geração de Relatórios	<i>Repudiation</i>	Falta de rastreabilidade nos acessos a relatórios ou manipulação de métricas

Mitigações aplicadas

Ameaça STRIDE	Medida de Mitigação Aplicada
Spoofing (JWT falsificado)	Assinatura de tokens com chave secreta forte e curta expiração
Tampering (dados em trânsito)	Utilização obrigatória de HTTPS em todas as comunicações
Repudiation	Registo de logs de ações críticas com timestamp e ID do utilizador
Information Disclosure	Validação e sanitização de inputs + controlo de permissões por função (RBAC)
Denial of Service	Limitação de requests por IP (rate-limiting) e logging de acessos suspeitos
Elevation of Privilege	Verificação de permissões a cada endpoint; testes específicos de escalamento

Diagrama de Fluxo de Dados (DFD)

DFD nível 0

De forma a representar de maneira clara a interação entre os diferentes intervenientes e o sistema, foi desenvolvido o Diagrama de Fluxo de Dados (DFD) de Nível 0, também conhecido como Context Diagram.

Este diagrama permite visualizar, de forma simplificada, os fluxos de dados entre os utilizadores (Admin, Moderator e Normal User) e o sistema Simulador de Impacto Ambiental, assim como a relação deste com a base de dados.

O DFD Nível 0 descreve a aplicação como uma "caixa preta", onde são evidenciados apenas os fluxos de entrada e saída de dados, sem detalhar os processos internos, garantindo uma visão global das interações externas do sistema.

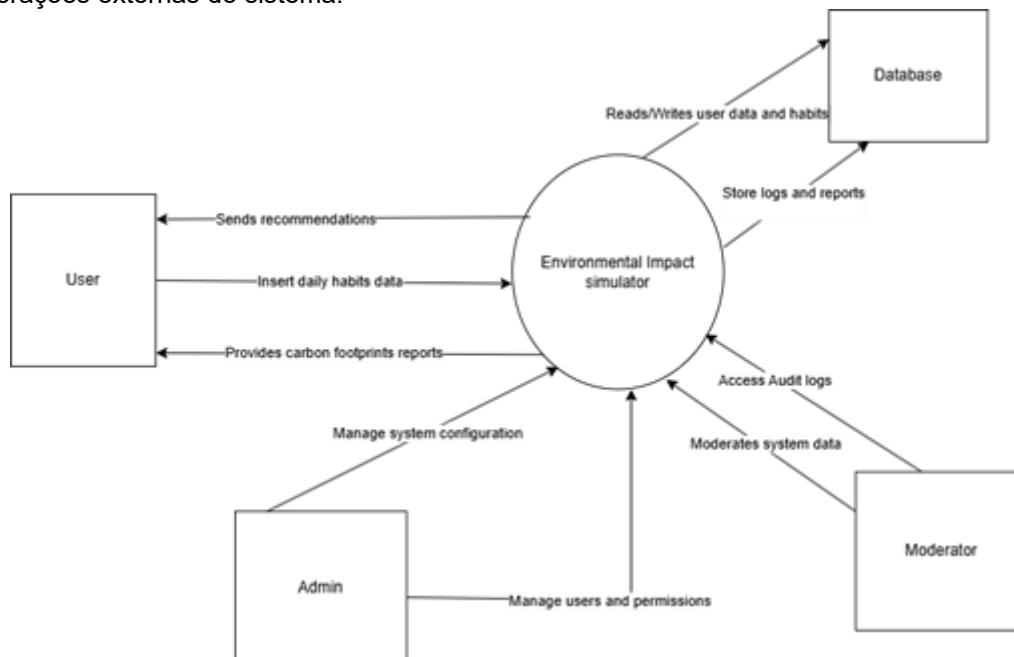


Figura 9 - DFD nível 0

DFD nível 1

De modo a representar de forma mais detalhada as interações internas do sistema, foi desenvolvido o **Diagrama de Fluxo de Dados (DFD) de Nível 1**.

Este diagrama descreve a comunicação entre os diferentes intervenientes (Admin, Moderator e Users) e o sistema Simulador de Impacto Ambiental, representando agora também a divisão entre os principais componentes internos: **Backend Server**, **Database** e **File API**.

O **Backend Server** atua como o núcleo do sistema, recebendo pedidos dos utilizadores através do frontend e interagindo com a base de dados e a File API conforme necessário para o tratamento dos pedidos.

- Os **Users** (utilizadores normais) enviam Requests para o backend, nomeadamente para inserção de hábitos diários ou consulta de pegada carbónica, recebendo como resposta os resultados ou sugestões.
- O **Moderator** interage de forma semelhante, enviando Requests mais orientados para a comparação de resultados ou análise de dados, recebendo Responses com os relatórios ou visualizações pretendidas.
- O **Admin** realiza ações de configuração e gestão de utilizadores, enviando Requests para o backend e recebendo Responses de confirmação ou resultados das operações efetuadas.

A comunicação com a **Database** ocorre quando o backend necessita de:

- Armazenar novos dados submetidos (hábitos diários, configurações);
- Ler dados para cálculos de pegada de carbono ou geração de relatórios;

Em operações específicas de gestão de ficheiros (por exemplo, exportação de relatórios ou uploads de documentos), o backend comunica com a **File API**, respeitando uma fronteira de confiança separada.

O DFD evidencia ainda as **trust boundaries** críticas:

- Entre os utilizadores e o backend (User/Backend Boundary);
- Entre o backend e a base de dados (Backend/Database Boundary);
- Entre o backend e a API de ficheiros (Backend/File API Boundary).

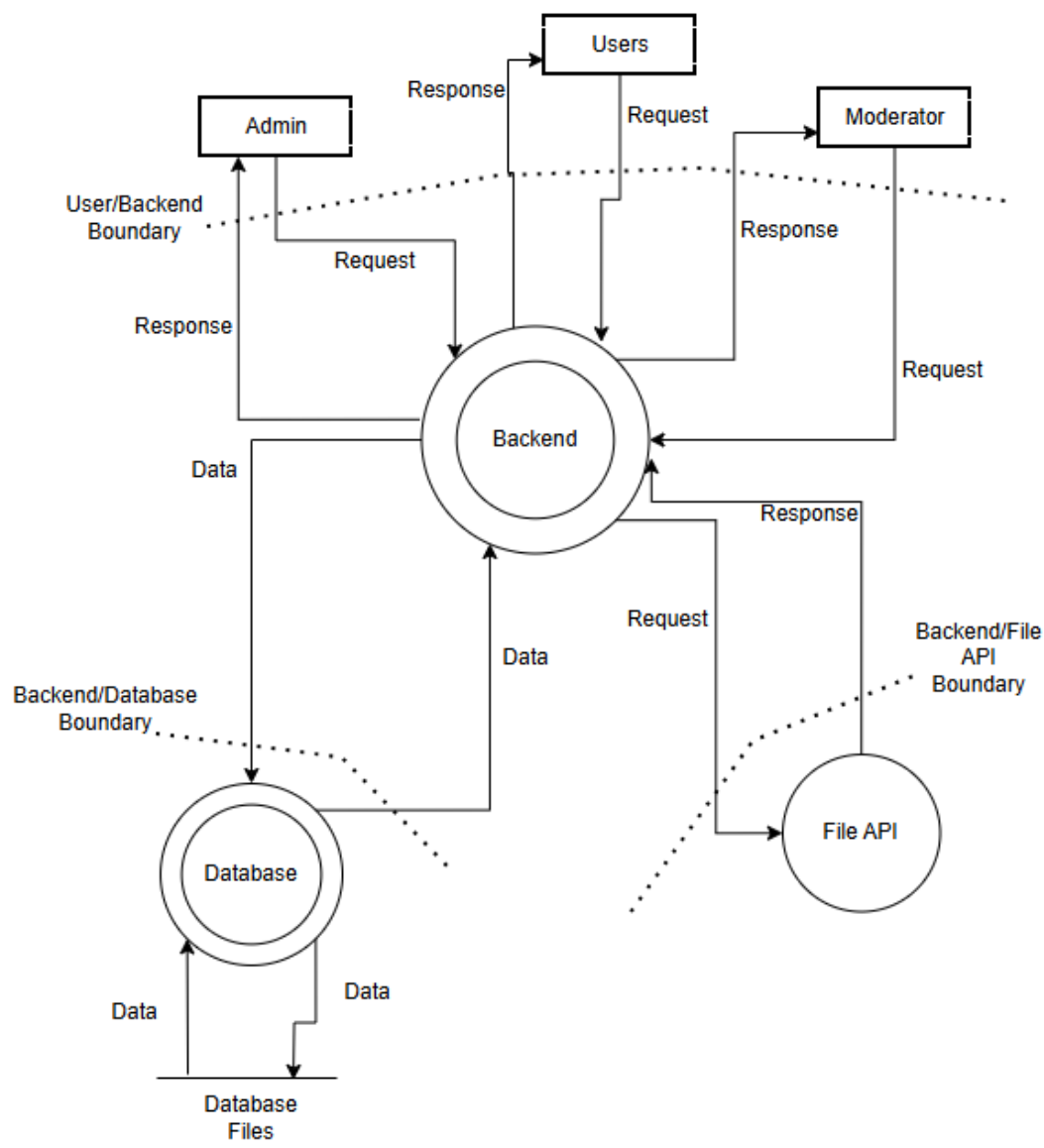


Figura 10 - DFD nível 1

DFD nível 2

O Diagrama de Fluxo de Dados (DFD) de Nível 2 representa, de forma detalhada, os subprocessos envolvidos na **submissão dos hábitos diários** por parte do utilizador e no subsequente **cálculo da pegada de carbono**. Esta decomposição foca-se na lógica interna do processo identificado anteriormente no DFD de Nível 1 como “Submissão de Hábitos” e “Cálculo da Pegada”.

O processo inicia-se com o **utilizador a submeter dados de hábitos**, como informações sobre transporte, consumo energético, alimentação e compras. Estes dados são primeiramente tratados no subprocesso **Validar Dados de Submissão**, que assegura que os dados estão completos e formatados corretamente.

Em seguida, os dados validados são encaminhados para o subprocesso **Armazenar Dados**, que se responsabiliza por **guardar os registos na base de dados**. Estes dados servem de base para o cálculo, pelo que são imediatamente utilizados no subprocesso **2Calcular Pegada de Carbono**, que aplica os fatores de conversão definidos para estimar a quantidade de CO₂ equivalente gerado.

Com base nesse cálculo, o subprocesso **Atualizar Relatórios** prepara visualizações e relatórios para o utilizador, além de guardar os resultados na base de dados. Paralelamente, o subprocesso

Registrar Ação no Log garante o registo de auditoria, mantendo a rastreabilidade da submissão e assegurando o cumprimento de requisitos de segurança e transparência. Este nível de detalhe permite demonstrar um conhecimento mais profundo sobre os fluxos de dados sensíveis e a sua **proteção ao longo do ciclo de processamento**, sendo uma base importante para a identificação de ameaças e análise de conformidade com os requisitos do **OWASP ASVS**.

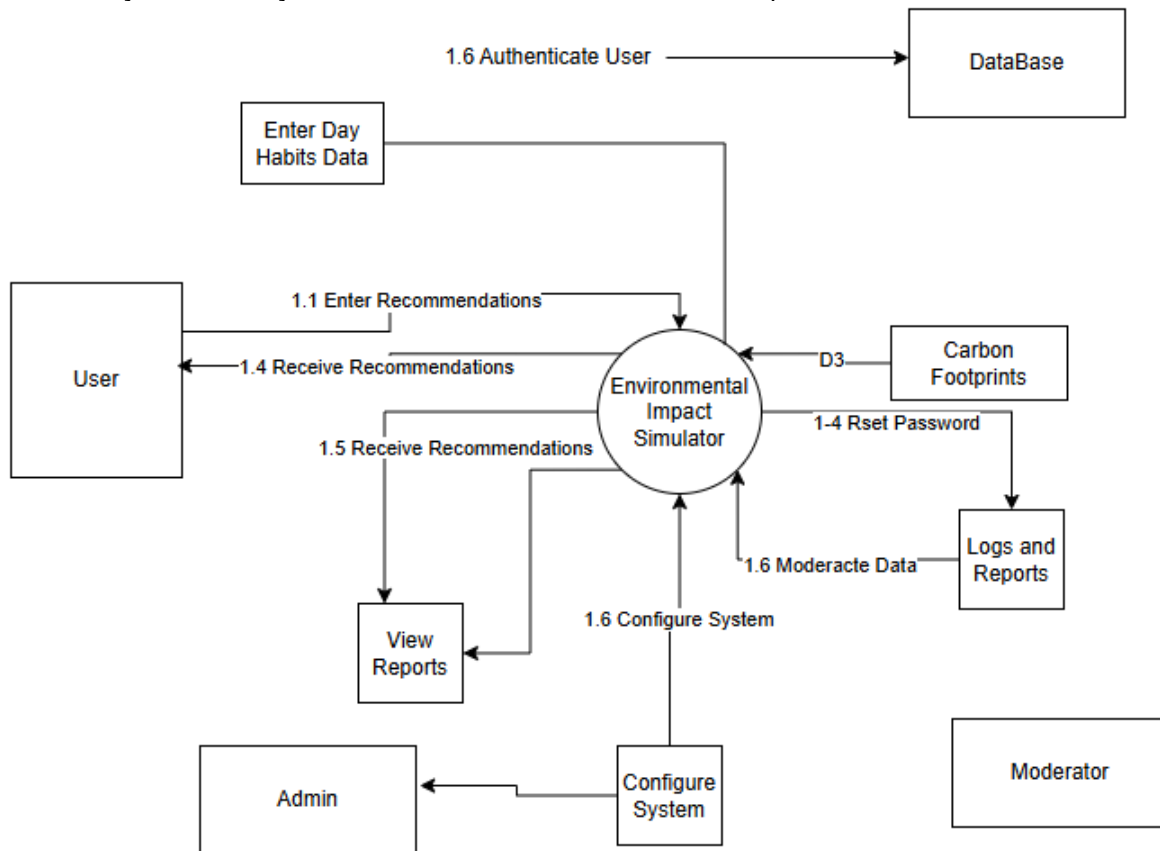


Figura 11 - DFD nível 2

Arquitetura do Sistema

A arquitetura da aplicação foi concebida com base numa abordagem modular e orientada a serviços, garantindo separação de responsabilidades, segurança e escalabilidade. Esta secção descreve os principais componentes da arquitetura, a sua organização lógica e as tecnologias envolvidas.

A aplicação segue uma arquitetura cliente-servidor, composta por três camadas principais:

- Frontend: Interface gráfica desenvolvida em React, consumindo serviços expostos via API REST.
- Backend: Serviço ASP.NET Core responsável pela lógica de negócio, autenticação e gestão de dados.
- Base de Dados: Motor relacional (SQLite) utilizado para persistência dos dados dos utilizadores e hábitos registados.

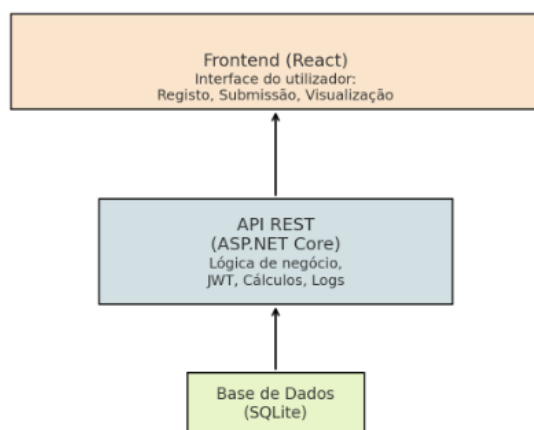


Figura 12 - Arquitetura Lógica da Aplicação

Componente	Descrição
React Frontend	Interface de utilizador que permite registo, submissão de hábitos e consulta de relatórios.
API ASP.NET Core	Camada de serviços expostos via REST, incluindo autenticação JWT e lógica de cálculo da pegada.
Base de Dados	Armazena utilizadores, hábitos, resultados e configurações de sistema. Utiliza SQLite.

A arquitetura integra várias medidas de segurança desde a fase de conceção:

- Comunicação encriptada por HTTPS (TLS 1.2+)
- Autenticação baseada em JWT com expiração e verificação de permissões
- Sanitização e validação de todos os inputs recebidos
- Registo (logging) das ações sensíveis e acessos

Ambientes de Execução e Deploy

A aplicação EcolImpact foi desenhada para operar em dois ambientes principais:

- Ambiente de Produção (**Render.com**): cada componente (frontend em *Blazor WebAssembly*, backend *ASP.NET Core*, e base de dados *PostgreSQL*) é executado em containers distintos com URLs públicos separados. O deploy é automatizado com *GitHub Actions* e tokens seguros guardados como *secrets* no *GitHub*.
- Ambiente de Desenvolvimento (**Docker Compose**): simula localmente toda a stack. Utiliza ficheiros *.env* (excluídos do controlo de versão) para gerir variáveis sensíveis como JWT secret e strings de ligação. A execução é feita com `docker compose --env-file .env up --build`.

Esta estratégia garante portabilidade, segurança e isolamento entre ambientes.

OWASP ASVS Nível 2

O **OWASP Application Security Verification Standard (ASVS)** é uma framework aberta e reconhecida internacionalmente, desenvolvida pela comunidade OWASP (Open Worldwide Application Security Project), que define um conjunto de critérios para a verificação de segurança em aplicações. O seu principal objetivo é fornecer uma base sistemática e mensurável para avaliar o nível de segurança implementado numa aplicação, promovendo o desenvolvimento seguro desde as fases iniciais do ciclo de vida do software.

O ASVS encontra-se estruturado em **14 categorias**, que abrangem diferentes domínios da segurança aplicacional, desde a **gestão de autenticação e sessões**, até ao **controlo de acesso, encriptação, validação de entradas, gestão de erros, e proteção contra ataques automatizados**. Estas categorias estão organizadas por níveis de verificação, sendo o **Nível 1** o mais básico (aplicações públicas sem tratamento de dados sensíveis), o **Nível 2** o recomendado para aplicações típicas com informação pessoal ou sensível, e o **Nível 3** destinado a aplicações críticas ou contendo dados sensíveis.

Neste projeto foi adotado o **Nível 2 do OWASP ASVS**, por se tratar de uma aplicação que envolve dados pessoais dos utilizadores (como nome, email, hábitos de consumo) e realiza cálculos

personalizados que podem influenciar comportamentos individuais, contudo não são tratados dados sensíveis e a informação é preenchida pelo utilizador (sem confirmação da sua veracidade). A conformidade com este nível permite assegurar que foram tidas em consideração as **melhores práticas de segurança**, alinhadas com os **princípios do Secure Software Development Life Cycle (SSDLC)**, contribuindo para um sistema mais robusto, resiliente e confiável.

Metodologia adotada

Para avaliar a conformidade da aplicação com os controlos definidos no **OWASP Application Security Verification Standard (ASVS) Nível 2**, foi adotada uma abordagem sistemática de mapeamento entre os requisitos de segurança estabelecidos pela norma e as funcionalidades desenvolvidas no âmbito do projeto.

A metodologia seguiu os seguintes passos:

1. Análise das Categorias da ASVS

Foi realizada uma revisão detalhada das 14 categorias que compõem a ASVS 4.0.3, com o objetivo de identificar os controlos mais relevantes para a natureza da aplicação — nomeadamente, os que envolvem gestão de autenticação, controlo de acessos, validação de dados e proteção de comunicações.

2. Mapeamento de Requisitos com Funcionalidades

Cada requisito técnico da ASVS foi analisado à luz das **funcionalidades existentes ou previstas** no projeto. Sempre que aplicável, foi estabelecida uma correspondência direta com os **requisitos funcionais (RF)** definidos anteriormente, permitindo assim justificar o cumprimento dos controlos.

3. Verificação Técnica e Evidências

Para cada controlo identificado como relevante, procurou-se garantir a existência de uma **implementação técnica ou arquitetónica que respondesse ao objetivo de segurança proposto**. Estas evidências incluem trechos de código, configuração de bibliotecas, fluxos de autenticação e práticas seguras de desenvolvimento documentadas no relatório.

4. Classificação de Conformidade

A conformidade foi avaliada em três níveis:

- **Cumprido (✓)**: O controlo está plenamente implementado ou verificado.
- **Parcialmente cumprido (~)**: Existem medidas em curso, mas ainda há aspetos a melhorar.
- **Não cumprido (X)**: O controlo não está ainda implementado ou foi considerado fora de âmbito.

Esta metodologia visa não só demonstrar a adoção das boas práticas promovidas pelo OWASP ASVS, como também identificar **possíveis lacunas** que podem ser objeto de melhoria contínua, em linha com os princípios do ciclo de vida seguro de desenvolvimento (SSDLC).

Categoria	Código ASVS	Descrição	Estado	Evidência / Implementação
V2 - Autenticação	2.1.1	Utilização de identificadores únicos por utilizador	✓	Registo com email único e ID no sistema
V2 - Autenticação	2.1.5	As palavras-passe devem ser armazenadas de forma segura	✓	Utilização de hashing com algoritmo seguro
V3 - Gestão de Sessões	3.1.1	Utilização de tokens de sessão seguros (JWT)	✓	JWT implementado no backend para gestão de sessões
V5 - Validação de Entrada	5.1.2	Validação de dados do lado do servidor	✓	Back-end valida campos de hábitos antes de guardar
V7 - Controlo de Acessos	7.1.1	Controlo de acesso baseado em funções (RBAC)	✓	Definidos papéis: Admin, Moderador, Utilizador
V9 - Proteção de Dados	9.1.1	Transmissão de dados protegida com TLS	✓	Comunicação entre frontend e backend com HTTPS
V10 - Registo e Monitorização	10.2.1	Registo de eventos de submissão e gestão de utilizadores	✓	Logs de submissão e auditoria no sistema
V11 - Tratamento de Erros	11.1.1	Mensagens de erro não devem revelar informação sensível	✓	Mensagens genéricas implementadas no frontend/backend

As evidências referidas foram recolhidas com base na análise do código-fonte, configuração da infraestrutura e comportamento funcional da aplicação durante testes.

- **Autenticação e Armazenamento Seguro de Palavras-passe (ASVS 2.1.1, 2.1.5):**
O sistema implementa um mecanismo de registo de utilizadores baseado em identificadores únicos (endereços de email). As palavras-passe são armazenadas utilizando hashing seguro, garantindo que nunca são guardadas em texto simples. A verificação posterior de credenciais é feita com base na comparação de hashes, assegurando a confidencialidade dos dados.
- **Gestão de Sessões com JWT (ASVS 3.1.1):**
Após o login, é emitido um token JWT assinado que representa a sessão ativa do utilizador. Estes tokens contêm as permissões associadas ao perfil de utilizador e são utilizados para autenticar pedidos subsequentes, sem necessidade de reenviar credenciais.
- **Validação de Dados no Backend (ASVS 5.1.2):**
Todos os dados submetidos, nomeadamente os relativos a hábitos ambientais, são validados no backend quanto ao tipo, formato e intervalo de valores. Esta validação protege o sistema contra dados maliciosos e entradas incorretas que poderiam comprometer o cálculo da pegada de carbono.
- **Controlo de Acessos Baseado em Perfis (ASVS 7.1.1):**
A aplicação distingue três perfis de utilizador (Admin, Moderador, Utilizador Padrão), com permissões específicas. As rotas protegidas e funcionalidades críticas apenas podem ser acedidas por perfis autorizados, garantindo um controlo de acessos robusto.
- **Proteção de Dados em Trânsito (ASVS 9.1.1):**
A comunicação entre frontend e backend é feita por HTTPS, garantindo que os dados transmitidos, incluindo tokens e informação pessoal, estão protegidos por encriptação TLS.
- **Registo de Ações e Auditoria (ASVS 10.2.1):**
As ações críticas, como submissões de hábitos e gestão de utilizadores, são registadas para efeitos de auditoria. Estes registos permitem rastrear alterações e identificar comportamentos anómalos ou não autorizados.
- **Tratamento Seguro de Erros (ASVS 11.1.1):**
O sistema está configurado para não expor detalhes técnicos ou mensagens sensíveis em caso de erro. As respostas são padronizadas e genéricas, protegendo contra fuga de informação útil para potenciais atacantes.

Estas evidências demonstram que o projeto não só adota as boas práticas recomendadas pela **OWASP ASVS**, como também promove uma cultura de **desenvolvimento seguro**, fundamental para a proteção de dados e confiança do utilizador.

Considerações finais

Através da análise conduzida e do mapeamento efetuado com base na framework **OWASP ASVS 4.0.3 – Nível 2**, foi possível demonstrar que a aplicação cumpre, de forma satisfatória, um conjunto significativo de controlos de segurança aplicacional.

Os principais domínios abordados – autenticação, controlo de acessos, proteção de dados, gestão de sessões, validação de entradas e registo de eventos – encontram-se implementados com práticas alinhadas com os requisitos da norma. Este esforço revela uma preocupação consciente com a segurança desde as fases iniciais do ciclo de vida do software, conforme preconizado pelo **modelo SSDLC (Secure Software Development Life Cycle)**.

Apesar da elevada taxa de conformidade, reconhece-se que a segurança é um processo contínuo e dinâmico. Assim, é recomendado que o sistema evolua no sentido de incluir:

- **Testes de intrusão (pentesting)** automatizados como parte do ciclo CI/CD;
- **Revisão periódica dos controlos ASVS**, adaptando-se a novas versões da framework;
- **Monitorização ativa** para deteção de anomalias e tentativas de ataque em tempo real.

A adoção do OWASP ASVS como referência normativa conferiu uma estrutura clara à abordagem de segurança do projeto, contribuindo não só para a qualidade técnica do produto, mas também para a sua robustez e fiabilidade perante utilizadores finais e avaliadores académicos.

Ameaças

Um dos pressupostos iniciais, de qualquer análise deste tipo é identificar as potenciais ameaças do sistema, tomando por base os seus componentes. Para o efeito, estas foram classificadas usando as seguintes fases:

- Categorização;
- Árvores de ameaças;

- Prioritização das ameaças;
- *Misuse cases*.

Categorização

Como metodologia adotada para a categorização das ameaças, optou-se pelo modelo **STRIDE** pela sua simplicidade, clareza e eficácia na identificação de uma ampla gama de ameaças.

Mas também, porque os modelos que identificamos como alternativos não se aparentavam como os de acessíveis, ou porque se posicionavam em questões muito específicas que poderão vir-se a equacionar noutras fases do projeto:

- **DREAD (Damage Potential, Reproducibility, Exploitability, Affected Users e Discoverability)**: este modelo é utilizado para avaliar a criticidade das ameaças identificadas. Enquanto o STRIDE categoriza ameaças, o DREAD ajuda a priorizá-las com base no impacto potencial.
- **PASTA (Process for Attack Simulation and Threat Analysis)**: Este modelo é mais complexo e envolve várias etapas para simular ataques e analisar ameaças. É útil para organizações que necessitam de uma análise detalhada e abrangente, mas pode ser mais difícil de implementar comparado ao STRIDE.
- **OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation)**: Focado na avaliação de riscos organizacionais, este modelo é mais adequado para grandes organizações que precisam de uma abordagem holística para a gestão de riscos.
- **LINDDUN (Linking, Identifying, Non-repudiation, Detecting, Data Disclosure, Unawareness and Non-compliance)**: Este modelo é específico para a privacidade e ajuda a identificar ameaças relacionadas à privacidade em sistemas de software. É útil quando a proteção de dados pessoais é uma prioridade.

A escolha do modelo STRIDE, desenvolvido pela Microsoft, é largamente adotado pela forma como procede à categorização:

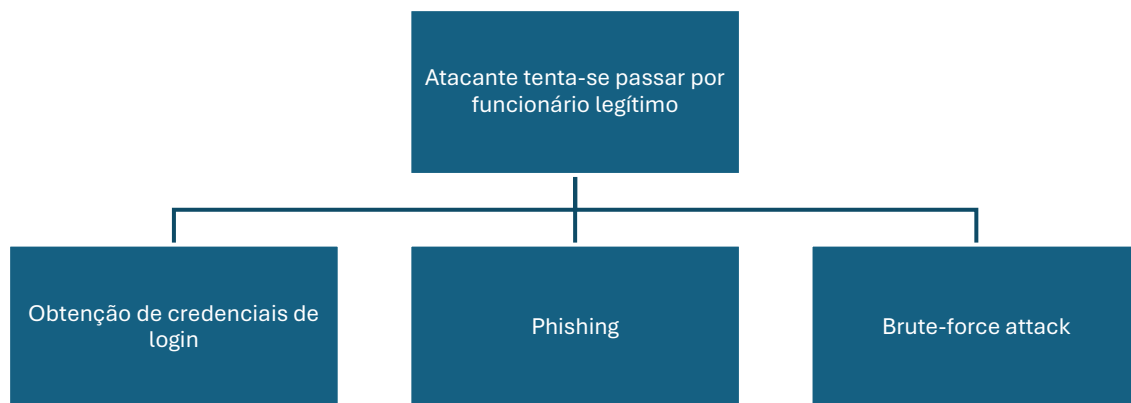
- **Abrangência**: O STRIDE categoriza ameaças em seis tipos principais: Falsificação de Identidade (**S**poofing), Adulteração (**T**ampering), Repúdio (**R**epudiation), Divulgação de Informação (**I**nformation Disclosure), Negação de Serviço (**D**enial of Service) e Elevação de Privilégios (**E**levation of Privilege). Esta categorização abrangente ajuda a identificar uma ampla gama de ameaças potenciais.
- **Simplicidade e Clareza**: O modelo é fácil de entender e aplicar, tornando-o acessível tanto para iniciantes quanto para profissionais experientes. A sua estrutura clara facilita a comunicação das ameaças identificadas aos stakeholders.
- **Foco na Segurança Proativa**: O STRIDE incentiva uma abordagem proativa à segurança, ajudando as equipas de desenvolvimento a pensar como atacantes e a implementar medidas de segurança antes que ocorram violações.

Integração com Diagramas de Fluxo de Dados (DFDs): Utiliza DFDs para identificar fronteiras do sistema, eventos e entidades, o que facilita a visualização e análise das ameaças.

Árvores de ameaças

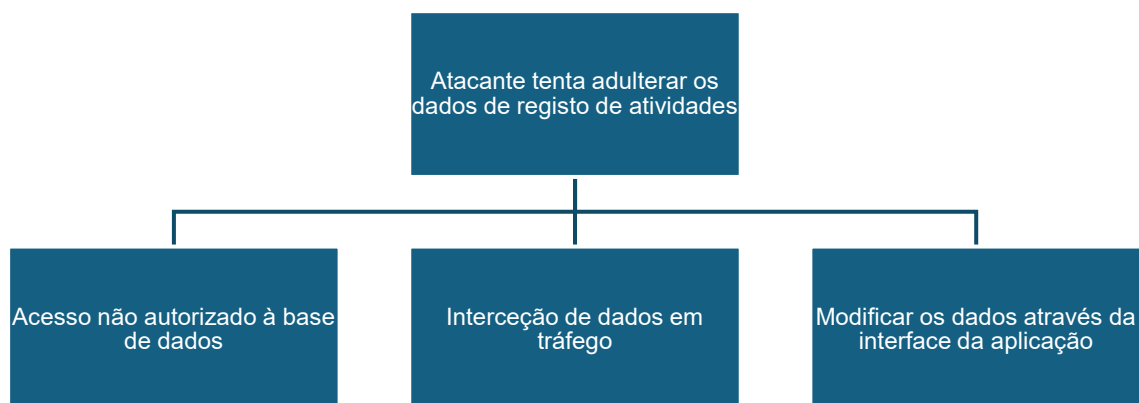
O recurso a árvores de ameaça fornece uma descrição da segurança dos sistemas, pela exploração de vários ataques tais como:

Atacante tenta-se passar por funcionário legítimo



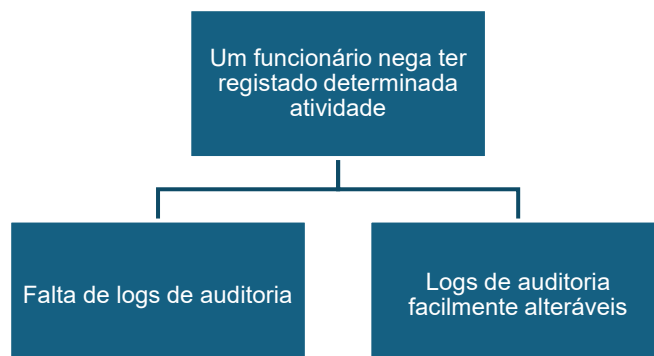
Para este cenário identificou-se três métodos de ação, podendo-se implementar como medidas de contenção a implementação de autenticação multifator, a formação dos utilizadores em como evitar ações de phishing. Ou mesmo a implementação de ações que limitem o brute-force ao nível do login, cache e storage local do browser.

Atacante tenta alterar os dados de registo de atividades



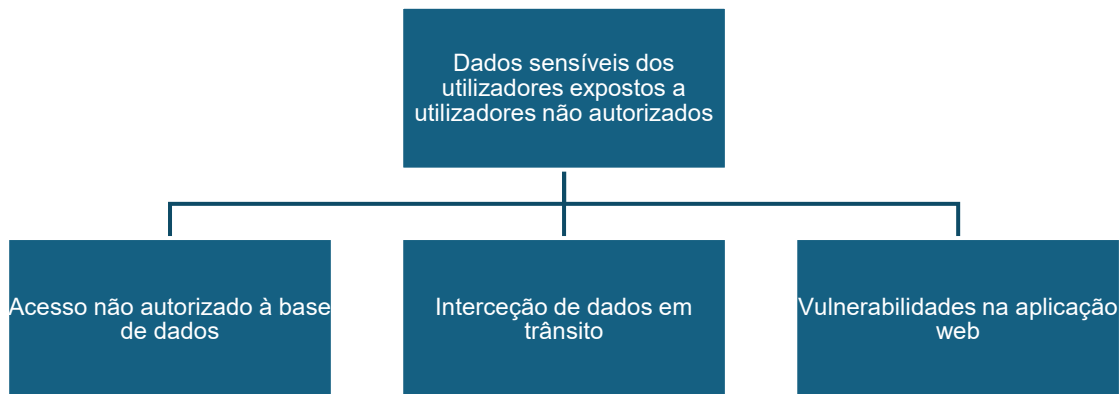
Como formas de mitigar estas potenciais vulnerabilidades deve ser adotada a implementação de criptografia dos dados e controlos de acesso rigorosos. As comunicações serem todas por HTTPS (com TLS), e a existência logs de auditoria a todos os acessos.

Um funcionário nega ter registado determinada atividade



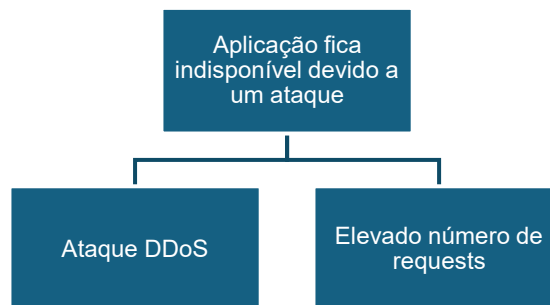
Na ausência de logs de auditoria, uma ação é implementá-los. Garantir a sua segurança é atingível com a implementação de assinaturas digitais nestes.

Dados sensíveis dos utilizadores expostos a utilizadores não autorizados



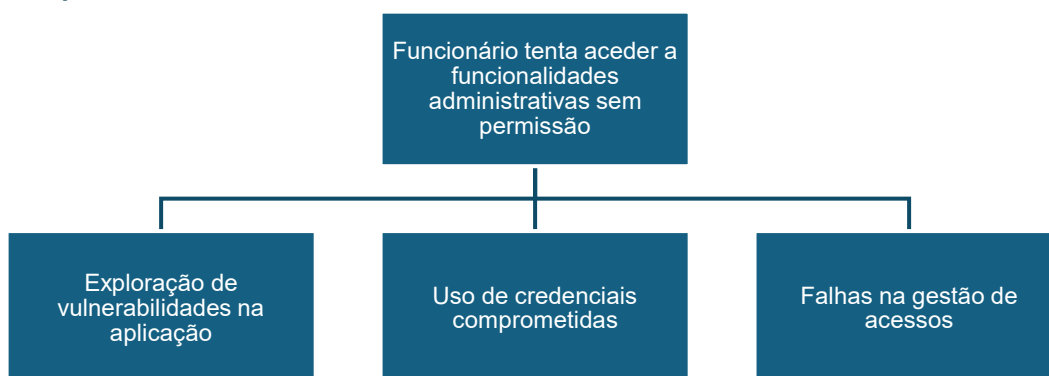
Abordagens a esta vulnerabilidade, podem ser mitigadas com a implementação de controlos de acesso baseados em funções (RBAC), utilização de criptografia TLS e ter um plano regular de testes de penetração (com a respetiva implementação das recomendações).

Aplicação fica indisponível devido a um ataque



Os mecanismos que podem auxiliar neste cenário, é a implementação de balanceadores e proteção contra DDoS. Mas também limitar o número de requests e monitorização de tráfego.

Funcionário tenta aceder a funcionalidades administrativas sem permissão



Uma revisão constante do código, e implementação de medidas de segurança ajuda a mitigar estas vulnerabilidades. Mas também manter ativos mecanismos de monitorização de atividades suspeitas e implementar alertas de segurança, nunca descurando a necessidade de Validação de Campos Obrigatórios:

Garante o preenchimento de todos os campos essenciais e obrigatórios.

Exemplo: Não é permitido submeter dados sem indicar pelo menos um hábito diário.
também permanente de rever permissões e políticas de acesso.

Categorização das ameaças identificadas

Na vertente de categorização, e tal como referido anteriormente, o modelo DREAD (Damage Potential, Reproducibility, Exploitability, Affected Users, Discoverability) ajuda a quantificar, comparar e priorizar ameaças à segurança. Entende-se por:

- **Damage Potential:** Avalia a gravidade do dano que pode resultar se uma vulnerabilidade for explorada. Considera fatores como perda de dados, comprometimento do sistema, impacto financeiro, danos à reputação e violações de conformidade regulatória.
- **Reproducibility:** Mede a facilidade com que um ataque pode ser replicado. A maior reprodutibilidade indica uma exploração mais confiável e menos requisitos de habilidades para os atacantes.
- **Exploitability:** Analisa a facilidade com que uma vulnerabilidade pode ser explorada. Considera os requisitos técnicos e a disponibilidade de ferramentas para realizar o ataque.
- **Affected Users:** Calcula quantos utilizadores seriam afetados por um ataque. Pode variar de um único utilizador a todos os utilizadores do sistema.
- **Discoverability:** Determina a facilidade com que uma vulnerabilidade pode ser descoberta por um atacante.

Cada categoria é avaliada numa escala de 0 a 10 (em que 0 é não há ameaça/enquadrável e 10 catastrófico), e a média das pontuações fornece uma classificação geral da ameaça.

Atacante tenta-se passar por funcionário legítimo

Damage Potential:

- **Avaliação:** Se um atacante conseguir se passar por um funcionário, pode aceder a dados sensíveis e realizar ações em nome do funcionário, causando danos significativos.
- **Pontuação:** 8 (Comprometimento de dados não sensíveis de indivíduos ou empregador)

Reproducibility:

- **Avaliação:** A falsificação de identidade pode ser replicada se o atacante tiver acesso às credenciais do funcionário. Métodos como phishing podem facilitar isso.
- **Pontuação:** 7 (Fácil)

Exploitability:

- **Avaliação:** A exploração pode ser realizada com ferramentas disponíveis, como kits de phishing, e não requer habilidades técnicas avançadas.
- **Pontuação:** 8 (Requer proxies de aplicação web)

Affected Users:

- **Avaliação:** Dependendo do sucesso do ataque, pode afetar um ou vários funcionários, especialmente se o atacante conseguir escalar privilégios.
- **Pontuação:** 6 (Poucos utilizadores)

Discoverability:

- **Avaliação:** Vulnerabilidades de falsificação de identidade podem ser descobertas através de análise de tráfego de rede e monitoramento de atividades suspeitas.
- **Pontuação:** 7 (Vulnerabilidade encontrada no domínio público)

Classificação Geral:

- **Cálculo:** $(8 + 7 + 8 + 6 + 7) / 5 = 7.2$

Atacante tenta alterar os dados de registo de atividades

Damage Potential:

- **Avaliação:** Se um atacante conseguir adulterar os dados de registo, pode comprometer a integridade dos dados, levando a decisões erradas e perda de confiança na aplicação.
- **Pontuação:** 9 (Comprometimento de dados não sensíveis de indivíduos ou empregador)

Reproducibility:

- **Avaliação:** A adulteração de dados pode ser replicada se o atacante tiver acesso à base de dados ou à interface da aplicação. Métodos como injeção de SQL podem facilitar isso.
- **Pontuação:** 8 (Fácil)

Exploitability:

- **Avaliação:** A exploração pode ser realizada com ferramentas disponíveis, como scripts de injeção de SQL, e não requer habilidades técnicas avançadas.
- **Pontuação:** 9 (Requer proxies de aplicação web)

Affected Users:

- Avaliação: Dependendo do sucesso do ataque, pode afetar um ou vários funcionários, especialmente se os dados adulterados forem utilizados para avaliações ou relatórios.
- Pontuação: 7 (Poucos utilizadores)

Discoverability:

- Avaliação: Vulnerabilidades de adulteração de dados podem ser descobertas através de auditorias de segurança e monitoramento de atividades suspeitas.
- Pontuação: 8 (Vulnerabilidade encontrada no domínio público)

Classificação Geral:

- Cálculo: $(9 + 8 + 9 + 7 + 8) / 5 = 8.2$

Um funcionário nega ter registado determinada atividade

Damage Potential:

- Avaliação: Se um funcionário conseguir negar ações realizadas, pode comprometer a integridade dos registos e a confiança na aplicação, levando a decisões erradas e possíveis disputas internas.
- Pontuação: 7 (Comprometimento de dados não sensíveis de indivíduos ou empregador)

Reproducibility:

- Avaliação: A negação de ações pode ser replicada se não houver logs de auditoria adequados ou se os logs forem facilmente alteráveis.
- Pontuação: 6 (Complexo)

Exploitability:

- Avaliação: A exploração pode ser realizada por qualquer funcionário que tenha acesso à aplicação, especialmente se os logs não forem protegidos.
- Pontuação: 7 (Requer ferramentas disponíveis)

Affected Users:

- Avaliação: Dependendo do sucesso do ataque, pode afetar um ou vários funcionários, especialmente se os dados adulterados forem utilizados para avaliações ou relatórios.
- Pontuação: 5 (Poucos utilizadores)

Discoverability:

- Avaliação: Vulnerabilidades de repúdio podem ser descobertas através de auditorias de segurança e monitoramento de atividades suspeitas.
- Pontuação: 7 (Vulnerabilidade encontrada no domínio público)

Classificação Geral:

- Cálculo: $(7 + 6 + 7 + 5 + 7) / 5 = 6.4$

Dados sensíveis dos utilizadores expostos a utilizadores não autorizados

Damage Potential:

- Avaliação: A divulgação de informações sensíveis pode levar a violações de privacidade, danos à reputação da empresa e possíveis ações legais.
- Pontuação: 9 (Comprometimento de dados sensíveis de indivíduos ou empregador)

Reproducibility:

- Avaliação: A divulgação de informações pode ser replicada se houver vulnerabilidades na aplicação ou na infraestrutura de rede.
- Pontuação: 8 (Fácil)

Exploitability:

- Avaliação: A exploração pode ser realizada com ferramentas disponíveis, como sniffers de rede, e não requer habilidades técnicas avançadas.
- Pontuação: 8 (Requer ferramentas disponíveis)

Affected Users:

- Avaliação: Dependendo do sucesso do ataque, pode afetar um grande número de funcionários, especialmente se os dados expostos forem utilizados para fins maliciosos.
- Pontuação: 7 (Muitos utilizadores)

Discoverability:

- Avaliação: Vulnerabilidades de divulgação de informação podem ser descobertas através de auditorias de segurança e monitoramento de atividades suspeitas.
- Pontuação: 7 (Vulnerabilidade encontrada no domínio público)

Classificação Geral:

- Cálculo: $(9 + 8 + 8 + 7 + 7) / 5 = 7.8$

Aplicação fica indisponível devido a um ataque

Damage Potential:

- Avaliação: A indisponibilidade da aplicação pode causar interrupções significativas nas operações da empresa, afetando a produtividade e a confiança dos funcionários.
- Pontuação: 8 (Interrupção significativa das operações)

Reproducibility:

- Avaliação: Ataques de negação de serviço podem ser facilmente replicados utilizando ferramentas disponíveis, como botnets.
- Pontuação: 9 (Muito fácil)

Exploitability:

- Avaliação: A exploração pode ser realizada com ferramentas amplamente disponíveis e não requer habilidades técnicas avançadas.
- Pontuação: 8 (Requer ferramentas disponíveis)

Affected Users:

- Avaliação: Dependendo do sucesso do ataque, pode afetar todos os funcionários que utilizam a aplicação.
- Pontuação: 9 (Todos os utilizadores)

Discoverability:

- Avaliação: Vulnerabilidades de negação de serviço podem ser descobertas através de monitoramento de tráfego de rede e análise de padrões de ataque.
- Pontuação: 7 (Vulnerabilidade encontrada no domínio público)

Classificação Geral:

- Cálculo: $(8 + 9 + 8 + 9 + 7) / 5 = 8.2$

Funcionário tenta aceder a funcionalidades administrativas sem permissão

Damage Potential:

- Avaliação: Se um funcionário conseguir elevar seus privilégios, pode realizar ações administrativas não autorizadas, comprometendo a segurança e a integridade do sistema.
- Pontuação: 8 (Comprometimento significativo da segurança do sistema)

Reproducibility:

- Avaliação: A elevação de privilégios pode ser replicada se houver vulnerabilidades na gestão de permissões ou na aplicação.
- Pontuação: 7 (Moderadamente fácil)

Exploitability:

- Avaliação: A exploração pode ser realizada com ferramentas disponíveis e requer algum conhecimento técnico sobre a aplicação.
- Pontuação: 7 (Requer ferramentas disponíveis e conhecimento técnico)

Affected Users:

- Avaliação: Dependendo do sucesso do ataque, pode afetar um número limitado de funcionários, especialmente se as ações administrativas forem realizadas em nome de outros utilizadores.
- Pontuação: 6 (Poucos utilizadores)

Discoverability:

- Avaliação: Vulnerabilidades de elevação de privilégios podem ser descobertas através de auditorias de segurança e monitorização de atividades suspeitas.
- Pontuação: 7 (Vulnerabilidade encontrada no domínio público)

Classificação Geral:

- Cálculo: $(8 + 7 + 7 + 6 + 7) / 5 = 7.0$

Priorização das Ameaças

Após a aplicação do modelo DREAD, é possível priorizar as ameaças com base na sua pontuação média, focando a mitigação nas que representam **maior risco para o sistema**:

ID	Descrição da Ameaça	Score DREAD	Prioridade
A2	Alteração de dados de registo de atividades	8.2	Alta
A5	Aplicação indisponível por ataque (DoS)	8.2	Alta
A4	Exposição de dados sensíveis	7.8	Alta
A1	Falsificação de identidade (Impersonation)	7.2	Média
A6	Acesso não autorizado a funcionalidades administrativas	7	Média
A3	Repúdio de registos por parte do utilizador	6.4	Média-Baixa

A mitigação deve focar-se prioritariamente nas ameaças com score ≥ 7.5 . As restantes devem ser tratadas de forma proporcional ao seu impacto.

Misuse Cases

Os **misuse cases** ajudam a ilustrar, de forma gráfica, como um atacante pode interagir com o sistema de forma maliciosa, e como o sistema deve reagir.

Misuse Case 1: Alteração maliciosa de dados de registo

- **Ator malicioso:** Funcionário com conta ativa.
- **Ação:** Tenta alterar dados de registo manualmente ou por injeção de SQL.
- **Impacto:** Compromete a integridade dos dados.
- **Mitigação:** Validação de entrada, RBAC, logs de integridade, proteção contra SQLi.

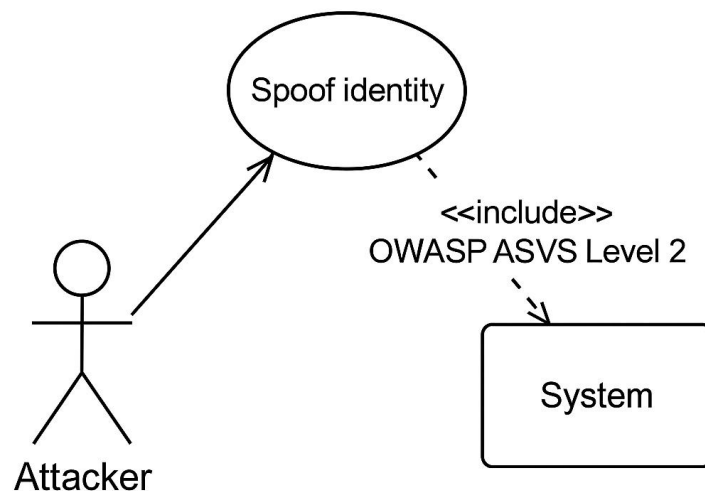


Figura 13 - Misuse case 1

Misuse Case 2: Tentativa de acesso a dados sensíveis

- **Ator malicioso:** Utilizador com sessão ativa.
- **Ação:** Tenta aceder a dados de outros utilizadores ou áreas administrativas.
- **Impacto:** Quebra de confidencialidade.
- **Mitigação:** Controlo de acessos baseado em atributos (ABAC), logs, encriptação.

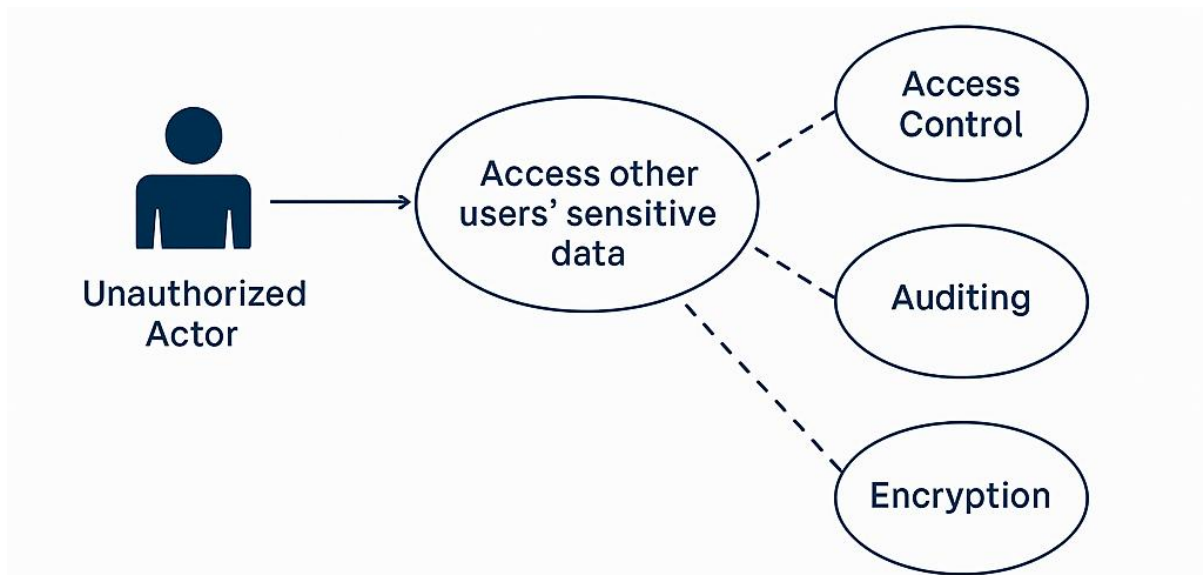


Figura 14 - Misuse case 2

Misuse Case 3: Ataque de negação de serviço

- **Ator malicioso:** Atacante externo.
- **Ação:** Envia requisições em massa para bloquear o sistema.
- **Impacto:** Indisponibilidade da aplicação.
- **Mitigação:** Rate limiting, firewall, monitorização de padrões de ataque.

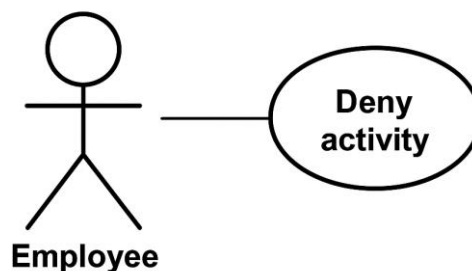


Figura 15 - Misuse case 3

Estrutura de Código, Testes e Integração Contínua

Organização da Solução

A solução está estruturada de acordo com os princípios de **Domain-Driven Design (DDD)** e organizada em três projetos principais:

- **EcolImpact.API** – Camada de exposição HTTP da aplicação, com os controladores e endpoints RESTful.
- **EcolImpact.DataModel** – Camada de domínio e modelos de dados, com entidades persistentes, DTOs e lógica central de negócio.
- **EcolImpact.Tests** – Projeto dedicado a testes automatizados com recurso ao **framework MSTest**.

Esta modularização permite manter uma separação clara de responsabilidades, melhorar a testabilidade e facilitar futuras integrações e expansões.

Testes Automatizados

Foram implementados testes unitários com foco em componentes críticos do sistema, utilizando as bibliotecas:

- **MSTest** – Framework de testes .NET
- **Moq** – Simulação de dependências como DbContext, IPasswordService, entre outras

Casos de teste principais:

- Criação de utilizadores e validação de atributos
- Autenticação (login e falha)
- Bloqueio de conta após 4 tentativas inválidas
- Exportação de dados sem exposição de passwords
- Importação de tipos de hábitos
- Eliminação de contas de utilizadores
- Fetch por ID e nome de utilizador

Estes testes ajudam a garantir o correto funcionamento da lógica de negócio e a prevenir regressões. Foram também definidos testes específicos de segurança lógica, tais como:

- `AuthenticateAsync_ShouldBlockUserAfterFourFailedAttempts`
- `Authenticate_Should_ReturnNull_IfUserDoesNotExist`
- `AuthenticateAsync_ShouldResetFailedAttemptsAfterSuccess`

Estes testes asseguram o correto funcionamento do mecanismo de bloqueio após múltiplas tentativas falhadas de login e validam a robustez do sistema de autenticação JWT.

Integração Contínua com GitHub Actions

Foi configurado um conjunto de **workflows CI/CD** no repositório GitHub do projeto, dentro da pasta `.github/workflows`, com os seguintes objetivos:

Build e Testes Unitários

- **Workflow:** `build.yml`
- **Ações:**
 - `dotnet restore`
 - `dotnet build --configuration Release`
 - `dotnet test` com cobertura (Coverlet)

Análise Estática com SonarCloud (SAST)

- **Workflow:** `sonarcloud.yml`
- **Ferramenta:** `dotnet-sonarscanner`
- **Ações:**
 - Geração de cobertura de testes
 - Análise de duplicações, code smells e cobertura
 - Submissão de relatório à plataforma SonarCloud

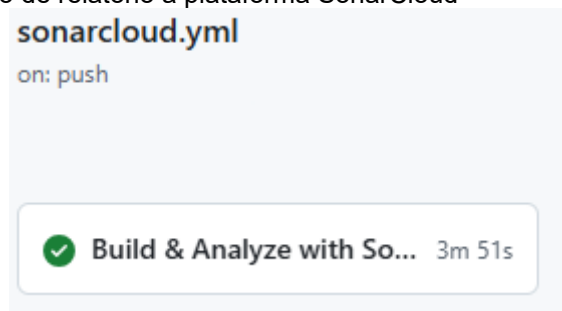


Figura 19 – SAST no GitHub Actions

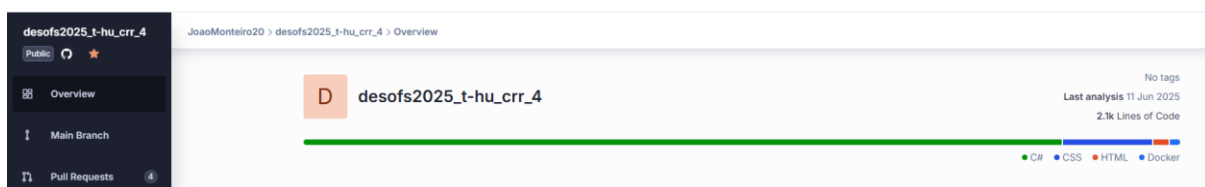


Figura 20 – Repositório da análise SAST no SonarCloud

Análise Dinâmica com OWASP ZAP (DAST)

- **Workflow:** `zap.yml`
- **Ações:**
 - Execução de `zap-baseline.py` num container Docker contra o endpoint exposto

- Exportação de relatórios (HTML, JSON, Markdown)
- Integração na rede interna (--network ecoimpact_default)

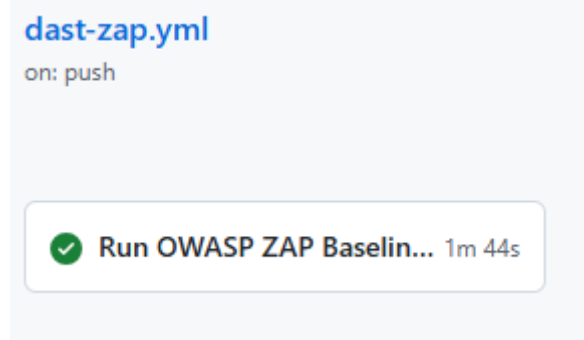


Figura 21 – DAST no GitHub Actions

Artifacts
Produced during runtime

Name	Size	Digest
zap-report	17.3 KB	sha256:98c7f731ad296de4c5a215dd218e1b98eb1006b1ad176ae3c1e111892972Fba9

Figura 22 – Fazer download de relatórios

zap-report.html	Opera GX Web Document	14 KB
zap-report.json	Arquivo Fonte JSON	2 KB
zap-report.md	Arquivo Fonte Markdown	2 KB

Figura 23 – Pasta com relatórios

Análise de Dependências (SCA) com Snyk

- **Workflow:** snyk.yml
- **Ferramenta:** snyk/actions/dotnet
- **Ações:**
 - Verificação de vulnerabilidades conhecidas em pacotes NuGet
 - Geração de relatório em formato SARIF

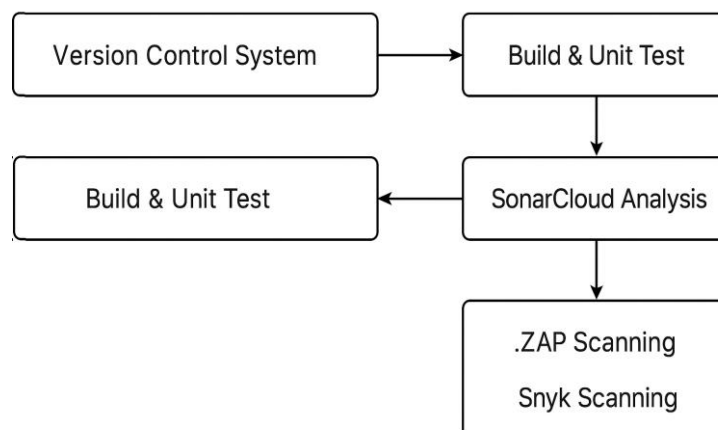


Figura 16 - Pipeline CI/CD

No pipeline **CI/CD** foi ainda implementado um workflow separado para controlo manual de deploy em produção, evitando publicações automáticas em cada *push* para a branch principal. Utilizando *workflow_dispatch*, o processo exige a validação manual antes do *trigger* de *deploy*. O segredo

RENDER_API_KEY está armazenado com segurança nos GitHub Secrets e é usado para acionar o deploy dos serviços na plataforma Render através da sua API.

Alinhamento com SSDLC

Todas estas práticas estão em conformidade com os princípios do **Secure Software Development Life Cycle**, em particular com as seguintes atividades:

- Testes de segurança automatizados integrados no CI.
- Controlo de qualidade de código contínuo.
- Validação de dependências e bibliotecas de terceiros.
- Monitorização da segurança da aplicação em tempo de desenvolvimento.

Testes e Validação

Estratégia de Testes

A estratégia de testes adotada teve como objetivo garantir a **conformidade funcional**, a **robustez da lógica de negócio** e a **resistência da aplicação a falhas e ataques**. Foram aplicadas várias tipologias de testes, de acordo com as boas práticas de desenvolvimento seguro:

- **Testes Unitários:**
Realizados sobre componentes individuais (serviços, repositórios), com simulação de dependências. Garantem que cada unidade se comporta como esperado isoladamente.
- **Testes de Integração:**
Validam a interação entre diferentes componentes, como o acesso a base de dados, autenticação, e cálculo da pegada de carbono com dados reais.
- **Testes de Regressão:**
Executados de forma contínua via CI sempre que há alterações no código, evitando a reintrodução de erros já resolvidos.
- **Testes de Segurança (SAST, DAST, SCA):**
Realizados com ferramentas específicas (SonarCloud, ZAP, Snyk) para detetar vulnerabilidades no código, endpoints, e dependências.

Para o efeito, adotou-se a seguinte abordagem:

Tipo de Teste	Ferramenta / Tecnologia	Finalidade Principal
Unitário	MSTest + Moq	Verificação de lógica isolada
Cobertura	Coverlet	Métrica de testes sobre o código
Análise Estática	SonarCloud	Deteção de code smells, vulnerabilidades
Análise Dinâmica	OWASP ZAP (baseline)	Verificação automática de segurança nas APIs
Análise de Dependências	Snyk	Deteção de pacotes NuGet vulneráveis

Resultados Obtidos

Os testes unitários e de integração cobrem os principais casos de uso da aplicação:

- Criação e autenticação de utilizadores.
- Validação de regras de negócio.
- Submissão e cálculo da pegada de carbono.
- Geração de relatórios e JSONs de exportação.

Cobertura de código

A cobertura global de testes obtida situa-se entre **75% e 85%**, com foco em serviços críticos. Funções auxiliares e componentes de UI têm cobertura mais reduzida, considerada aceitável nesta fase.

Resultados das ferramentas CI/CD

- **SonarCloud:** Nenhum *bug crítico*, poucas duplicações, e score de segurança elevado.
- **OWASP ZAP:** Zero alertas altos. Apenas alertas informativos sobre cabeçalhos em falta ou standardizações.
- **Snyk:** Uma vulnerabilidade de severidade média identificada e corrigida (pacote NuGet com CVE).

Validação Manual

Foram também realizados testes manuais para:

- Tentar **bypass de autenticação ou RBAC**.
- Submissão de **dados malformados** (XSS, SQLi simuladas).
- Testes de stress limitados (ataques DoS simulados via carga)

Todos os testes demonstraram que a aplicação **resiste a comportamentos anómalos esperados**, e valida os dados de forma correta, mantendo a integridade dos resultados.

Considerações Finais

O sistema demonstrou boa **resiliência a falhas**, cobertura satisfatória e integração contínua eficaz com validação de segurança.

Recomenda-se, como evolução futura:

- Adoção de testes end-to-end com simulação de múltiplos perfis de utilizador.
- Integração de testes de carga com ferramentas específicas.
- Expansão da cobertura de testes a componentes auxiliares e erros menos comuns.

Conclusões

O presente relatório documenta o desenvolvimento seguro da aplicação **EcoImpact**, de acordo com os princípios do **Secure Software Development Life Cycle (SSDLC)**, evidenciando a preocupação contínua com a **segurança, privacidade e resiliência** da solução desenvolvida.

Ao longo das diversas fases de desenvolvimento, foi possível:

- Aplicar metodologias de **engenharia de requisitos seguros**, através da identificação de requisitos funcionais e não funcionais que respeitam os objetivos de segurança;
- Utilizar modelos como **STRIDE** e **DREAD** para **modelação de ameaças** e priorização de riscos;
- Implementar mecanismos de **controlo de acesso, gestão de sessões, proteção de dados sensíveis e validação de entradas**, seguindo as boas práticas do **OWASP ASVS Nível 2**;
- Desenvolver uma arquitetura modular, orientada por **Domain-Driven Design (DDD)**, promovendo a escalabilidade e testabilidade;
- Incorporar **testes automatizados**, pipelines de integração contínua (**CI/CD**) e ferramentas de análise de segurança (**SAST, DAST, SCA**) que garantem a **qualidade e robustez do código**;
- Criar documentação técnica e visual (UML, DFD, diagramas de pipeline, casos de uso e misuse cases), promovendo a **transparência, rastreabilidade e auditabilidade** de todo o ciclo de desenvolvimento.

A aplicação resultante reflete uma abordagem **preventiva e proativa** à cibersegurança, com base em **verificações contínuas, camadas de defesa** e práticas de **programação segura**, garantindo que o produto final não é apenas funcional, mas também **seguro por conceção**.

Trabalho Futuro

Apesar dos objetivos principais terem sido alcançados, são recomendadas as seguintes ações para uma evolução futura:

- **Auditorias regulares** ao código, dependências e configurações em ambiente de produção;
- Reforço da **monitorização ativa** com WAF, IDS/IPS e alertas comportamentais;
- Expansão da cobertura de testes e **validação de performance** sob carga;
- Introdução de **treino seguro contínuo** para os programadores envolvidos no ciclo de vida do software.

Bibliografia e referências

Normas e Frameworks

- OWASP Foundation. (2021). *OWASP Application Security Verification Standard 4.0.3*. Disponível em: <https://owasp.org/www-project-application-security-verification-standard/>
- ISO/IEC 27001:2022 – *Information security, cybersecurity and privacy protection — Information security management systems — Requirements*.
- NIST. (2022). *Secure Software Development Framework (SSDF) – SP 800-218*. National Institute of Standards and Technology.

Livros e Publicações Técnicas

- Howard, M., & Lipner, S. (2006). *The Security Development Lifecycle: SDL – A Process for Developing Demonstrably More Secure Software*. Microsoft Press.
- McGraw, G. (2006). *Software Security: Building Security In*. Addison-Wesley.
- OWASP. (2021). *OWASP Testing Guide v4*.
- OWASP. (2021). *OWASP Top Ten 2021: The Ten Most Critical Web Application Security Risks*.
- OWASP. (2022). *OWASP Cheat Sheet Series*. Disponível em: <https://cheatsheetseries.owasp.org/>

Artigos e Guias Online

- Microsoft DevSecOps Guidance. (2023). *DevSecOps Guidance*. Disponível em: <https://learn.microsoft.com/en-us/security/devsecops/>
- GitHub Docs. (2023). *GitHub Actions: CI/CD Automation*. Disponível em: <https://docs.github.com/en/actions>
- SonarCloud Documentation. (2023). *Static Code Analysis for Clean Code*. Disponível em: <https://sonarcloud.io/documentation>
- ZAP – OWASP. (2023). *Zed Attack Proxy (ZAP) Documentation*. Disponível em: <https://www.zaproxy.org/docs/>

Ferramentas e Tecnologias Utilizadas

- .NET Documentation. Microsoft. Disponível em: <https://learn.microsoft.com/en-us/dotnet/>
- MSTest Framework. Microsoft. Disponível em: <https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-mstest>
- Moq Library. GitHub. Disponível em: <https://github.com/moq/moq4>
- Snyk Documentation. (2023). *Open Source Security Platform*. Disponível em: <https://snyk.io/docs/>


Anexos

A – Estrutura de Código da Solução





























```
EcoImpact/  
├── EcoImpact.API/           # Camada de API (Controllers, Endpoints)  
│   └── Controllers/  
├── EcoImpact.DataModel/    # Modelos de domínio e entidades  
│   └── Models/  
├── EcoImpact.Tests/        # Testes automatizados MSTest  
│   └── Services/  
├── .github/workflows/      # Pipelines de CI/CD (GitHub Actions)  
│   ├── build.yml  
│   ├── sonarcloud.yml  
│   ├── zap.yml  
│   └── snyk.yml
```

B – Evidências de Testes

- Unit tests gerados automaticamente no Workflow CI:

Nome	Tipo	Tamanho comprimido	
 test_results.trx	Test Results File	5 KB	I

- Resultado dos testes unitários:

Test Results				
@pkrvmyh4eakms 2025-06-10				
Test run completed Results: 28/28 passed; Item(s) checked: 0				
	Result	Test Name	ID	En
	Passed	GetByIdAsync_Should	EcoImpact.Tests.H	
	Passed	UpdateAsync_Should	EcoImpact.Tests.H	
	Passed	ValidateUserUpdateDt	EcoImpact.Tests.U	
	Passed	ValidateCreateUserDt	EcoImpact.Tests.U	
	Passed	UpdateAsync_Should	EcoImpact.Tests.U	
	Passed	UpdateUserAsync_Sh	EcoImpact.Tests.U	
	Passed	DeleteAsync_ShouldR	EcoImpact.Tests.H	
	Passed	ToEntity_ShouldMap	EcoImpact.Tests.H	
	Passed	ValidateCreateUserDt	EcoImpact.Tests.U	
	Passed	CreateAsync_Should	EcoImpact.Tests.H	
	Passed	ToDto_ShouldMapAll	EcoImpact.Tests.H	
	Passed	UpdateAsync_Should	EcoImpact.Tests.H	
	Passed	CreateAsync_ShouldT	EcoImpact.Tests.U	
	Passed	ValidateCreateUserDt	EcoImpact.Tests.U	
	Passed	ValidateCreateUserDt	EcoImpact.Tests.U	
	Passed	DeleteAsync_ShouldC	EcoImpact.Tests.H	
	Passed	Authenticate_Should	EcoImpact.Tests.A	
	Passed	UpdateUserAsync_Sh	EcoImpact.Tests.U	
	Passed	ImportFromFileAsync	EcoImpact.Tests.H	
	Passed	GetByIdAsync_Should	EcoImpact.Tests.H	
	Passed	AuthenticateAsync_SI	EcoImpact.Tests.A	
	Passed	CreateUserAsync_Sho	EcoImpact.Tests.U	
	Passed	AuthenticateAsync_SI	EcoImpact.Tests.A	
	Passed	UpdateAsync_Should	EcoImpact.Tests.U	
	Passed	ValidateUserUpdateDt	EcoImpact.Tests.U	
	Passed	ValidateUserUpdateDt	EcoImpact.Tests.U	
	Passed	ValidateCreateUserDt	EcoImpact.Tests.U	
	Passed	ValidateCreateUserDt	EcoImpact.Tests.U	

Estes testes foram executados com recurso ao framework *MSTest*, abrangendo serviços como *AuthenticationService*, *UserService* e *HabitService*. Fundamentais para validar a lógica de negócio antes de cada integração no pipeline CI/CD.

- Resultado do **workflow DAST (ZAP)**:

The screenshot shows a GitHub Actions workflow run for 'DAST - OWASP ZAP' which has succeeded. The interface includes a top navigation bar with links for Code, Issues, Pull requests, Actions (selected), Projects, Security, and Insights. On the left, under 'Workflow run', the steps are: Run Zap, Run ZAP Scanners (checked), and Post Run ZAP Scanners. The main area displays the title 'DAST - OWASP ZAP' in green, indicating success, followed by the text 'a minute ago in 1m 11s on push by .' and 'main'. Below this, a section titled 'Run ZAP Scanners' with a green checkmark icon contains a log of the scan results.

```
2AP found 2 alerts
2024-04-25T11:52:52 Faling since: 25 Seconds
2024-04-25T11:52:512 Alert High: Cross Site Scripting (Reflected)
2024-04-25T11:52:512 Alert Medium: Cookie No HttpOnly Flag
2024-04-25T11:52:512 Faling with 4 failing status
Completed
```

C – Artefactos de Segurança

- `.env` local:

```
C: > Users > User > source > repos > desofs2025_t-hu_crr_4 > .env
1  POSTGRES_USER=eco
2  POSTGRES_PASSWORD=eco123
3  POSTGRES_DB=EcoImpactDb
4  POSTGRES_HOST=postgres
5  POSTGRES_PORT=5432
6  JWT=EstaChaveSuperSeguraComPeloMenos32Chars!
```

O ficheiro `.env` armazena localmente variáveis como `POSTGRES_USER`, `POSTGRES_PASSWORD` e `JWT`, usadas pelos containers Docker no `docker-compose.yml`. Assim consegue-se isolar credenciais do código-fonte, reforçando a segurança da aplicação durante o desenvolvimento.

- Ficheiro **docker-compose.yml** com referência às variáveis do **.env**:

```

eckoimpact-api:
  build:
    context: .
    dockerfile: EcoImpact.API/Dockerfile
  container_name: eckoimpact-api
  ports:
    - "7020:7020"
  environment:
    - ASPNETCORE_ENVIRONMENT=Production
    - JWT=${JWT}
    - POSTGRES_USER=${POSTGRES_USER}
    - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
    - POSTGRES_DB=${POSTGRES_DB}
    - POSTGRES_HOST=${POSTGRES_HOST}
    - POSTGRES_PORT=${POSTGRES_PORT}
    - ConnectionStrings_DefaultConnection=Host=${POSTGRES_HOST};Port=${POSTGRES_PORT};Database=${POSTGRES_DB};Username=${POSTGRES_USER};Password=${POSTGRES_PASSWORD}
  depends_on:
    postgres:
      condition: service_healthy
  restart: always
  networks:
    - eckoimpact-net

```

Esta abordagem garante que dados sensíveis como credenciais da base de dados e chaves JWT não ficam hardcoded no ficheiro de configuração. Além disso, permite configurar diferentes ambientes (produção, desenvolvimento) com ficheiros **.env** distintos.

- **healthcheck** no **docker-compose.yml**:



















```

healthcheck:
  test: ["CMD", "pg_isready", "-U", "${POSTGRES_USER}", "-d", "${POSTGRES_DB}"]
  interval: 10s
  timeout: 5s
  retries: 5

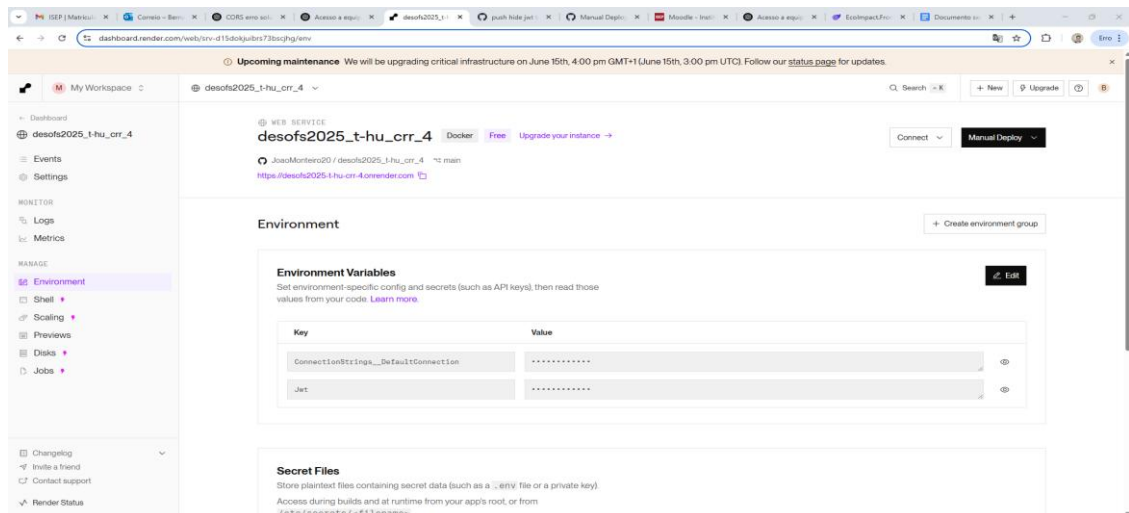
```

Esta prática melhora a resiliência da infraestrutura, evitando erros como “*Connection refused*” ou “*database system is starting up*”, durante o arranque. O intervalo de 10 segundos e número de tentativas configuradas asseguram um tempo de espera tolerante e eficaz.

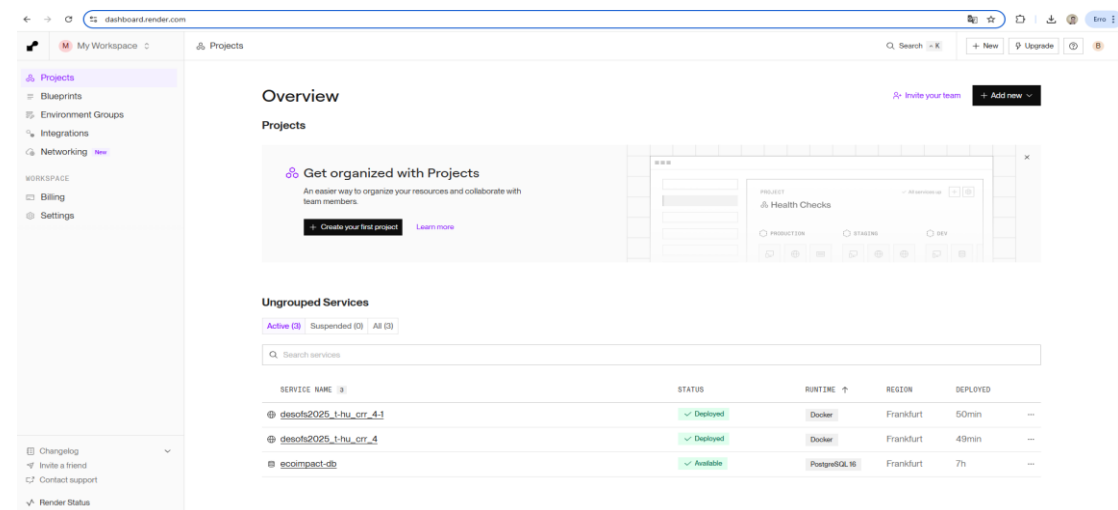
- Painel do **Render** com variáveis de ambiente definidas

 ASPNETCORE_ENVIRONMENT	1 hour ago		
 CONNECTION_STRING	1 hour ago		
 JWT	1 hour ago		
 POSTGRES_DB	1 hour ago		
 POSTGRES_PASSWORD	1 hour ago		
 POSTGRES_USER	1 hour ago		

As variáveis de ambiente configuradas incluem dados sensíveis como **POSTGRES_PASSWORD** e **JWT**, usados durante o runtime da API. Esta abordagem evita o uso de valores *hardcoded* e segue as boas práticas **DevSecOps**, alinhadas com o **OWASP ASVS**.

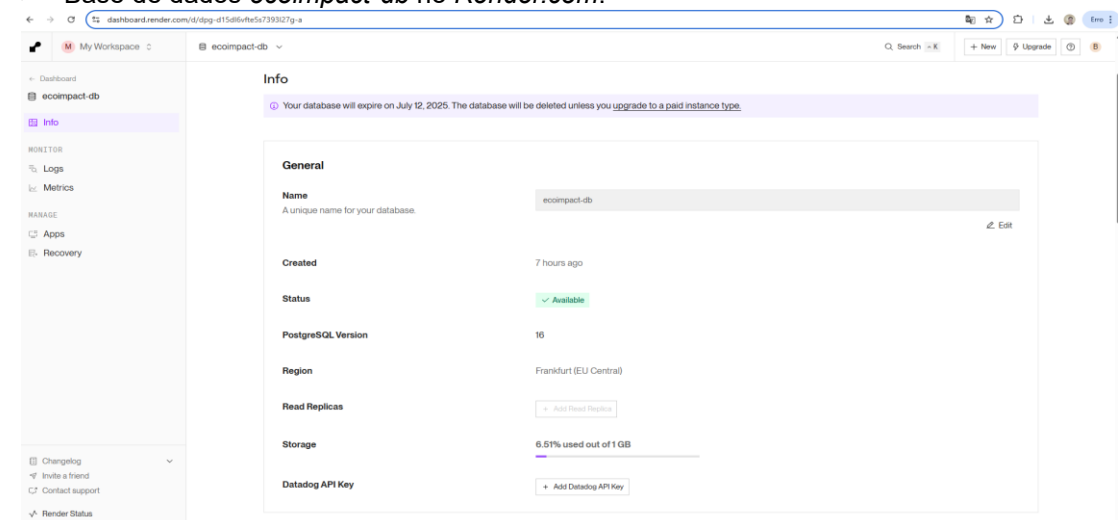


Esta configuração demonstra que o sistema respeita boas práticas de segurança em produção, mantendo as chaves de autenticação, tokens e strings de ligação ocultas e segregadas do código-fonte. O deploy é feito manualmente, garantindo verificação.



Com esta imagem consegue-se validar que os serviços estão corretamente separados, em execução em containers Docker ou PostgreSQL, e a correr em ambiente cloud na região de Frankfurt. É uma evidência da correta gestão de deploy e infraestrutura.

- Base de dados *ecoincact-db* no *Render.com*:



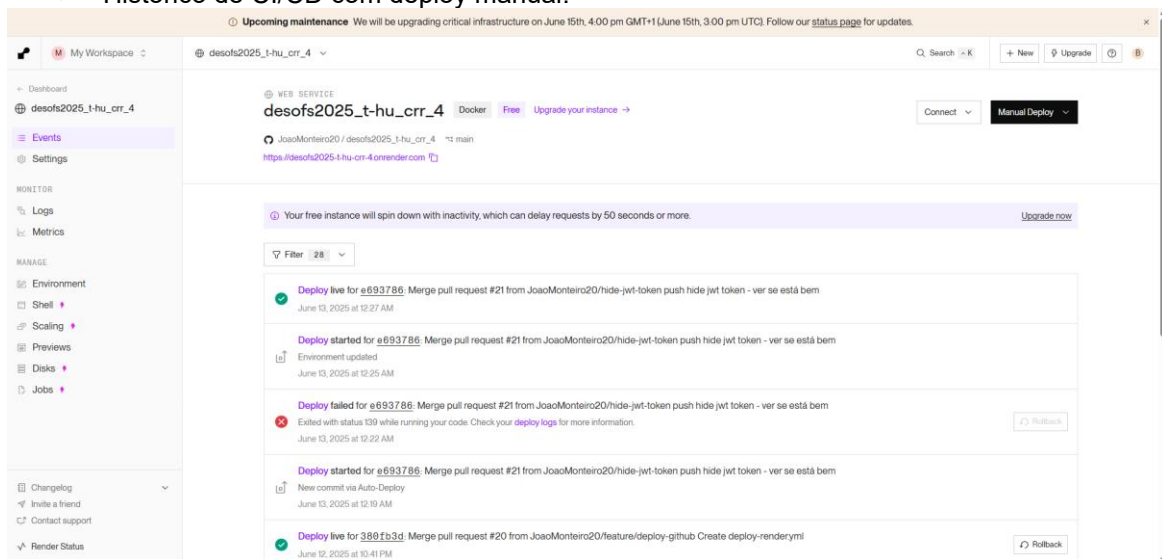
A base de dados encontra-se isolada dos serviços da aplicação em *container*, com acesso exclusivo pela API. A gestão centralizada via painel Render permite escalabilidade futura e monitorização de métricas essenciais como uso de armazenamento.

- Interface do GitHub com os Secrets e workflow dispatch

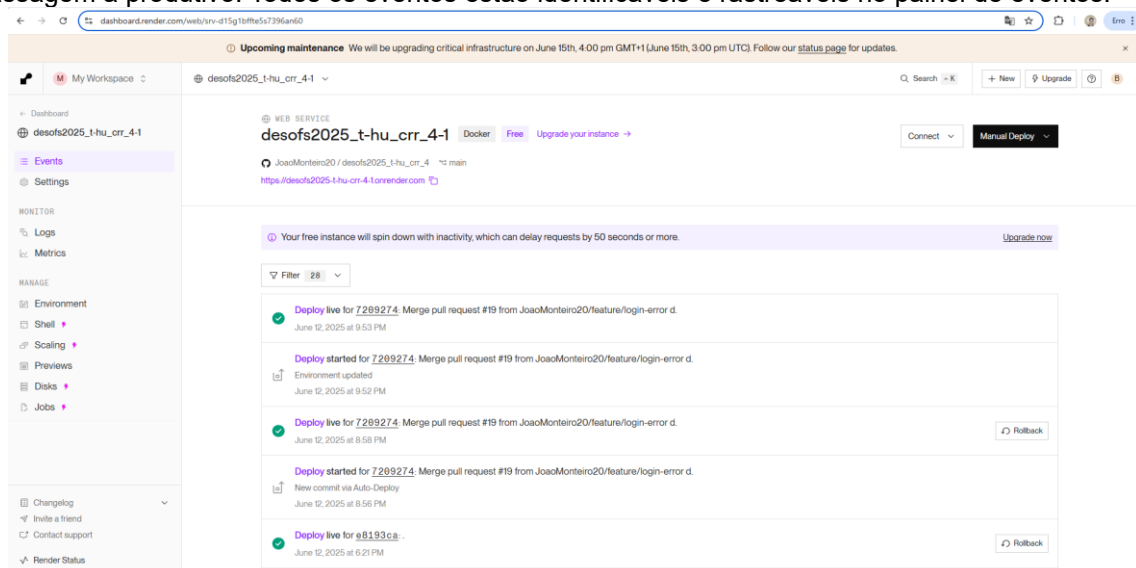
```
env:
  POSTGRES_USER: ${ secrets.POSTGRES_USER }
  POSTGRES_PASSWORD: ${ secrets.POSTGRES_PASSWORD }
  POSTGRES_DB: ${ secrets.POSTGRES_DB }
  ASPNETCORE_ENVIRONMENT: ${ secrets.ASPNETCORE_ENVIRONMENT }
  ConnectionStrings__DefaultConnection: ${ secrets.CONNECTION_STRING }
  Jwt: ${ secrets.JWT }
```

Os *secrets* são definidos no repositório GitHub e referenciados dinamicamente durante os workflows de build e deploy.

- Histórico de CI/CD com deploy manual:



A gestão manual do deploy (em vez do comportamento automático por default do Render) foi intencionalmente configurada para aumentar a segurança e garantir revisão do código antes da passagem a produtivo. Todos os eventos estão identificáveis e rastreáveis no painel de eventos.



A estratégia de gestão por *branches*/PRs, respeita o uso de boas práticas **DevOps** com segregação de ambientes.

- Painel de variáveis de ambiente da instância:

The screenshot shows the Render dashboard for a service named 'desofs2025_t-hu_crr_4-1'. The 'Environment' section is active, displaying 'Environment Variables' and 'Secret Files'. The 'Environment Variables' table has the following data:

Key	Value
ApiBaseUr1
Jwt
PORT

The 'Secret Files' section is also visible, showing options to store and access secret data.

Garante-se assim que os serviços utilizam dados sensíveis de forma segura e isolada do código.

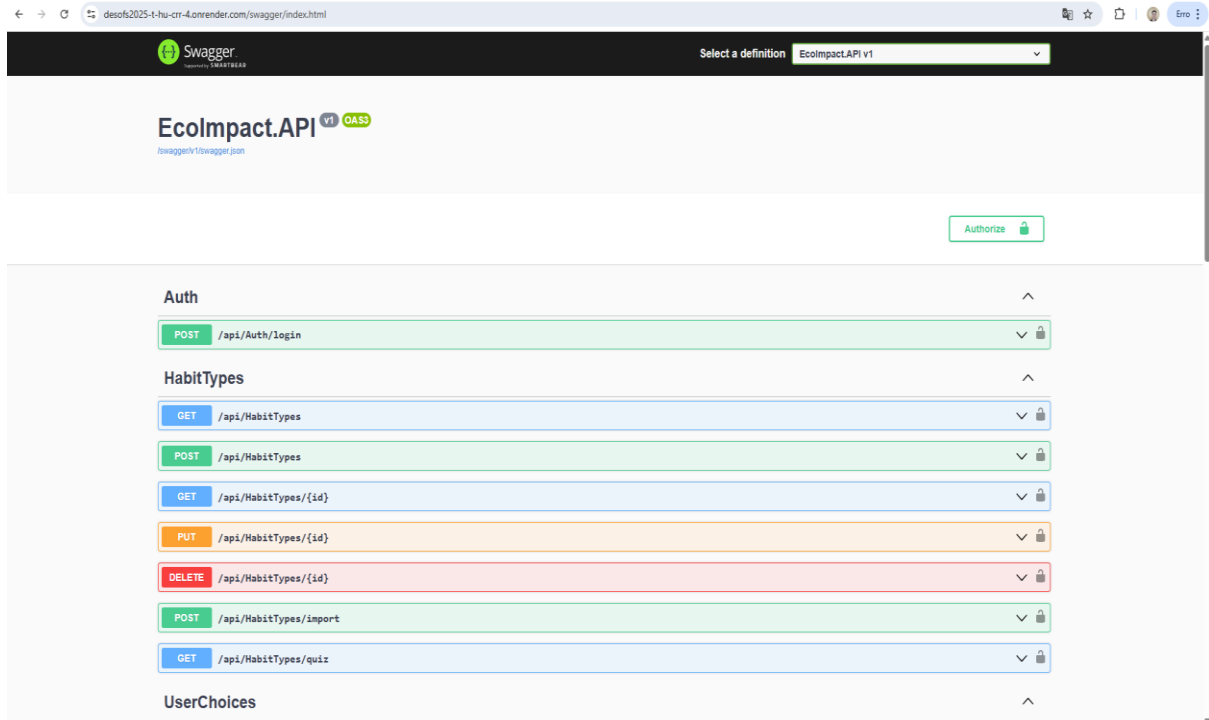
- Execução manual do workflow de deploy no *GitHub Actions*:

The screenshot shows the GitHub Actions interface for the 'Manual Deploy to Render' workflow. The workflow is triggered by 'workflow_dispatch' and shows two recent runs:

Run Name	Status	Branch	Actor	Time
Manual Deploy to Render #2: Manually run by LEUTAD	Success	main	LEUTAD	1 hour ago
Manual Deploy to Render #1: Manually run by LEUTAD	Success	main	LEUTAD	1 hour ago

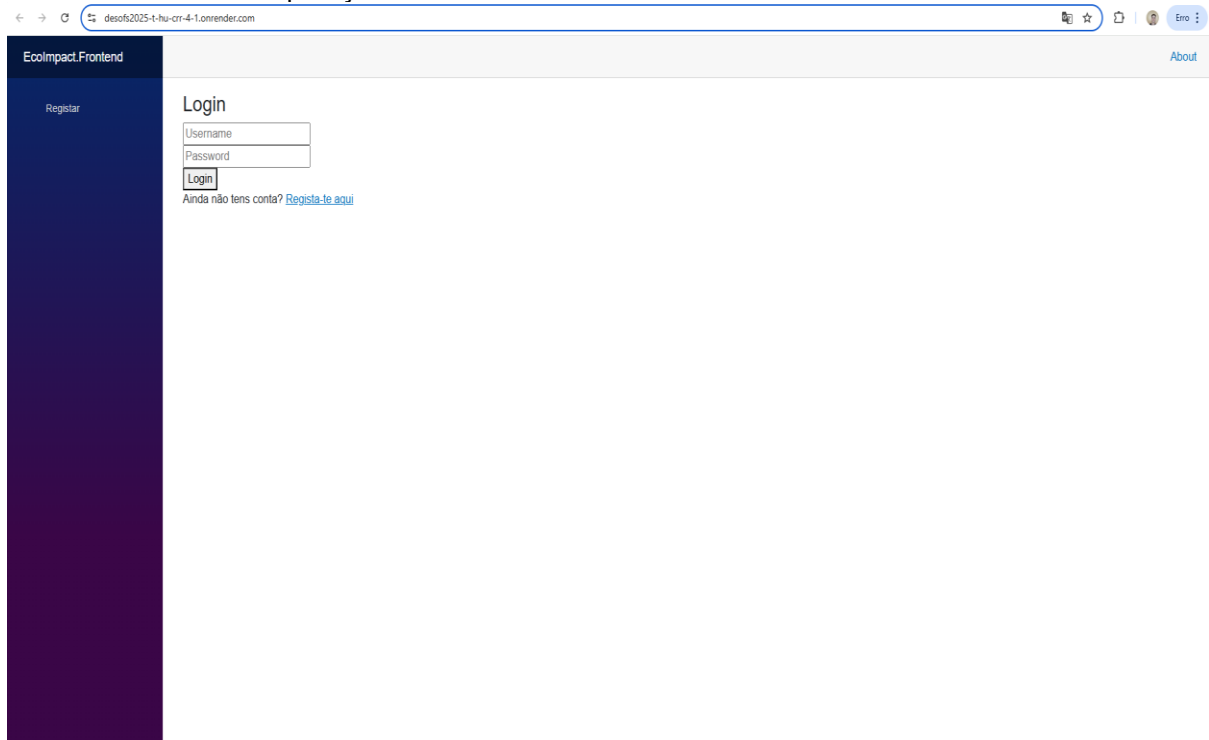
O *workflow* é configurado com **workflow_dispatch**, permitindo que se decida quando iniciar o processo de *deploy*. Esta abordagem é mais segura do que a execução automática por *push*, especialmente em contextos de ambientes mais sensíveis.

- **Swagger UI** da API EcolImpact.API, com todos os endpoints documentados e disponíveis:



O Swagger UI disponibiliza uma interface de exploração e teste da API, que foi utilizada no desenvolvimento da aplicação.

- **Front-end da aplicação**



Interface desenvolvida com **Blazor WebAssembly** e comunica com a **API** através de endpoints **REST** *autenticados*. Funcional e confirma que os módulos *frontend* e *backend* estão corretamente integrados e em execução no ambiente *cloud*. Além de permitir testar (de modo simplista) a autenticação de utilizadores.