

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

Introdução	4
Análise e Design	6
Modelo de Domínio	6
Functional and Non-functional Requirements	8
Architecture, Design and Threat	Error! Bookmark not defined.
Authentication	Error! Bookmark not defined.
Session Management	Error! Bookmark not defined.
Access Control	Error! Bookmark not defined.
Data Protection	Error! Bookmark not defined.
Communication	Error! Bookmark not defined.
Malicious Code	Error! Bookmark not defined.
Files and Resources	Error! Bookmark not defined.
Configuration	Error! Bookmark not defined.
Architecture, Design and Threat	Error! Bookmark not defined.
Authentication	Error! Bookmark not defined.
Session Management	Error! Bookmark not defined.
Access Control	Error! Bookmark not defined.
Data Protection	Error! Bookmark not defined.
Communication	Error! Bookmark not defined.
Malicious Code	Error! Bookmark not defined.
Files and Resources	Error! Bookmark not defined.
Configuration	Error! Bookmark not defined.
Use Cases	10
Arquitetura	11
‘Threat Modeling’	12
Integração do Processo de Threat Modeling	12
Descrição dos Níveis do Processo Representado	13
1. Identify Potential Threats and Vulnerabilities	13
2. Assess Impact and Likelihood	14
3. Develop Mitigation Strategies	14
4. Implement and Monitor	14
Diagrama de Fluxo de Dados (DFD)	15
Nível 0	15

Nível 1.....	16
ASVS.....	18
Architecture, Design and Threat	18
Authentication.....	18
Session Management	18
Access Control	19
Data Protection	19
Communication	19
Malicious Code	20
Files and Resources	20
Configuration.....	20
Validation and Sanitization.....	21
Stored Cryptography	21
Error Handling and Logging.....	22
API and Web Service Security.....	22
Ameaças	23
Categorização	23
Árvores de ameaça.....	24
Categorização	27

Fase 1

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.

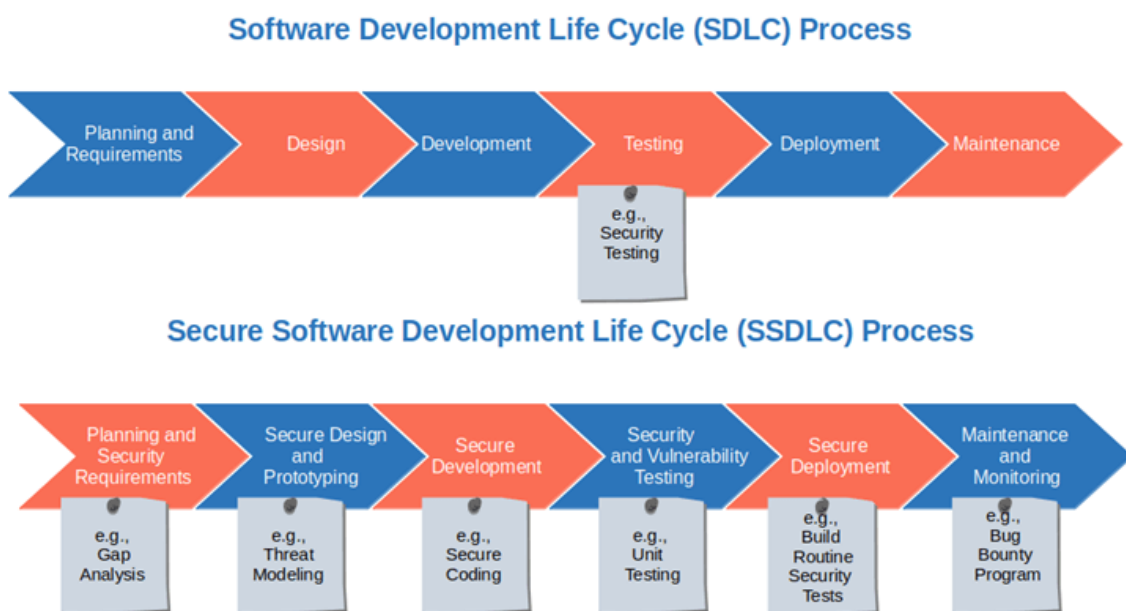


Figura 1 - <https://codesigningstore.com/wp-content/uploads/2023/02/secure-software-development-life-cycle-process.png>

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 correcções 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 detectar 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.

Neste contexto, o projeto desenvolvido consiste num **Simulador de Impacto Ambiental**, cujo principal objetivo é permitir que os utilizadores compreendam o impacto ambiental das suas escolhas diárias (como transporte, alimentação ou consumo de energia), incentivando hábitos mais sustentáveis.

A aplicação terá funcionalidades como o **cálculo da pegada de carbono**, promovendo a consciencialização e o envolvimento com práticas ecológicas. Tendo sido assumido o cenário da sua aplicação num âmbito 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.

Ao longo deste relatório, será apresentado todo o processo de desenvolvimento seguro desta aplicação, incluindo as fases **de análise e design, modelação de ameaças (Threat Modeling), implementação, testes e deployment**, com foco na integração de boas práticas de segurança em todas as etapas.

Análise e Design

Neste capítulo, é descrito o processo de **análise e desenho** que preparou a fase de implementação do projeto. Inicialmente, será explorado o domínio do problema, com o apoio do respetivo **Modelo de Domínio**, que representa a interação entre os diferentes componentes e entidades que integram o sistema.

Modelo de Domínio

Na figura seguinte é apresentado o domínio da ação do projeto descrito, contendo os intervenientes necessários.

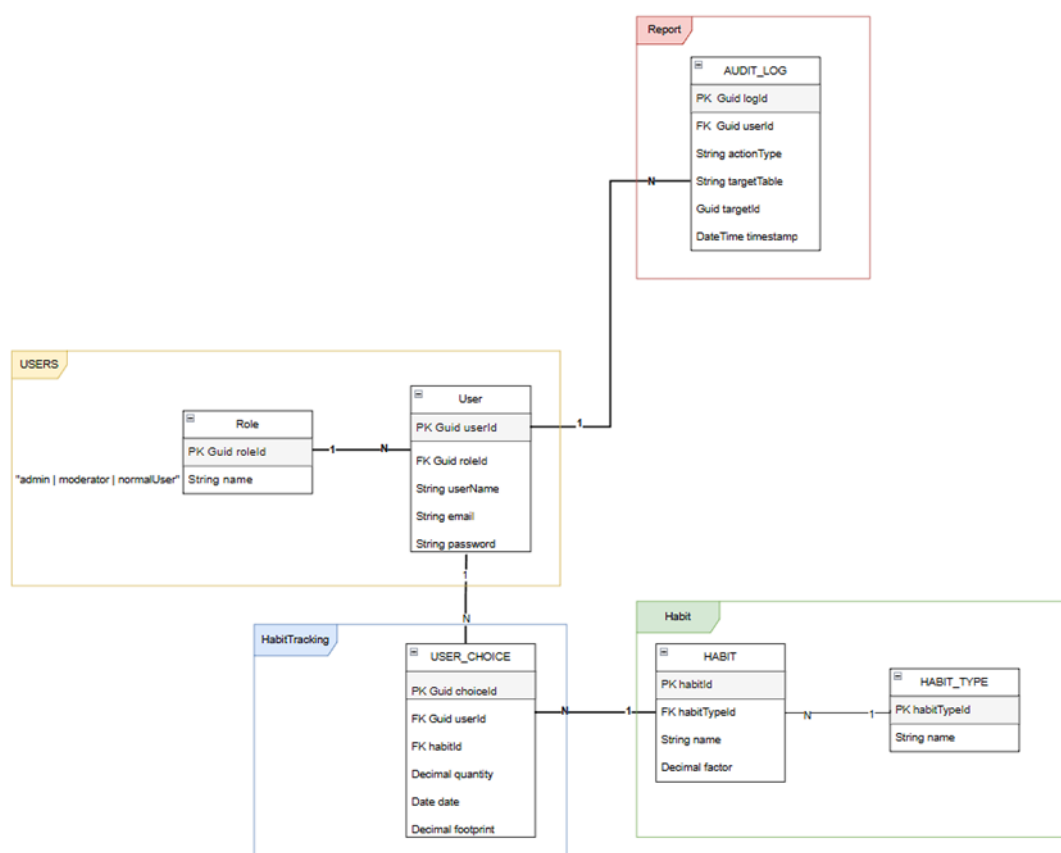


Figura 2 - Modelo de Domínio

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.

Functional and Non-functional Requirements

De seguida, são apresentados os **requisitos do sistema**, organizados em duas categorias principais: **requisitos funcionais**, que descrevem as funcionalidades do sistema, e **requisitos não funcionais**, que definem restrições técnicas, de desempenho, de usabilidade e de segurança.

Functional Requirements

RF01 – Registo e Autenticação

- O sistema deve permitir o registo de novos utilizadores com nome, e-mail e palavra-passe.
- O sistema deve permitir login com verificação por token JWT.
- O sistema deve permitir recuperação de palavra-passe por e-mail.
- O sistema deve distinguir três tipos de utilizador: Admin, Analista (Moderador) e Utilizador Padrão.

RF02 – Gestão de Perfil

- O utilizador deve poder editar nome, e-mail e palavra-passe.
- O utilizador deve poder eliminar a sua conta.

RF03 – Submissão de Hábitos

- O utilizador deve poder inserir dados diários relacionados com:
 - Transporte
 - Alimentação
 - Energia
 - Consumo
- O sistema deve validar e armazenar estes dados.

RF04 – Cálculo da Pegada de Carbono

- O sistema deve calcular automaticamente a pegada de carbono com base nos dados submetidos.
- Os cálculos devem ser atualizados sempre que os dados do utilizador forem alterados.

RF05 – Visualização e Relatórios

- O sistema deve gerar relatórios gráficos e textuais da pegada de carbono.
- O utilizador deve poder ver um histórico mensal e anual.

- O sistema deve apresentar os dados em formato de gráfico e JSON exportável.

RF06 – Sugestões e Gamificação

- O sistema deve fornecer sugestões personalizadas para reduzir a pegada de carbono.
- O utilizador deve poder aceitar desafios semanais.
- O sistema deve permitir comparar resultados com médias da comunidade (anonimizadas).

RF07 – API e Integração

- O backend deve expor endpoints RESTful autenticados.
- Deve permitir integrações com APIs externas de emissão de carbono (ex: footprint APIs).
- Os dados submetidos devem estar disponíveis via API para o frontend.

RF08 – Administração e Moderação

- Administradores devem poder gerir categorias de hábitos.
- Moderadores devem visualizar relatórios agregados sem dados sensíveis.
- O sistema deve filtrar e alertar para dados irrealistas.

Non-functional Requirements

RNF01 – Segurança

- As passwords devem ser encriptadas (bcrypt).
- Toda a comunicação deve usar HTTPS.
- O sistema deve proteger contra: SQL Injection, XSS, CSRF e brute force.
- Deve usar autenticação e autorização para todas as rotas sensíveis.
- O sistema deve incluir logs de acesso e mecanismos de deteção de anomalias.

RNF02 – Desempenho e Escalabilidade

- O sistema deve suportar pelo menos 10.000 utilizadores simultâneos.
- As respostas da API devem demorar no máximo 500ms para 95% das chamadas.
- O sistema deve escalar horizontalmente (baseado em containers).

RNF04 – Confiabilidade e Manutenção

- O sistema deve suportar rollback de deploys via CI/CD.

- Deve manter backups automáticos diários.
- Logs devem ser persistidos e pesquisáveis.
- Deve ser possível fazer atualizações sem downtime.

Use Cases

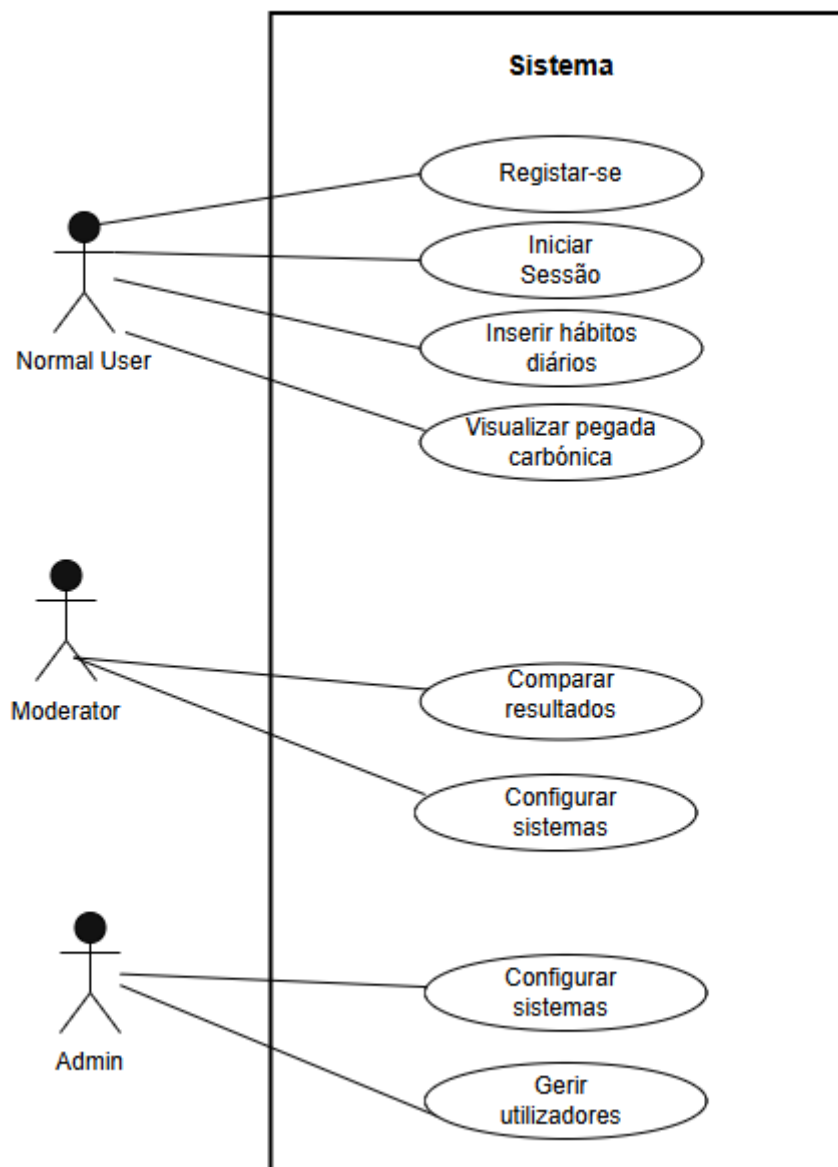


Figura 3 - Diagrama Casos de Uso

As funcionalidades do sistema são as seguintes:

- **US1:** Eu, como Utilizador, pretendo **registar-me** no sistema para começar a usá-lo.
- **US2:** Eu, como Utilizador, pretendo **iniciar sessão** para aceder à minha conta.
- **US3:** Eu, como Utilizador, pretendo **inserir os meus hábitos diários** (ex.: transporte, alimentação, consumo) para obter a minha pegada de carbono.
- **US4:** Eu, como Utilizador, pretendo **visualizar a minha pegada carbónica** para compreender o impacto das minhas escolhas.
- **US7:** Eu, como Moderador, pretendo **aceder a relatórios agregados e comparações estatísticas** para analisar padrões de comportamento dos utilizadores.
- **US8:** Eu, como Moderador, pretendo **configurar o sistema** (parâmetros de cálculo, categorias de hábitos) de forma a adaptar o simulador à realidade.
- **US9:** Eu, como Administrador, pretendo **configurar o sistema** ao nível global, incluindo gestão de categorias e regras base.
- **US10:** Eu, como Administrador, pretendo **gerir os utilizadores** (ex.: ativar, remover contas, atribuir ou remover permissões).

Atribuição do score

A atribuição de valores, que determina o “score ecológico” de cada utilizador foi considerado de uma forma simplista, em que apenas se soma um valor pré-estabelecido em base de dados com a fórmula **score = Math.Max(0, 100 - (footprint * 10))**.

Para uma solução final, teríamos que ter utilizado a formula determinística da pegada ecológica, que mede a quantidade de recursos naturais que o indivíduo consome para manter o seu estilo de vida expresso em **hectares globais (gha)**, e tendo por comparativo o valor extrapolado da capacidade do planeta (que é cerca de 1,7 gha por pessoa).

Contudo, dada a especificidade da cadeira e do objetivo ser a aplicação de framework, segurança e testes de segurança à aplicação esta componente arquitetural não se torna relevante. Ainda assim, avaliamos a formulação e identificamos qual o esforço e requisitos que seriam necessários para a sua implementação.

Arquitetura

O sistema desenvolvido para o projeto **Simulador de Impacto Ambiental** é composto por dois componentes principais: o **backend** e a **base de dados**.

Esta aplicação cliente comunica com o **backend** através de chamadas HTTP/HTTPS, trocando dados em formatos seguros. O **backend** é responsável por receber e processar todos os pedidos enviados, validando-os, aplicando a lógica de negócio e assegurando a aplicação correta dos controlos de segurança definidos.

Além disso, o backend interage com uma **base de dados** relacional, onde são armazenadas informações persistentes como dados de utilizadores, registos de hábitos diários, resultados de pegada de carbono e configurações do sistema.

Em casos específicos, como a gestão de uploads ou documentos gerados, o backend poderá também interagir com um **sistema de ficheiros** ou uma **API externa de gestão de ficheiros**, respeitando sempre os princípios de segurança de dados e isolamento de componentes.

Desta forma, a arquitetura adotada assegura uma separação clara de responsabilidades entre a camada de lógica de negócio (backend) e a camada de persistência de dados (base de dados), promovendo a escalabilidade, segurança e manutenção eficiente do sistema.

‘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.

Neste projeto, foi adotado o modelo STRIDE como base para a identificação das ameaças, categorizando-as em seis grupos:

- Spoofing (Falsificação de identidade)
- Tampering (Adulteração de dados)
- Repudiation (Repúdio de ações)
- Information Disclosure (Divulgação não autorizada de informação)
- Denial of Service (Negação de serviço)
- Elevation of Privilege (Elevação de privilégios)

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.

Integração do Processo de Threat Modeling

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 4 - Threat Modeling Process

Descrição dos Níveis do Processo Representado

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 deteção 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;
- Deteção 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.

Diagrama de Fluxo de Dados (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.

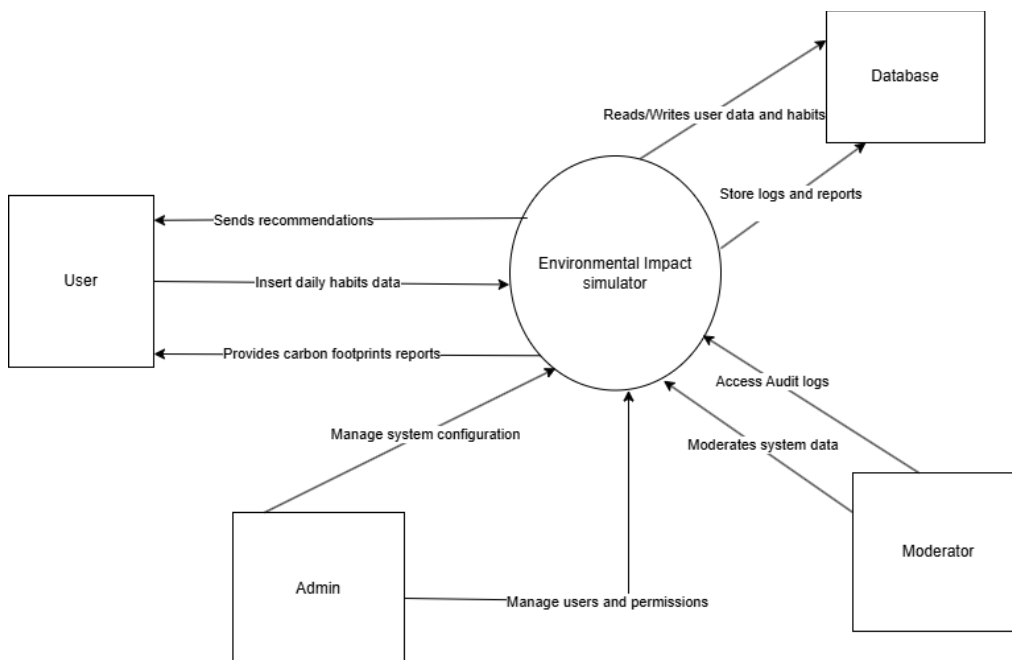


Figura 5 - DFD Nível 0

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).

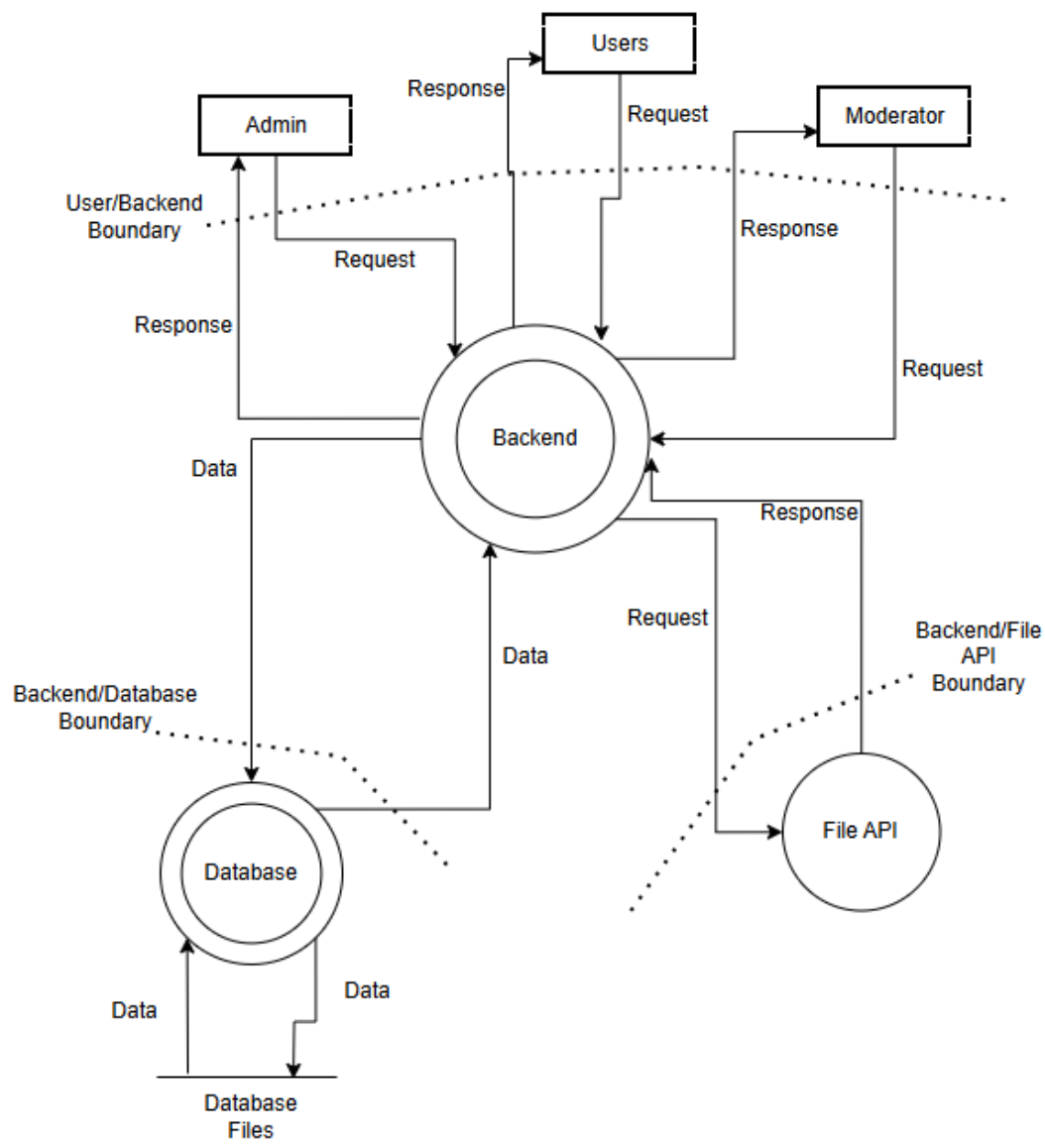


Figura 6 - DFD Nível 1

ASVS

Tomando por base a checklist e guidelines que a OWASP Application Security Verification Standard (ASVS) Framework, priorizando para a solução.

Architecture, Design and Threat

A arquitetura do sistema foi planeada com foco em segurança, seguindo o ciclo de vida SSDLC, com integração de threat modeling e controlo de riscos em todas as fases de desenvolvimento. O sistema adota:

- Uma arquitetura modular, com componentes isolados e documentados;
- Threat modeling com base em STRIDE para identificar ameaças recorrentes;
- Definição clara de limites de confiança e de dados sensíveis;
- Centralização de controlos reutilizáveis para evitar duplicação ou ausência de mecanismos de proteção;
- Checklist interna de desenvolvimento seguro para todos os programadores e validadores.

Authentication

A autenticação é implementada com foco em segurança desde a criação até à recuperação de contas, garantindo proteção contra automação e armazenamento seguro. O sistema adota:

- Política de Passwords com mínimo de 12 caracteres, medidor de força, permissão de colar e verificação contra leaks;
- Proteção contra ataques automatizados, com limite de tentativas, CAPTCHA e throttling por IP;
- MFA obrigatório em operações críticas, com suporte a FIDO2, TOTP e notificações seguras;
- Gestão segura do ciclo de vida dos autenticadores, com expiração e revogação controlada de tokens;
- Armazenamento de credenciais com Argon2id/bcrypt/PBKDF2, salt e pepper, e uso de HSM/Vault;
- Recuperação de conta segura, sem perguntas de conhecimento, com tokens únicos e verificação MFA.

Session Management

A adopção de mecanismos de gestão de sessões, tendo-se optado por incluir:

- Session Token Security: garantia que os tokens de sessão não são apresentados quer em URLs ou mensagens de erros;
- Session Binding: após autenticação de utilizador novos tokens são gerados, com garantia de aleatoriedade e imprevisibilidade.
- Session Storage: o armazenamento dos tokens de sessão com metodologia de cookies com atributos seguros ('Secure', 'HttpOnly' e 'SameSite').

- Session Termination: após logout do utilizador, os tokens são anulados para evitar session hijacking.
- Token-based Session Management: adopção de assinatura digitais e encriptação para evitar stateless session tokens.

Access Control

O acesso dos utilizadores deve respeitar o princípio da restrição:

- Enforcement Points: implementar controlos de acesso em pontos como gateways ou servidores;
- Least Privilege: garantir o cumprimento do princípio do menor privilégio 'Principle of Least Privilege (PoLP)';
- Attribute-based Access Control: o acesso a determinadas funcionalidades ou dados, não estar autorizado através de contas de utilizador;
- Fail Securely: no caso de erro ou excepção na aplicação, deverá ser adoptado o default de negar em vez de autorizar acessos;
- Anti-CSRF Mechanisms: implementar mecanismos de protecção contra ataques 'Cross-Site Request Forgery (CSRF)', que garantam no acesso a dados sensíveis ou APIs sejam feitos por utilizadores e não por mecanismos de "manipulação" de sessões no browser. Seja por os atributos nos cookies (SameSite attribute), tokens (CSRF Tokens – Synchronizer Token Pattern) ou implementação de CAPTCHA.

Data Protection

Neste tópico, considera-se a protecção dos dados sensíveis de acesso não-autorizado e brechas:

- Sensitive Data Identification: identificar e classificar os dados por níveis de protecção/criticidade;
- Protection Requirements: aplicar como requisitos encriptação, integridade, retenção, privacidade e confidencialidade nos dados identificados como sensíveis;
- Client-side Data Protection: aplicar mecanismos de anti-caching que garantem que os dados sensíveis não são armazenados pelo browser;
- Sensitive Data Transmission: o envio dos dados sensíveis não é enviado em parâmetros de query strings, mas sim na message body ou headers;
- Data Retention: os dados sensíveis devem ser apagados de forma automática, sempre que não sejam necessários.

Communication

As comunicações entre componentes da aplicação estão protegidas por ligações seguras (TLS), incluindo comunicações internas entre microserviços, bases de dados, APIs e terceiros. As boas práticas implementadas incluem:

- Uso exclusivo de TLS 1.2 ou superior;
- Ciphers seguros e certificados confiáveis, com validação de CA;

- Revogação de certificados com OCSP stapling quando aplicável;
- Logging de falhas de comunicação TLS e rejeição de protocolos inseguros;
- Isolação de tráfego por rede entre ambientes e containers distintos.

Malicious Code

A prevenção de código malicioso e funcionalidade indesejada é assegurada com:

- Análise estática e dinâmica de código fonte (SAST e DAST);
- Revisões obrigatórias de pull requests e rastreabilidade de alterações via Git;
- Verificação automática de dependências (ex.: Snyk, Trivy);
- Proibição de dependências obsoletas, binários não documentados, contas backdoor ou funções de debug abertas em produção;
- Validação de que não existem "easter eggs", "subdomain takeovers", nem funcionalidades escondidas ou inseguras nos clientes ou no backend.

Files and Resources

A gestão de ficheiros segue práticas rigorosas para prevenir DoS, execução arbitrária e SSRF, garantindo integridade e confidencialidade. O sistema adota:

- Controlo de uploads com limite de tamanho (ex.: 100 MB) e quota por utilizador (ex.: 500 MB);
- Validação de integridade via assinatura MIME e alerta em caso de ficheiros maliciosos;
- Execução segura, com renomeação via UUID, armazenamento fora do web-root e varredura anti-malware;
- Transferência controlada, forçando headers seguros e bloqueando extensões perigosas;
- Defesa contra SSRF, com allow-list de domínios e monitorização de tráfego de saída.

Configuration

Por gestão da configuração, considera-se que a aplicação seja devidamente parametrizada e mantida:

- Segregation of Components: a aplicação de controlos de segurança, regras de firewall, API gateways, e reverse proxies para segregar os componentes por níveis;
- Build Pipeline: garantir que o build avisa quanto a componentes não-seguros ou antigos, e toma medidas preventivas;
- Sandboxing and Isolation: isolar os componentes em ambientes como containers, sandbox, nomeadamente os componentes críticos da aplicação de acesso de atacantes;
- Unsupported Technologies: evitar a utilização de tecnologias antigas, ou identificadas como inseguras.

Validation and Sanitization

A solução implementa mecanismos robustos de validação e sanitização de entradas, de acordo com as recomendações da OWASP ASVS e seguindo o ciclo de vida seguro de desenvolvimento (SSDLC):

- **Input Validation:** Todas as entradas do utilizador são validadas no backend, com utilização de listas de permissões (“allowlist”) para formatos esperados e restrição de tipos de dados. Além disso também garantir o preenchimento de todos os campos essenciais e obrigatórios.
- **Output Encoding:** Dados que são devolvidos em respostas HTTP são devidamente codificados, prevenindo ataques como Cross-Site Scripting (XSS).
- **Sanitization:** Para dados que aceitam conteúdo mais flexível (ex.: campos de texto), são aplicados filtros de sanitização para remover scripts e caracteres maliciosos.
- **Automated Validation:** Ferramentas automatizadas de validação de payloads e esquemas (ex.: JSON Schema Validator) são integradas no pipeline de CI/CD.
- **Strict HTTP Method Validation:** Apenas métodos HTTP estritamente necessários (GET, POST, etc.) são aceites, com rejeição e logging das tentativas indevidas.

Stored Cryptography

Os dados sensíveis são protegidos com criptografia forte em repouso, garantindo a integridade e confidencialidade das informações:

- **Data Classification:** Dados são classificados quanto à sua criticidade, definindo requisitos para criptografia.
- **Encryption at Rest:** Utilização de algoritmos robustos (AES-256) para a criptografia de informações sensíveis armazenadas em bases de dados. (Como por exemplo passwords)
- **Key Management:** As chaves criptográficas são geridas de forma segura, com políticas de rotação periódica e segregação de acesso.
- **No Plain text Storage:** Nunca são armazenadas senhas ou informações sensíveis em texto plano. Senhas, por exemplo, utilizam algoritmos de hashing adequados (bcrypt, Argon2).
- **Integrity Check:** Verificação de integridade para dados criptografados, prevenindo adulterações.

Error Handling and Logging

A solução adota práticas seguras para tratamento de erros e logging como a utilização de um middleware que garante uma globalização das exceptions da aplicação, evitando a exposição de informações internas e garantindo rastreabilidade para auditoria:

- No Sensitive Data in Errors: Mensagens de erro não expõem detalhes técnicos, stack traces ou informações sensíveis (como paths internos ou variáveis).
- Fail Securely: Por padrão, operações falham de forma segura, sem revelar lógica interna ou comprometer a segurança.
- Centralized Logging: Uso de logging centralizado com registros detalhados de atividades críticas e tentativas de acesso não autorizado.
- Log Protection: Logs são protegidos contra acesso não autorizado e armazenados em sistemas monitorados.
- Alerting: Implementação de alertas para eventos suspeitos, como tentativas de acesso inválidas ou padrões anômalos.

API and Web Service Security

Foram aplicados controlos rigorosos de segurança para APIs e web services, alinhados com o OWASP ASVS e as melhores práticas do setor:

- Secure HTTP Headers: Todas as respostas incluem headers de segurança essenciais (Content-Type, X-Content-Type-Options: nosniff, Strict-Transport-Security, Referrer-Policy, Content-Security-Policy, entre outros).
- CORS Policy: Política CORS restritiva, aceitando apenas domínios confiáveis, rejeitando null origins e prevenindo uso indevido de *.
- Authentication and Authorization: Todas as APIs exigem autenticação forte, e o acesso é controlado com base em perfis e permissões.
- CSRF Protection: Mecanismos anti-CSRF implementados em endpoints críticos, utilizando SameSite em cookies e/ou tokens anti-CSRF.
- Rate Limiting and Throttling: Aplicação de rate limits e mecanismos de detecção de abuso em APIs públicas e sensíveis.

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:

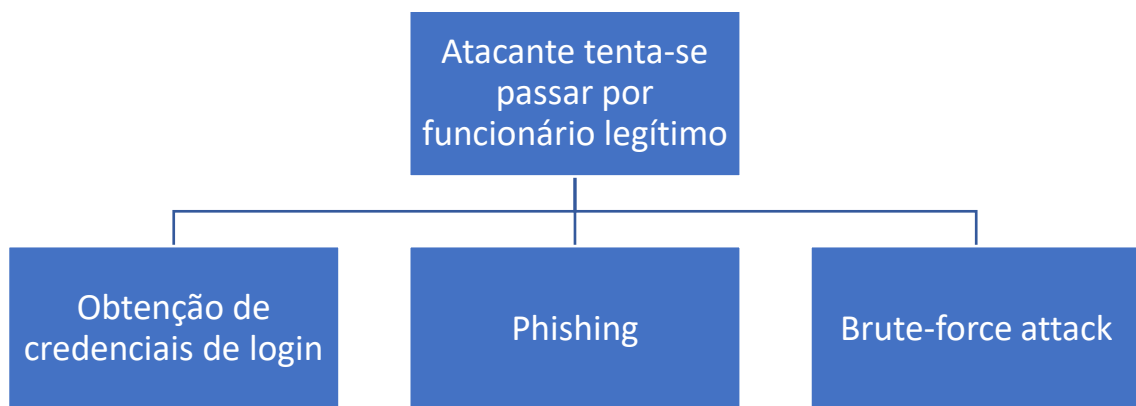
- **Abrangência**: O STRIDE categoriza ameaças em seis tipos principais: Falsificação de Identidade (**Spoofing**), Adulteração (**Tampering**), Repúdio (**Repudiation**), Divulgação de Informação (**Information Disclosure**), Negação de Serviço (**Denial of Service**) e Elevação de Privilégios (**Elevation 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 ameaça

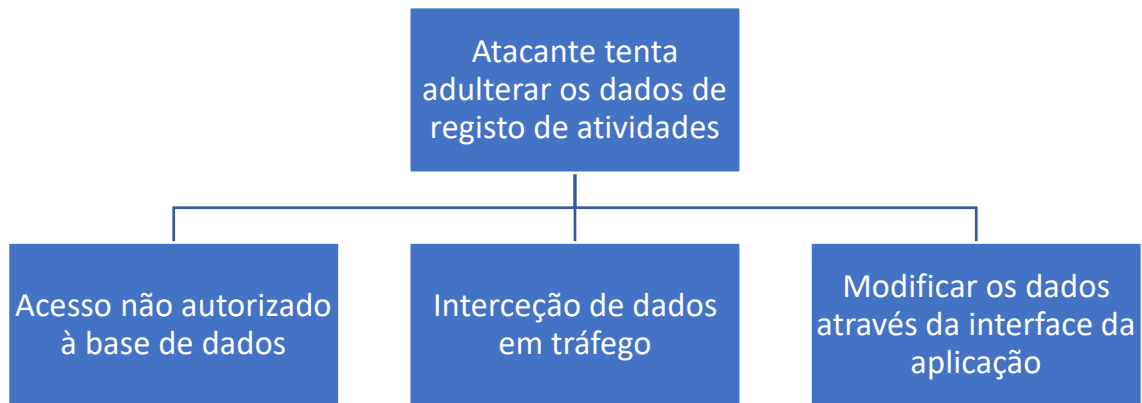
O recurso a árvores de ameaça fornece uma descrição da segurança dos sistemas, pela exploração de vários ataques tais como:

Atacante tenta-se passar por funcionário legítimo



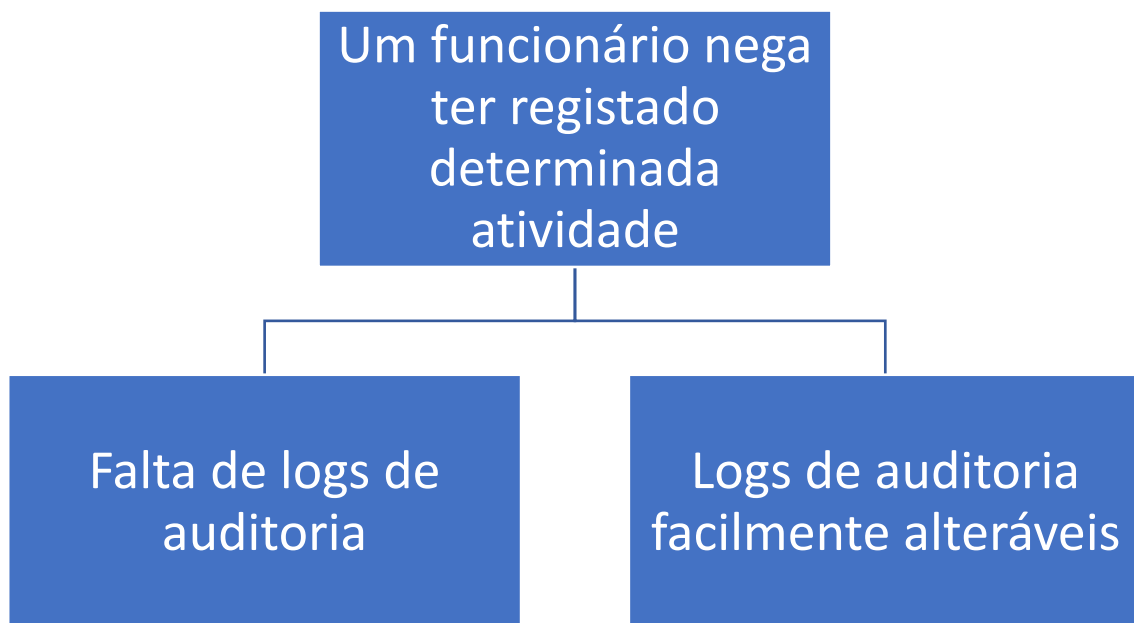
Para este cenário identificou-se três métodos de ação, podendo-se implementar como medidas de contenção a implementação de autenticação multifator, a formação dos utilizadores em como evitar ações de phishing. Ou mesmo a implementação de ações que limitem o brute-force ao nível do login, cache e storage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

Atacante tenta alterar os dados de registo de atividades



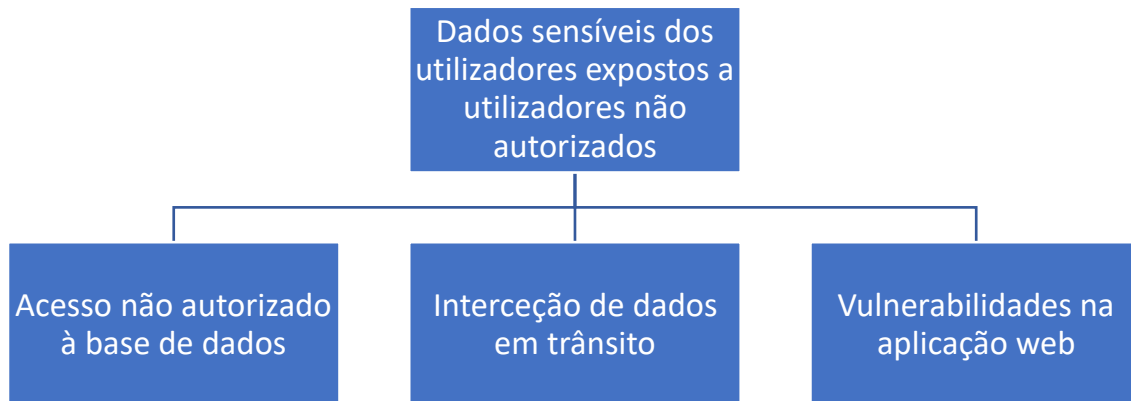
Como formas de mitigar estas potenciais vulnerabilidades deve ser adotada a implementação de criptografia dos dados e controlos de acesso rigorosos. As comunicações serem todas por HTTPS (com TLS), e a existência logs de auditoria a todos os acessos.

Um funcionário nega ter registado determinada atividade



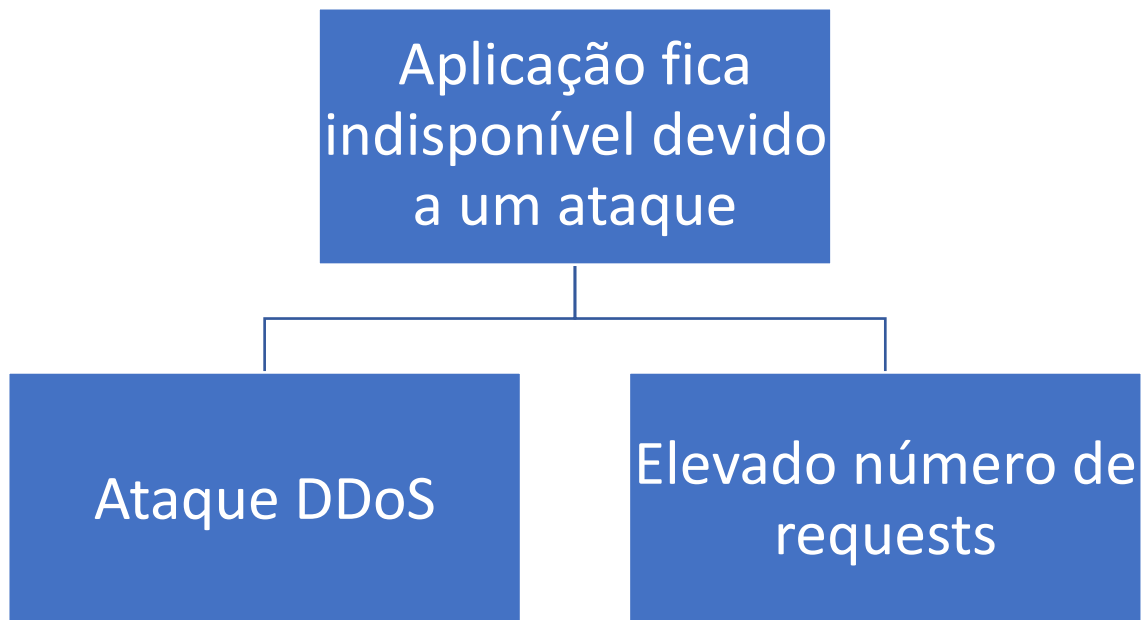
Na ausência de logs de auditoria, uma ação é implementá-los. Garantir a sua segurança é atingível com a implementação de assinaturas digitais nestes.

Dados sensíveis dos utilizadores expostos a utilizadores não autorizados



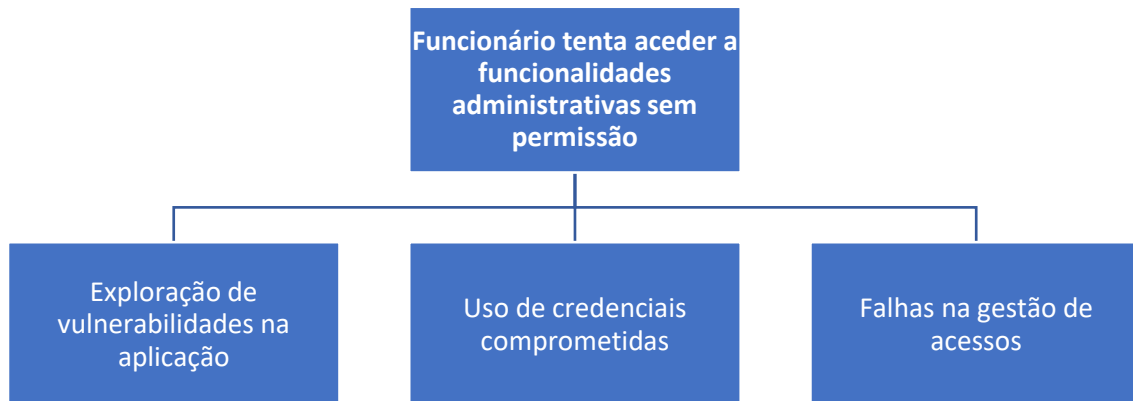
Abordagens a esta vulnerabilidade, podem ser mitigadas com a implementação de controlos de acesso baseados em funções (RBAC), utilização de criptografia TLS e ter um plano regular de testes de penetração (com a respetiva implementação das recomendações).

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

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

Fase 2-Sprint 1

Introdução

Este relatório documenta as principais tarefas realizadas no desenvolvimento do projeto **EcoImpact**, com foco na estrutura de código, testes e integração de pipelines de segurança. O sistema visa calcular a pegada de carbono dos utilizadores com base nos seus hábitos diários, fornecendo sugestões para redução de impacto ambiental.

O projeto está organizado segundo o paradigma DDD (Domain-Driven Design), com uma arquitetura modular composta por diferentes agregados.

Estrutura de Projeto

A solução está estruturada em três projetos principais:

- **EcoImpact.API**: Camada de exposição HTTP (controladores e endpoints)
- **EcoImpact.DataModel**: Modelos de dados e classes de domínio
- **EcoImpact.Tests**: Testes automatizados usando MSTest

Implementação de Testes Automatizados

Os testes unitários foram implementados em EcoImpact.Tests, com uso extensivo de:

- **Moq** para mock de dependências (como DbContext e IPasswordService)
- **MSTest** como framework de testes

Testes principais:

- Criação de utilizadores (UserService)
- Fetch de utilizadores por ID e username
- Eliminação de utilizadores e validação de comportamentos
- Exportação de dados de utilizadores para JSON
- Importação de Habit Types
- Log in e falha do mesmo
- Após 4 tentativas de login falhadas o user fica bloqueado
- Verificar que a lista de users exportada não contém passwords

Pipelines de CI/CD com GitHub Actions

Foram criados vários workflows automatizados na pasta .github/workflows:

Build e Testes Unitários

- Workflow: build.yml
- Ações:
 - dotnet restore
 - dotnet build com configuração Release
 - dotnet test com cobertura de código (coverlet + cobertura)

Análise Estática com SonarCloud (SAST)

- Workflow: sonarcloud.yml
- Ferramenta: dotnet-sonarscanner
- Ações:

- Gera relatório de cobertura via Coverlet
- Submete resultados para SonarCloud
- Verifica métricas de código (code smells, duplications, coverage)

Análise Dinâmica com OWASP ZAP (DAST)

- Workflow: zap.yml
- Executa zap-baseline.py via Docker, analisando a API exposta no porto 7020
- Exporta resultados para HTML, Markdown e JSON
- Corre dentro de --network ecoimpact_default para acesso ao container ecoimpact-api

Software Composition Analysis (SCA)

- Workflow: snyk.yml
- Ferramenta: snyk/actions/dotnet
- Comando: snyk test --file=EcoImpact.API/EcoImpact.API.csproj --sarif-file-output=snyk-api.sarif
- Deteta vulnerabilidades em dependências NuGet