**Universidade de Aveiro**
**Ano** 2025

**Ana Luísa Silva Loureiro**

**Filipe Ramalho de Oliveira**

**Gonçalo Gomes Monteiro**

**Miguel Soares Francisco**

**Miguel Oliveira Pinto**

# EasyReg – Plataforma de Gestão de Normas de Segurança

**Universidade de Aveiro**
**Ano 2025**

**Ana Luísa Silva Loureiro**

**Filipe Ramalho de Oliveira**

**Gonçalo Gomes Monteiro**

**Miguel Soares Francisco**

**Miguel Oliveira Pinto**

# EasyReg – Plataforma de Gestão de Normas de Segurança

Documento apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à conclusão da unidade curricular Projeto em Informática, realizada sob a orientação científica do Doutor João Rafael Almeida, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e Mariana Andrade, Licenciada em Engenharia Informática.

**palavras-chave**

Conformidade de segurança, Gestão de evidências, *Frameworks* regulamentares, Automatização de auditorias, Gestão de conformidade

**resumo**

O EasyReg é uma plataforma concebida para simplificar e automatizar a gestão de conformidades de segurança em várias normas e *frameworks*. Visa ajudar as organizações a garantir que as suas práticas estão alinhadas com os requisites regulamentares e da indústria, oferecendo um sistema centralizado para gerir controlos, políticas, evidências e auditorias.
A plataforma suporta várias funções de utilizador, incluindo colaboradores, auditores internos, auditores externos, proprietários de projetos e administradores. Cada cargo tem responsabilidades específicas, como o carregamento de evidências, a revisão de controlos ou a realização de auditorias. As organizações são representadas como inquilinos (*Tenants*) e cada projeto associado a um inquilino pode estar ligado a uma *framework* de segurança, como a OWASP, ASVS ou a ISO 27001. Os controlos e subcontrolos definem os requisitos a implementar enquanto as políticas fornecem orientações ou exigências. Os utilizadores podem associar políticas a controlos e submeter evidências que demonstrem a conformidade. Os auditores internos são responsáveis pela análise e validação das evidências submetidas, garantindo que estão em conformidade com os requisitos da *framework*. Posteriormente, os auditores externos revêm estas validações, garantindo uma camada adicional de confiança e transparência.
Ao centralizar os processos de conformidade e ao definir responsabilidades claras, o EasyReg permite que as equipas se mantenham organizadas, reduzam o trabalho manual e sigam um caminho consistente e organizado para a conformidade.

**keywords**

Security Compliance, Evidence Tracking, Regulatory Frameworks, Audit Automation, Compliance Management

**abstract**

EasyReg is a platform designed to simplify and automate security compliance management across various standards and frameworks. It helps organizations streamline the process of ensuring that their practices meet regulatory and industry requirements by offering a centralized system to manage controls, policies, evidence, and audits. The platform supports multiple user roles, including contributors, internal auditors, external auditors, project owners, and administrators. Each role has specific responsibilities, such as uploading evidence, reviewing controls, or conducting audits. Organizations are represented as tenants, and each project under a tenant can be linked to a security framework like OWASP ASVS or ISO 27001. Controls and subcontrols define what must be implemented, and policies provide guidelines or requirements. Users can associate policies with controls and submit evidence to demonstrate compliance. Internal auditors are responsible for reviewing and validating the submitted evidence, ensuring it aligns with the framework's requirements. External auditors later review these internal validations, ensuring an additional layer of trust and transparency.

By centralizing compliance efforts and defining clear responsibilities, EasyReg enables teams to stay organized, reduce manual work, and maintain a consistent and auditable path toward regulatory compliance.

**reconhecimento do uso de ferramentas IA**

**Reconhecimento do uso de tecnologias e ferramentas de Inteligência Artificial (IA) generativa, softwares e outras ferramentas de apoio.**

Reconheço o uso do ChatGPT 4o (Open AI *https://chat.openai.com*) para ajudar com a formalização do texto em português, e para a tradução de alguns excertos e palavras, a utilização do DeepL Translator (DeepL https://www.deepl.com/en/translator).

# Índice

# Índice de Figuras

# Índice de Abreviaturas

RBAC - Role-Based Access Control
OWASP - Open Web Application Security Project
IAM - Identity and Access Management
SIEM - Security Information and Event Management
MVC - Model-View-Controller
ISO/IEC 27001 - International Organization for Standardization / International Electrotechnical Commission 27001
NIST - National Institute of Standards and Technology
API - Application Programming Interface
SSO - Single Sign-On
CRUD - Create , Read, Update, Delete

# Capítulo 1

# Introdução

"Security is a process, not a product." – Bruce Schneier [1]

À medida que o nosso mundo se torna cada vez mais digital, torna-se também evidente a necessidade crescente de segurança da informação. Com o crescimento das organizações e a adoção de infraestruturas tecnológicas cada vez mais complexas, a conformidade com *frameworks* e normas de segurança reconhecidas tornou-se uma parte essencial da gestão de riscos. *Frameworks* regulamentares como a OWASP ASVS, ISO/IEC 27001 e NIST fornecem controlos técnicos e processos estabelecidos que permitem às organizações manterem-se seguras, protegendo dados sensíveis, aumentando a privacidade e reduzindo as vulnerabilidades dos seus sistemas.

No entanto, alcançar e manter a conformidade implica um processo complexo, envolvendo vários intervenientes, desde gestores de projetos, programadores, técnicos de conformidade e auditores externos, cada um deles com responsabilidades diferentes em matéria de implementação, documentação e subsequente revisão, e, muitas vezes, todo o processo é normalmente um esforço manual compilado através de folhas de cálculo, formulários de avaliação de riscos, documentos desconexos e comunicação ad hoc. Estes procedimentos ineficientes e fragmentados, agravados pela falta de centralização, aumentam significativamente o risco de não conformidade com os requisitos regulamentares, falhas durante auditorias e violações de dados.

## 1.1 Motivação

Apesar da importância crítica da conformidade com os requisitos de cibersegurança, muitas organizações continuam presas a ferramentas desatualizadas e métodos ad hoc para gerir o cumprimento das normas. Devido à falta de integração, a uma experiência de utilizador ineficaz e à ausência de automação, as partes interessadas encontram-se numa posição frágil para monitorizar o progresso, atribuir responsabilidades ou realizar revisões em tempo útil. Esta situação agrava-se significativamente quando as organizações necessitam de adotar múltiplos padrões de segurança em simultâneo.

As plataformas existentes no mercado - Gapps, Drata e Vanta - tentaram colmatar esta necessidade com ferramentas de conformidade baseadas na nuvem. Contudo, estas ferramentas apresentam limitações, seja pela ausência de suporte de *frameworks* personalizados, o facto de terem modelos de dados excessivamente estruturados, pela fraca capacidade multilíngue ou por modelos de licenciamento demasiado comerciais que dificultam a sua adoção por pequenas organizações ou instituições de ensino.

Assim, a plataforma EasyReg foi concebida tendo em conta a flexibilidade, a extensibilidade e a facilidade de utilização. Ao contrário das ferramentas tradicionais, não se limita a disponibilizar *checklists* estáticas de conformidade, mas também fluxos de trabalho dinâmicos que refletem a natureza evolutiva das estruturas de segurança. Ao permitir que diferentes utilizadores - como colaboradores, auditores internos, auditores externos e proprietários de projetos - interajam no mesmo ambiente, a plataforma promove uma colaboração eficiente e transparente ao longo de todo o processo de conformidade.

### 1.1.1 Problemas da plataforma Gapps

O Gapps [2] é uma das plataformas mais utilizadas no domínio da gestão da conformidade de segurança e da automatização de auditorias. Criada pela Darkbanner, oferece uma interface que permite o mapeamento de controlos relacionados com várias *frameworks* conhecidas como ISO 27001, SOC 2 e NIST, permitindo ainda aos utilizadores submeter e avaliar evidências em vários projetos. Suporta também a atribuição de diferentes roles a utilizadores – gestores de projeto, auditores e colaboradores – uma das funcionalidades que adaptamos ao nosso projeto.
Contudo, apesar de ter uma base sólida e relevância no mercado, o Gapps apresenta algumas limitações que restringem o seu uso em contextos mais amplos. Um dos problemas mais evidentes é a sua interface pouco intuitiva, frequentemente sobrecarregada de informação numa única página, com uma fraca hierarquia visual e ausência de percursos claros de navegação entre *frameworks*, controlos e evidências relacionadas. Consequentemente, os utilizadores - especialmente os com menor conhecimento técnico - podem ter dificuldade em localizar informações específicas ou em acompanhar facilmente o progresso global da conformidade dos seus projetos.

Outro aspeto crítico diz respeito ao desempenho e à capacidade de resposta da plataforma. Em contextos que envolvam grandes volumes de dados - como a gestão de centenas de controlos distribuídos por múltiplas *frameworks* e projetos - o Gapps sofre atrasos significativos no carregamento de informação, que se torna especialmente evidente ao aceder às *dashboards*, ao aplicar filtros aos controlos ou ao navegar entre páginas.

Além disso, a falta de modularidade e extensibilidade do Gapps limita a sua capacidade de integração com *frameworks* externas ou com políticas de segurança.
Estas limitações evidenciam claramente a necessidade de uma solução mais moderna, flexível e fácil de utilizar - uma alternativa que pretendemos oferecer com o EasyReg, focando a otimização do desempenho, a simplicidade da interface e o suporte de relações dinâmicas entre políticas, controlos e subcontrolos.

## 1.2 Objetivos

O nosso projeto tem como objetivo conceber e desenvolver uma plataforma de gestão da conformidade fiável, intuitiva e extensível, que supere as limitações das soluções atualmente disponíveis no mercado.

Os objetivos de alto nível do projeto são os seguintes:

**Simplificação da gestão da conformidade -** Oferecer uma experiência intuitiva a utilizadores autorizados, permitindo-lhes definir projetos, associá-los a *frameworks* de segurança e acompanhar a implementação de controlos e políticas ao longo do tempo.

**Gestão de evidências e preparação para auditorias -** Disponibilizar uma plataforma estruturada para que os colaboradores submetam evidências, que possam ser posteriormente validadas por auditores internos e externos, garantindo um processo controlável e auditável.

**Flexibilidade na gestão de *frameworks* -** Suportar múltiplos modelos de segurança e permitir a definição e atualização fácil de controlos, subcontrolos e itens de políticas, de forma personalizada por administradores da plataforma.

**Autenticação e autorização como serviço -** Integrar o Keycloak como solução centralizada de gestão de identidade e acesso, assegurando um processo seguro e baseado em funções para autenticação e autorização em toda a aplicação.

**Arquitetura modular e escalável -** Adotar as melhores práticas da indústria no desenho da arquitetura, de forma a facilitar futuras extensões, adição de funcionalidades, novos serviços e integrações com sistemas externos.

**Experiência de utilizador otimizada -** Fornecer uma interface web moderna, acessível e orientada para o desempenho, apoiada por um modelo de dados claro que permita aos utilizadores navegar autonomamente pelas suas responsabilidades de conformidade e segurança.

Ao atingir estes objetivos, esperamos estabelecer o EasyReg como uma solução abrangente que não só apoia as necessidades organizacionais atuais, mas também se adapta a requisitos futuros à medida que surgem novas normas ou que as existentes evoluem.

## 1.3 Esboço do documento

O presente documento encontra-se organizado em oito capítulos, correspondendo cada um deles a uma fase essencial do processo de conceção, desenvolvimento e avaliação da plataforma EasyReg:

**Capítulo 1 - Introdução:** Este presente capítulo descreve a motivação e inspiração para o desenvolvimento do EasyReg.

**Capítulo 2 - Contexto e Estado da Arte:** Analisa as ferramentas e plataformas existentes no domínio da gestão de conformidades de segurança. Este capítulo destaca as suas principais caraterísticas, identifica as limitações comuns e evidencia a lacuna tecnológica que o EasyReg pretende colmatar.

**Capítulo 3 - Análise de Requisitos do Sistema:** Apresenta os resultados da recolha de requisitos, contemplando tanto os funcionais como os não funcionais. Introduz os principais atores do sistema e estabelece a correspondência entre as histórias de utilizador e casos de utilização, descrevendo as interações esperadas de cada função na plataforma.

**Capítulo 4 - Arquitetura do sistema:** Detalha a estrutura da plataforma, incluindo o modelo de domínio, as decisões de arquitetura e a estratégia de implementação do sistema. Ilustra ainda o modo como os componentes interagem para assegurar as funcionalidades principais.

**Capítulo 5 - Implementação:** Descreve o processo de desenvolvimento da plataforma, com foco nas suas camadas principais: a camada de apresentação, autenticação, negócios e base de dados. São discutidas as principais decisões de implementação e as tecnologias utilizadas.

**Capítulo 6 - Gestão do projeto:** Explica como a nossa equipa foi organizada, as ferramentas de comunicação e os métodos de colaboração que utilizámos ao longo do desenvolvimento e a estratégia de calendarização que adotámos para garantir o progresso e as entregas dentro dos prazos definidos.

**Capítulo 7 - Conclusões:** Resume os principais resultados do projeto, reflete sobre as lições aprendidas e propõe potenciais melhorias ou evoluções futuras para a plataforma EasyReg.

# Capítulo 2

# Contexto e estado da arte

Este capítulo analisa as plataformas, serviços e ferramentas atualmente disponíveis que funcionam de forma semelhante ao EasyReg, ajudando a gerir normas de segurança. Compreender a tecnologia disponível é essencial para poder identificar as limitações dessas soluções, o que abre espaço para o desenvolvimento de uma alternativa mais flexível e de utilização mais intuitiva, como é o caso do EasyReg.

## 2.1 Plataformas existentes

Nos últimos anos, surgiram várias plataformas para ajudar as organizações a alcançar e manter a conformidade com as normas de segurança da informação. Entre as mais reconhecidas encontram-se o Gapps, o Drata e o Vanta.

### 2.1.1 Gapps (por Darkbanner)

O Gapps destaca-se como uma das plataformas mais relevantes neste domínio, tendo sido referida anteriormente como uma das principais fontes de inspiração para o EasyReg. Oferece funcionalidades como o mapeamento de controlos, a submissão de evidências e a gestão de *frameworks*. No entanto, como já foi referido, apresenta limitações de utilização e desempenho, que pretendemos ultrapassar com o EasyReg.

### 2.1.2 Drata

O Drata é uma plataforma de gestão de conformidades automatizada bem conhecida, concebida para *startups* e empresas. Integra-se com fornecedores e ferramentas de nuvem - como o GitHub, AWS e Google Workspace - para monitorizar e automatizar a recolha de evidências.
Apesar da sua robustez em ambientes onde as integrações são uma prioridade, mostra uma transparência limitada na forma como as *frameworks* são modeladas internamente. Além disso, o seu custo elevado e o seu foco empresarial tornam-no inacessível a equipas mais pequenas ou a contextos académicos.

### 2.1.3 Vanta

Vanta é outra solução amplamente utilizada, especializada em simplificar a conformidade para *startups*. Tal como o Drata, ele enfatiza a automação por meio de integrações.
No entanto, a sua interface, bastante direcionada e restritiva, deixa pouco espaço para a personalização. Esta rigidez pode representar um obstáculo para as equipas que necessitam de controlo sobre a forma como as políticas e os controlos são estruturados ou para as que gerem *frameworks* híbridas.

# Capítulo 3

# Análise de Requisitos do Sistema

## 3.1 Levantamento de Requisitos

Para desenvolver um sistema mais eficaz e amplamente adotado, realizámos uma análise de necessidades com o objetivo de identificar os principais pontos problemáticos enfrentados pelos nossos utilizadores-alvo, bem como as funcionalidades essenciais para os resolver. Ao compreender as suas dificuldades e expectativas, procurámos conceber uma plataforma que não só respondesse aos seus requisitos, mas que também superasse as suas expectativas.

## 3.2 Requisitos Funcionais

### 3.2.1 Autenticação Segura e Controlo de Acessos

A autenticação dos utilizadores é realizada através do Keycloak, uma solução de gestão de identidades e acessos *open-source*, que fornece funcionalidades como autenticação centralizada, gestão de sessões, federação de identidades, e suporte a *Single Sign-On*. O Keycloak permite integrar múltiplas aplicações com um único ponto de autenticação, reduzindo a complexidade e aumentando a segurança da gestão de acessos. Suporta diversos protocolos padrão da indústria como OpenID Connect, OAuth 2.0 e SAML 2.0, o que facilita a sua integração com sistemas externos e aplicações modernas. No contexto da nossa plataforma, o Keycloak é inicializado automaticamente através de um container Docker aquando do arranque do sistema, garantindo que a autenticação está sempre disponível e isolada do *backend* principal - desenvolvido em Django.

Através da sua interface administrativa, é possível configurar domínios de utilizadores, papéis (*roles*), grupos e políticas de acesso, embora essas configurações estejam implementadas no ficheiro das variáveis de ambiente. Os *tokens* de acesso gerados pelo Keycloak são posteriormente validados pela aplicação *backend*, que controla o acesso aos recursos de acordo com os privilégios definidos, sendo que têm um tempo de limite máximo de sessão associado ao utilizador.

A gestão de permissões é feita com base em funções RBAC, permitindo associar diferentes níveis de acesso consoante o perfil do utilizador - como por exemplo o Administrador, o Auditor interno ou o Contribuidor.

O sistema assegura que o acesso a documentos, relatórios, evidências e funcionalidades críticas está restringido consoante os privilégios atribuídos.

### 3.2.2 Gestão de Auditorias

O sistema permite a criação, edição, visualização e remoção de auditorias, tanto internas como externas, garantindo flexibilidade na organização dos processos de verificação.

Cada auditoria está associada a uma *framework* específica, incluindo a sua estrutura hierárquica de controlos e subcontrolos, bem como a políticas internas da organização.

A rastreabilidade é garantida através de um sistema de monitorização em tempo real, que permite acompanhar o estado e o progresso de cada auditoria.

### 3.2.3 Gestão de Evidências

A plataforma disponibiliza um repositório centralizado de evidências, permitindo o armazenamento seguro e organizado dos ficheiros submetidos. Os utilizadores podem carregar documentos em vários formatos, incluindo PNG, JPEG e PDF, assegurando flexibilidade na aceitação dos diferentes tipos de evidência.

Quando uma evidência não é aprovada nas auditorias internas, é possível submeter novas versões - mesmo por utilizadores distintos - com base nos apontamentos recebidos, promovendo a colaboração entre equipas.

Adicionalmente, o sistema integra um mecanismo de controlo de versões, que regista todas as alterações realizadas e mantém um histórico completo de cada evidência submetida.

### 3.2.4 Gestão da Conformidade e Relatórios

O sistema permite gerar relatórios automáticos com base na informação recolhida durante as auditorias e nas evidências submetidas, facilitando a comunicação dos resultados a entidades internas ou externas.

Estes relatórios incluem indicadores-chave de desempenho e dados estruturados para suportar a tomada de decisões.

Através de métricas, os utilizadores podem monitorizar o progresso de conformidade com diferentes *frameworks* e políticas.

Estão disponíveis filtros para segmentar a visualização da informação, permitindo uma análise precisa por auditoria.

## 3.3 Atores

Para compreender melhor a utilização do sistema, identificam-se os diferentes tipos de utilizadores (atores) que interagem com a plataforma. Estes atores representam perfis reais com responsabilidades específicas dentro do contexto de gestão de auditorias e conformidade.

### 3.3.1 Administrador

O Administrador é responsável pela configuração geral do sistema e pela gestão de papéis e permissões dos utilizadores. Tem acesso privilegiado à plataforma e pode definir standards de conformidade, gerir configurações técnicas e garantir que a estrutura de controlo de acessos está corretamente implementada. Este papel tem uma visão transversal de todas as funcionalidades da aplicação.

### 3.3.2 Dono da Empresa

O Dono da Empresa (*Company Owner*) é responsável pela gestão do espaço organizacional (*Tenant*) na plataforma. Tem autonomia para configurar o ambiente de trabalho da sua empresa, adicionar utilizadores, atribuir papéis e acompanhar o progresso da conformidade a nível global. Atua como principal responsável pela operação da organização dentro do sistema.

### 3.3.3 Contribuidor

O Contribuidor é o utilizador encarregado de submeter evidências relacionadas com auditorias e requisitos de conformidade. Pode atuar sobre diferentes controlos, subcontrolos ou políticas, anexar documentos em diversos formatos e acompanhar o estado das submissões. Este papel é essencial para a colaboração entre equipas e constitui o alicerce operacional da equipa, garantindo que a informação necessária está disponível para validação.

### 3.3.4 Auditor Interno

O Auditor Interno tem como responsabilidade rever as evidências submetidas e elaborar relatórios com base nessa análise. Atua como ligação entre os Contribuidores e os Auditores Externos, avaliando o grau de conformidade com as políticas e *frameworks* aplicáveis, classificando os resultados em três estados: *Major Non-Conformity*, *Minor Non-Conformity* e *Approved*. Pode também identificar não conformidades e solicitar evidências adicionais sempre que necessário.

### 3.3.5 Auditor Externo

O Auditor Externo conduz auditorias independentes após a aprovação interna de todos os controlos e subcontrolos associados a uma *framework* específica dentro de um projeto. O seu principal objetivo é validar os relatórios elaborados pelos Auditores Internos. Este perfil avalia a conformidade da organização com os standards definidos, analisa as evidências disponíveis e sinaliza eventuais inconsistências ou pontos críticos a corrigir.

## 3.4 Personas

Para garantir que a nossa plataforma responde de forma eficaz às necessidades reais dos utilizadores, definimos personas representativas com base nos principais papéis identificados na secção anterior (Atores).

Estas personas ajudam a compreender os objetivos, motivações, frustrações e necessidades específicas dos utilizadores finais, permitindo orientar o desenvolvimento do sistema para oferecer uma melhor experiência de utilização e uma maior adequação aos requisitos do negócio.

### 3.4.1 James Daniels – O Contribuidor

**Perfil:**
Idade: 32
Localização: Nova Iorque, EUA
Cargo: Analista de Conformidade de Segurança
Experiência: 5 anos em conformidade de cibersegurança
Características: Detalhista, organizado, prefere fluxos de trabalho estruturados

**Contexto:**
James trabalha numa empresa tecnológica de média dimensão e é responsável por submeter documentação de conformidade para *frameworks* como SOC 2 e ISO 27001. Interage com várias equipas para recolher evidências e garantir que os relatórios de conformidade estão atualizados.

**Objetivos e motivações:**
Pretende um local centralizado para acompanhar tarefas de conformidade.
Necessita de orientações claras sobre quais evidências submeter.
Valoriza lembretes automáticos sobre prazos de auditorias.

**Problemas:**
Acha cansativo recolher documentação de várias fontes.
Preocupa-se com a possibilidade de falhar atualizações críticas.
Tem dificuldades em acompanhar o progresso em múltiplas *frameworks*.

**Necessidades do EasyReg:**
Lista automática de tarefas de conformidade.
Modelos pré-definidos para submissão de documentos.
Ferramentas de colaboração com auditores.

### 3.4.2 Sarah Smith – A Dona da Empresa

**Perfil:**
Idade: 45
Localização: São Francisco, EUA
Cargo: CEO de uma *startup fintech* em crescimento
Experiência: 20 anos em gestão de negócios
Características: Estratégica, orientada para objetivos, prefere visões de alto nível

**Contexto:**
Sarah fundou uma *startup fintech* e é responsável por garantir que a empresa cumpre os standards de segurança e conformidade, de forma a manter a confiança dos clientes e estabelecer parcerias seguras. Apesar de não ser especialista técnica, precisa de acompanhar o estado da conformidade da sua empresa.

**Objetivos e motivações:**
Garantir que a empresa está preparada para auditorias sem necessidade de micro-gestão.
Ter acesso a um *dashboard* com indicadores-chave de conformidade.
Receber relatórios mensais que resumam os riscos de conformidade.

**Problemas**:
Considera a conformidade um tema complexo e difícil de gerir.
Não tem tempo para rever detalhes técnicos.
Preocupa-se com possíveis coimas ou sanções por incumprimento.

**Necessidades do EasyReg:**
*Dashboard* intuitivo com visão geral da conformidade.
Relatórios automáticos com análise de riscos.
Sistema de alertas para problemas de conformidade.

### 3.4.3 Thomas Johnson – O Auditor Interno

**Perfil:**
Idade: 38
Localização: Londres, Reino Unido
Cargo: Auditor Interno de IT
Experiência: 10 anos em avaliação de risco e conformidade
Características: Analítico, orientado por processos, valoriza documentação estruturada

**Contexto:**
Thomas é responsável por garantir que os processos de conformidade estão corretamente implementados antes da realização de auditorias externas. Trabalha com contribuidores e gestores para identificar lacunas e implementar ações corretivas.

**Objetivos e motivações**:
Acompanhar de forma eficiente o progresso da conformidade.
Ter listas de verificação claras por cada *framework*.
Utilizar uma fonte única e confiável para toda a documentação de auditoria.

**Problemas:**
Dificuldade em acompanhar múltiplas *frameworks* (SOC 2, ISO 27001, NIST, etc.).
Considera os processos manuais de auditoria demorados.
Precisa de uma forma organizada de documentar o esforço de conformidade.

**Necessidades do EasyReg:**
Sistema centralizado de acompanhamento de conformidade.
Recolha automática de evidências para reduzir trabalho manual.
*Frameworks* pré-configurados para facilitar a implementação.

### 3.4.4 Laura Rodrigues – A Auditora Externa

**Perfil:**
Idade: 42
Localização: Ovar, Portugal
Cargo: Consultora Sénior de Conformidade (Auditora Externa)
Experiência: 15 anos em cibersegurança e conformidade regulatória
Características: Objetiva, meticulosa, valoriza relatórios estruturados

**Contexto:**
Laura é auditora independente e trabalha com várias organizações, conduzindo avaliações oficiais de conformidade com base em *frameworks* regulatórios. Analisa evidências submetidas e elabora relatórios para certificações e reguladores.

**Objetivos e motivações:**
Aceder rapidamente à documentação pronta para auditoria.
Garantir controlo de acessos baseado em papéis para proteger dados sensíveis.
Receber relatórios normalizados para facilitar o trabalho de validação.

**Problemas:**
Recebe documentação inconsistente ou incompleta das empresas.
Tem dificuldade em gerir a conformidade quando há múltiplos *frameworks* em simultâneo.
Precisa de um repositório seguro e bem estruturado.

**Necessidades do EasyReg:**
Portal seguro para submissão de evidências.
Relatórios automáticos de conformidade.
Ferramenta de mapeamento entre *frameworks* com controlos sobrepostos.

### 3.4.5 John Evans – O Administrador do Sistema

**Perfil:**
Idade: 29
Localização: Toronto, Canadá
Cargo: Administrador de Segurança e IT
Experiência: 7 anos em administração de sistemas
Características: Técnico, metódico, prefere soluções automatizadas

**Contexto:**
John é responsável por configurar e manter o EasyReg dentro da organização. Gere os papéis dos utilizadores, as definições de segurança e as integrações com outras ferramentas da infraestrutura de IT, garantindo a operação segura da plataforma.

**Objetivos e motivações:**
Implementar um controlo de acesso baseado em papéis (RBAC) detalhado.
Ter um sistema automatizado de registo de ações (*audit logging*).
Integrar facilmente com ferramentas de segurança existentes (como SIEM ou IAM).

**Problemas:**
A gestão manual de utilizadores consome muito tempo.
Necessita de manter os logs de auditoria seguros e imutáveis.
Enfrenta desafios ao escalar o sistema conforme a empresa cresce.

**Necessidades do EasyReg:**
Sistema intuitivo de gestão de papéis e permissões.
Registos de auditoria automáticos e seguros.
API para integração com ferramentas externas de segurança.

## 3.5 Casos de Uso

Neste subcapítulo, foram criados casos de uso com foco no ator, baseados nas instruções do nosso orientador.

### 3.5.1 Contribuidor

O Contribuidor consegue aceder ao *Tenant* a que pertence e aos seus Projetos incluídos dentro da mesma. Dentro de cada projeto específico, consegue submeter evidências contendo um título, uma mensagem obrigatória, e um ficheiro ou uma hiperligação para um ficheiro caso seja necessário, e consegue também editar as evidências, removê-las ou visualizar o estado de auditoria de cada uma delas.

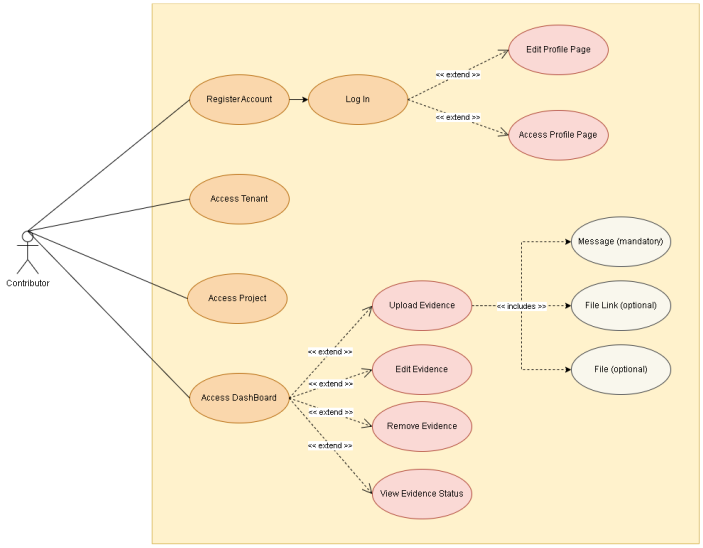


Figura 3.1: Casos de Uso do Contribuidor

### 3.5.2 Dono da Empresa

O Dono da Empresa tem as mesmas permissões que o Contribuidor para além de também conseguir criar *Tenants* e Projetos dentro da mesma. Consegue também gerir os utilizadores dentro do Tenant, incluindo os seus roles.


Figura 3.2: Casos de Uso do Dono da Empresa

### 3.5.3 Auditor Interno

O Auditor Interno após aceder ao *Tenant* a que pertence e a um Projeto em específico, consegue visualizar todas as evidências que ainda não têm estado definido e que foram submetidas para os controlos, subcontrolos ou políticas, conseguindo avaliá-las num dos três diferentes estados mencionados anteriormente - tal como consegue modificar o estado de uma auditoria interna que já esteja realizada.

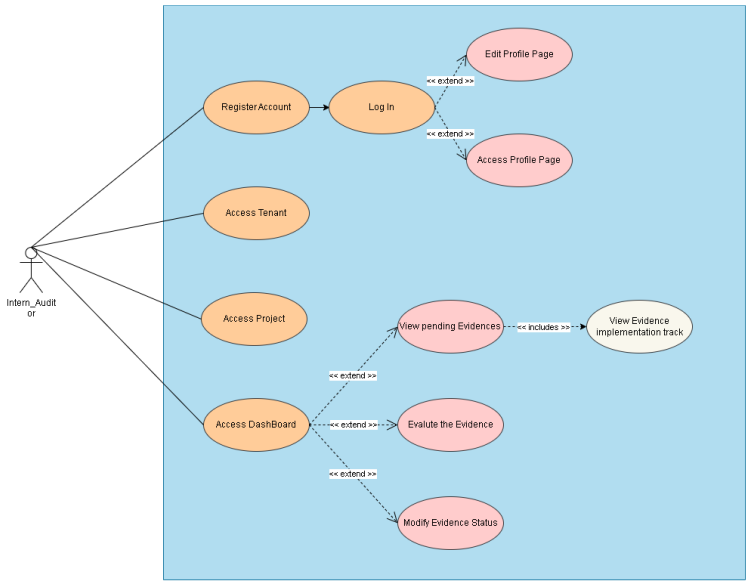


Figura 3.3: Casos de Uso do Auditor Interno

### 3.5.4 Auditor Externo

O Auditor Externo após aceder a um *Tenant* a que pertence e a um Projeto em específico consegue visualizar todas as auditorias internas que ainda não têm estado definido e consegue, após todos os controlos (e subcontrolos) estarem com uma auditoria interna aprovada, fazer uma auditoria externa sobre todas essas auditorias internas realizadas e aprovar ou reprovar a conformidade.


Figura 3.4: Casos de Uso do Auditor Externo

### 3.5.5 Administrador

O Administrador consegue gerir os utilizadores que pertencem ao sistema incluindo roles, informações relevantes e estados de perfil. Consegue também gerir os *Tenants* da mesma maneira que o Dono da Empresa – porém com um maior controlo - e consegue visualizar todo o progresso realizado até ao momento de cada Projeto de cada *Tenant*.

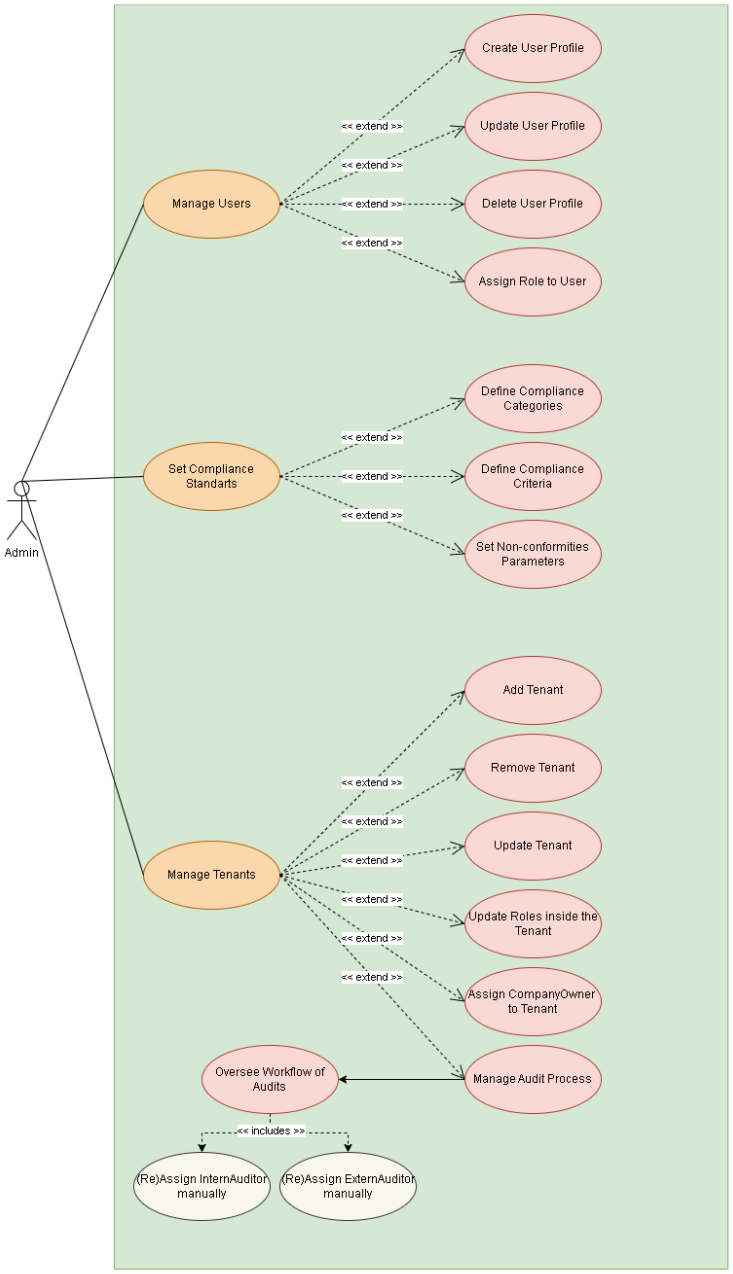


Figura 3.5: Casos de Uso do Administrador

## 3.6 Requisitos Não-Funcionais

Para além das funcionalidades essenciais que definem o comportamento direto do sistema, identificamos requisitos não funcionais que garantem a qualidade, fiabilidade, segurança e sustentabilidade da plataforma ao longo do tempo. Estes requisitos são fundamentais para assegurar uma experiência de utilização consistente, segura e eficiente, bem como para facilitar a sua evolução futura.

### 3.6.1 Adaptabilidade

A plataforma deve ser acessível através de um navegador web moderno e estar otimizada para dispositivos móveis. Isto garante que os utilizadores possam aceder ao sistema em diferentes contextos, incluindo em movimento, promovendo a flexibilidade e acessibilidade da solução.

### 3.6.2 Segurança

A plataforma web deve cumprir com os standards definidos pela OWASP, assegurando a proteção contra vulnerabilidades comuns, como *SQL Injection*, *Cross-Site Scripting (XSS)*, entre outras. A segurança é uma prioridade, especialmente considerando que o sistema lida com dados sensíveis relacionados com auditorias e conformidade.

### 3.6.3 Manutenibilidade

O sistema deve ser desenvolvido de forma modular e estruturada, facilitando futuras atualizações, correções de *bugs* e integração de novas funcionalidades. Esta característica reduz o tempo e os custos de manutenção, garantindo maior sustentabilidade da plataforma no longo prazo.

### 3.6.4 Usabilidade

A interface deve ser intuitiva, simples de utilizar e acessível a diferentes perfis de utilizadores, mesmo aqueles com pouca literacia digital. O objetivo é minimizar a curva de aprendizagem e maximizar a eficiência nas tarefas realizadas, promovendo a adoção natural da ferramenta por parte das equipas.

### 3.6.5 Armazenamento e Recuperação de Dados

Devem ser implementadas políticas de cópia de segurança (*backups*) regulares, permitindo a recuperação dos dados em caso de falhas, corrupção ou perda de informação. Este requisito é essencial para garantir a fiabilidade do sistema e a continuidade do negócio em cenários adversos.

# Capítulo 4

# Arquitetura do Sistema

## 4.1 Modelo de domínio

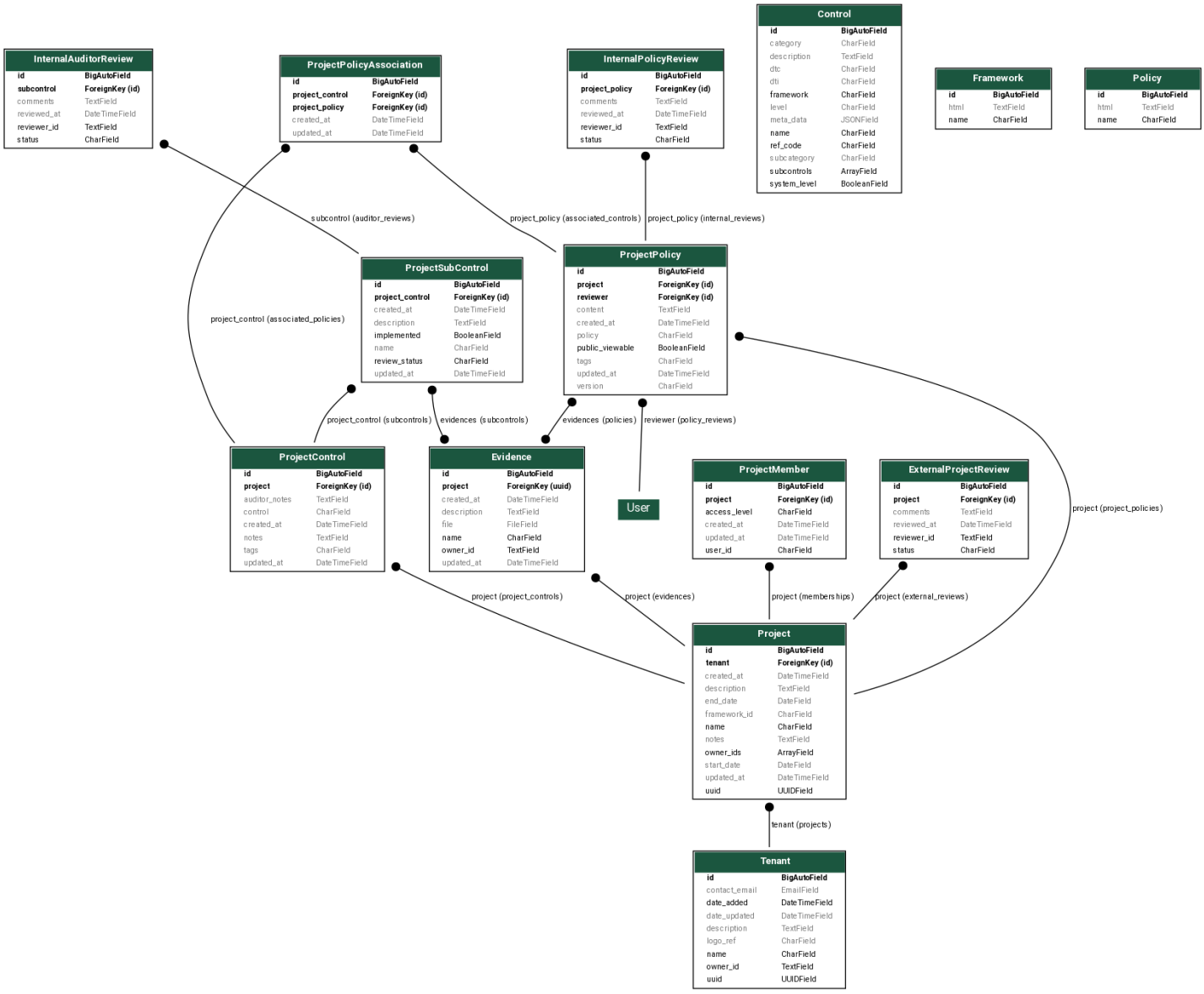


Figura 4.1: Modelo do Domínio

**Tenant**: Representa uma organização cliente. Cada *Tenant* pode conter múltiplos projetos. É associado a um owner_id (utilizador Keycloak) e possui dados como nome, contacto e descrição.

**Project:** Um projeto criado dentro de um Tenant. Contém metadados (nome, descrição, datas) e associações com *frameworks*, controlos, políticas, evidências e membros.

**ProjectControl / ProjectSubControl:** Representam os controlos e subcontrolos de segurança associados a um projeto específico. Cada controlo pode ter vários subcontrolos, notas, auditorias e evidências.

**ProjectPolicy:** Representa uma política de segurança associada a um projeto. Contém dados como visibilidade, *reviewer* e versão.

**Evidence:** Documentação submetida pelos utilizadores para justificar a conformidade com um controlo, subcontrolo ou política. Está ligada ao projeto e ao utilizador que a submeteu.

**ProjectPolicyAssociation:** Tabela intermédia que liga controlos e políticas dentro de um projeto, permitindo associar explicitamente um controlo a uma política.

**InternalAuditorReview / InternalPolicyReview:** Representam o processo de revisão interna. O auditor pode avaliar evidências associadas a subcontrolos ou políticas.

**ExternalProjectReview:** Regista a revisão feita por um auditor externo ao projeto, numa segunda camada de verificação.

**ProjectMember:** Liga utilizadores a projetos, definindo o seu *access_level* (Contribuidor, Auditor interno, Auditor externo).

**Framework / Control / Policy:** São entidades que representam as normas de segurança em si. Estas vêm de uma base de dados MongoDB e são lidas como referência para associar a projetos.

## 4.2 Arquitetura

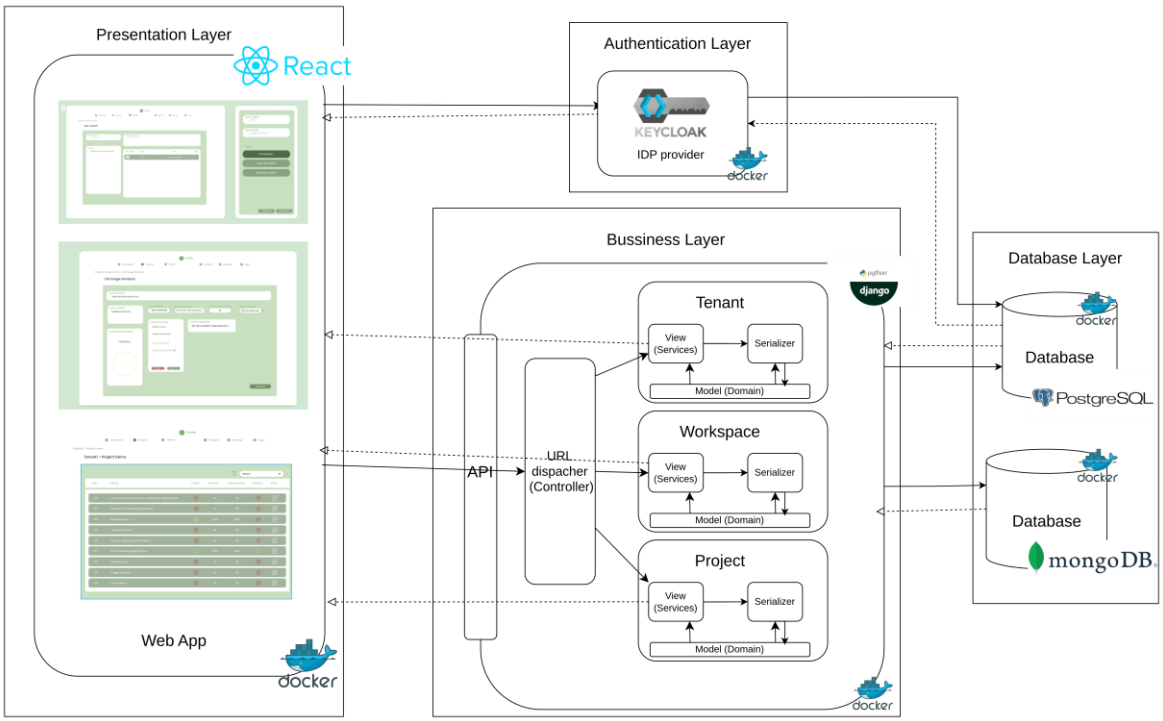


Figura 4.2: Diagrama da Arquitetura

A arquitetura da aplicação está organizada em quatro camadas principais, cada uma com responsabilidades distintas, assegurando modularidade, escalabilidade e separação de preocupações. Toda a aplicação está em *containers* Docker, garantindo portabilidade e consistência entre ambientes.

### 4.2.1 Camada de Apresentação

Esta camada corresponde à interface do utilizador, desenvolvida em React.js, e é executada num ambiente isolado via Docker. A aplicação web (*Web App*) permite que os utilizadores interajam com o sistema de forma intuitiva.
A comunicação com o *backend* é feita através de chamadas à API REST, sendo todos os pedidos autenticados e autorizados através da camada de autenticação.

### 4.2.2 Camada de Autenticação

A autenticação e a gestão de identidades são realizadas pelo Keycloak, um *Identity Provider* que permite *Single Sign-On,* gestão de utilizadores, grupos e permissões. Também se encontra num *container* Docker.

### 4.2.3 Camada de Negócios

Esta camada é o núcleo funcional da aplicação, implementada com o framework Django, e segue uma arquitetura MVC, organizada em três domínios principais:

**Tenant:** Responsável pela gestão dos *Tenants* (grupos no Keycloak), incluindo subgrupos (Projetos) e os seus membros.

**Workspace**: Gestão dos controlos de conformidade, *frameworks* e políticas associadas, com dados armazenados em MongoDB.

**Project**: Representa os projetos reais, contendo evidências, controlos aplicáveis, políticas e auditorias internas e externas.

Cada módulo tem os seguintes componentes:

**View (Services):** Implementa a lógica de negócio e responde aos pedidos da API.
**Serializer:** Trata da conversão de dados entre JSON e objetos Python (modelo).
**Model (Domain):** Representa a lógica de domínio e estrutura da base de dados.

A comunicação entre os módulos e o cliente (*frontend*) é feita através de um *dispatcher* de URLs (*Controller*/API Router), que organiza todos os *endpoints* REST expostos.

### 4.2.4 Camada de Dados

A camada de dados é composta por duas bases de dados, separadas consoante o tipo de informação:

**PostgreSQL:** Utilizado para dados estruturados, transacionais e relacionados com entidades do sistema como projetos, utilizadores e auditorias. Este é o repositório principal da camada de negócio.

**MongoDB:** Utilizado para armazenar dados semiestruturados, como *frameworks* de segurança, controlos e políticas, permitindo flexibilidade e desempenho na manipulação de documentos complexos.

Ambas as bases de dados estão também se encontram em *containers* Docker.

## 4.3 Documentação da API

### 4.3.1 Tenant:

Garante a gestão dos *Tenants* (organizações) na plataforma, incluindo a criação de grupos e a atribuição de utilizadores a cada *Tenant* através da integração com o Keycloak.

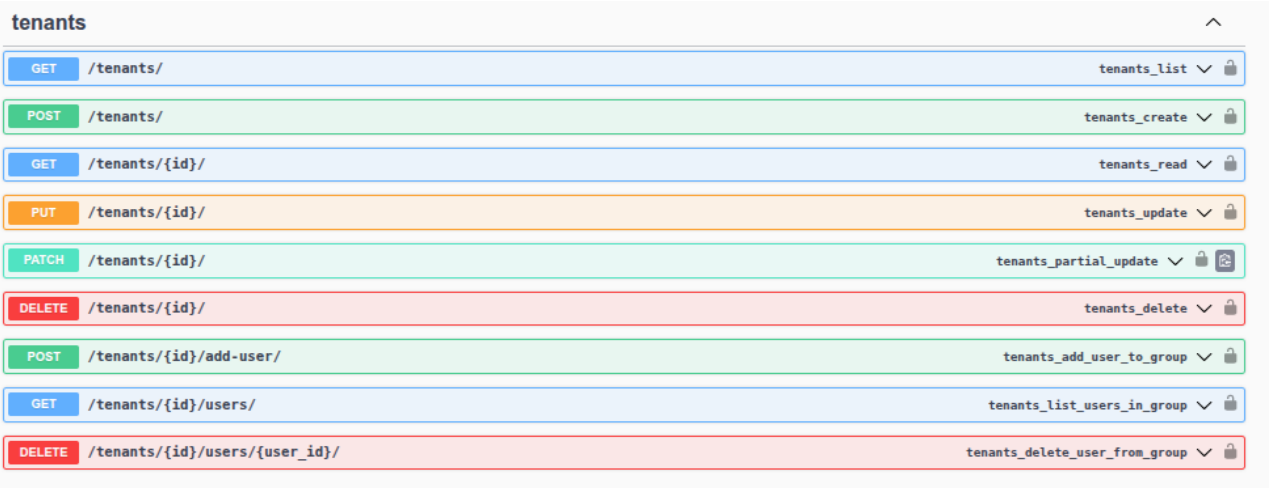


Figura 4.3: API dos *Tenants*

### 4.3.2 Projetos:

Permite criar e gerir projetos dentro de um *Tenant*, associando controlos, políticas e membros com diferentes papéis (Contribuidor, Auditor interno, Auditor externo).

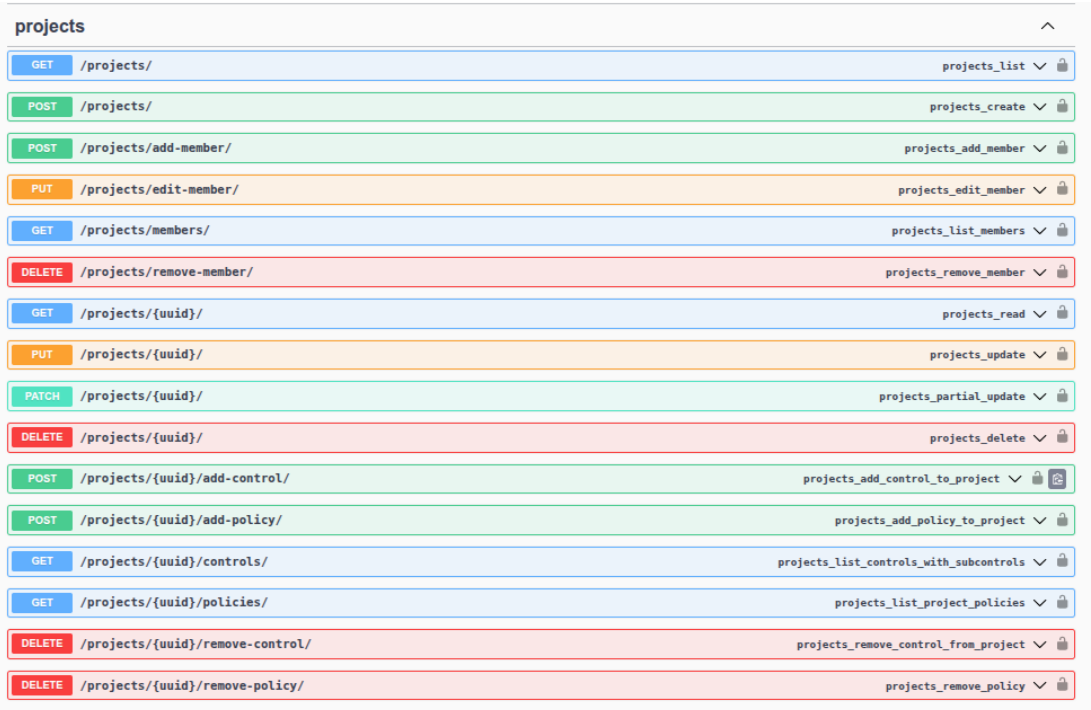


Figura 4.4: API dos Projetos

### 4.3.3 Evidências:

*Endpoints* responsáveis por criar, consultar e manter evidências de conformidade que suportam os controlos ou políticas associadas a cada projeto.

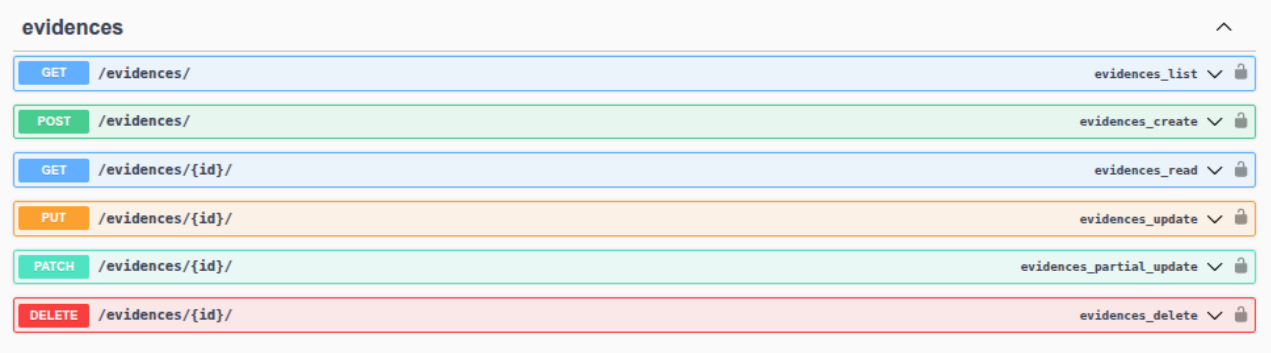


Figura 4.5: API das Evidências

### 4.3.4 Evidências nos Projetos:

Facilita a associação de evidências específicas a controlos, subcontrolos ou políticas dentro de um projeto, assegurando rastreabilidade e organização documental.

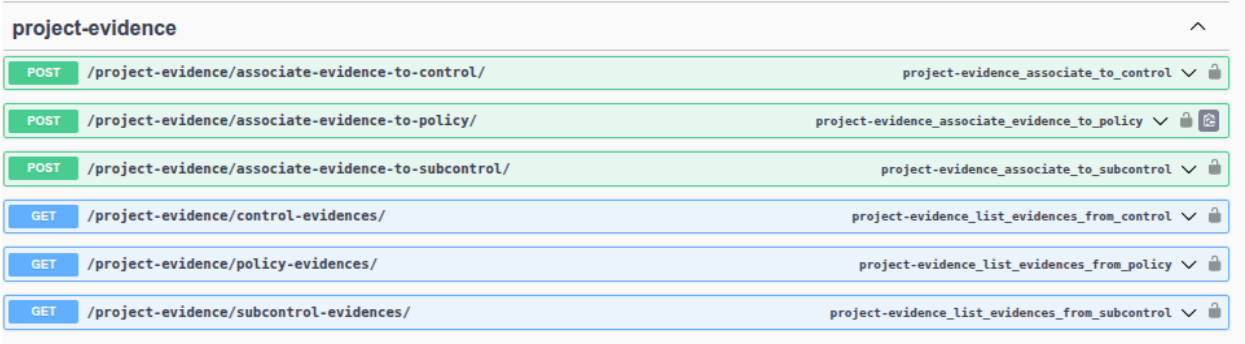


Figura 4.6: API de Evidências nos projetos

### 4.3.5 Revisão do Auditor Interno no Projeto:

#### 4.3.5.1 *Frameworks*:

Permite que o auditor interno avalie os controlos e subcontrolos de um projeto, registando progresso e histórico de revisões com base nas evidências existentes.

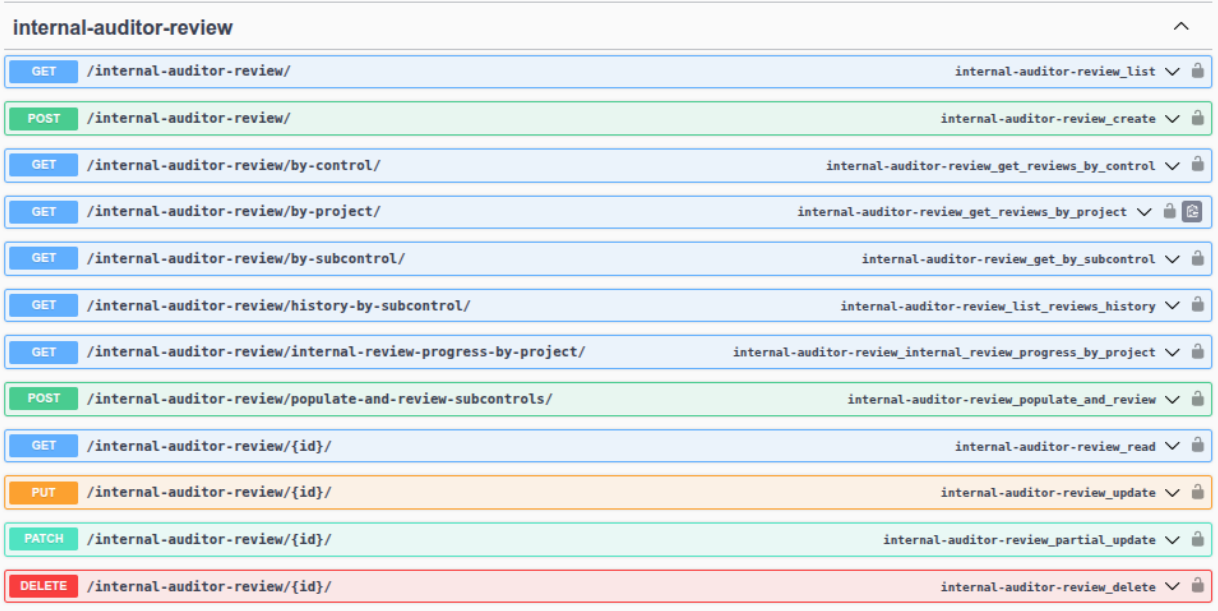


Figura 4.7: API para a revisão do Auditor Interno para as *frameworks*

#### 4.3.5.2 Políticas:

Dá suporte à revisão e validação de conformidade com políticas associadas ao projeto, garantindo que todas as políticas aplicáveis são devidamente analisadas.

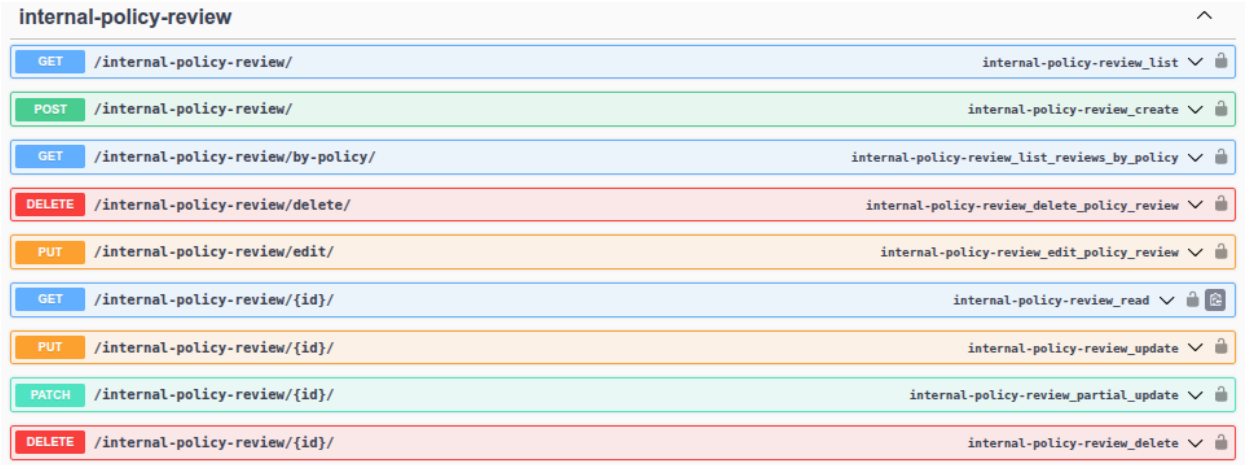


Figura 4.8: Revisão do Auditor Interno para as políticas

### 4.3.6 Revisão do Auditor Externo:

Dá acesso ao auditor externo para visualizar e avaliar o estado geral de conformidade dos projetos após a validação interna, promovendo uma segunda camada de verificação.


Figura 4.9: Revisão do Auditor Externo

## 4.4 Deploy

O sistema foi instalado numa máquina virtual do IEETA, utilizando Docker Compose para gerir todos os serviços necessários à aplicação.

A infraestrutura é composta por vários serviços, todos definidos no ficheiro *docker-compose.yml*:

**Backend**: Aplicação Django que expõe a API e gere a lógica de negócio.

**Frontend**: Aplicação React responsável pela interface com o utilizador.

**Base de dados (PostgreSQL):** Armazena dados estruturados como projetos, utilizadores, evidências e auditorias.

**Base de dados (MongoDB):** Armazena informações mais flexíveis, como *frameworks*, controlos e políticas.

**Keycloak**: Sistema de autenticação e gestão de utilizadores.

**NGINX:** Proxy reverso que encaminha os pedidos para os serviços corretos.

Todos os serviços estão ligados entre si por redes internas definidas no Docker, e os dados são mantidos através de volumes persistentes.

# Capítulo 5

# Implementação

Este capítulo apresenta a implementação e as principais caraterísticas de cada camada da aplicação web, de acordo com os requisitos definidos nos capítulos anteriores. Além disso, são explicadas algumas das decisões que tomámos durante o seu processo de desenvolvimento, de modo a justificar as escolhas técnicas e funcionais realizadas.

## 5.1 Camada de Apresentação

### 5.1.1 Design da interface do utilizador

Sendo a camada com a qual o utilizador interage diretamente com a aplicação web, o seu design passou por vários processos, cada um com o objetivo de garantir uma interface intuitiva e de fácil utilização. Para cumprir estes requisitos, utilizámos os princípios aprendidos no curso de Interação Humano-Computador, focando-nos especificamente nos 10 princípios gerais para o design de interação de Jakob Nielsen [3].

### 5.1.2 Desenvolvimento e layout da interface do utilizador

Para o desenvolvimento da aplicação web, utilizámos a *framework* de React.js, Next.js. Com o objetivo de garantir a consistência visual e simplicidade na implementação de estilos, recorremos exclusivamente à *framework* de CSS Tailwind CSS, que oferecia todas as funcionalidades necessárias.

O layout da aplicação web foca-se nos separadores *Tenants* e *Projects*, a partir dos quais a navegação do utilizador é linear do princípio ao fim (de uma página inicial à página pretendida). A informação é apresentada, maioritariamente, através de tabelas, e a interação do utilizador é realizada por meio de botões que facilitam a navegação e a execução de tarefas.

### 5.1.3 Integração com a camada de Negócios

Para receber e carregar informações nas bases de dados, é necessária uma ligação à camada de negócios através da camada de apresentação. Para esse fim, utilizámos a biblioteca Axios, um cliente HTTP baseado em promessas compatível com Node.js e browsers. Esta biblioteca permite a realização de pedidos HTTP assíncronos à camada de negócios, facilitando o envio de dados, bem como a gestão eficiente das respetivas respostas.

## 5.2 Camada de Autenticação

A camada de autenticação é essencial para garantir um acesso seguro à aplicação web e a privacidade dos dados no âmbito deste projeto. Funciona como mecanismo de defesa, permitindo que apenas os utilizadores autorizados interajam com funcionalidades específicas da aplicação, garantindo a integridade e fiabilidade dos dados.

Para implementar esta funcionalidade, optámos por utilizar o Keycloak, uma vez que fornece funcionalidades robustas, incluindo autenticação de utilizadores, controlo de autorizações e suporte para SSO, que permite que os utilizadores se autentiquem uma vez e continuem a aceder, de forma segura, ao sistema sem necessidade de introduzir repetidamente as suas credenciais. O Keycloak disponibiliza também funcionalidades que podem ser ativadas pelo administrador, como a autenticação multifator, políticas de palavra-passe e registo de auditoria, que reforçam coletivamente a segurança da aplicação, reduzindo os riscos associados ao acesso não autorizado e à fraude de identidade.

## 5.3 Camada de Negócios

### 5.3.1 RESTful API

Para transportar informação da camada de apresentação para as camadas de *backend*, utilizámos a *framework* Django REST para integrar uma API RESTful. O modelo RESTful permitiu-nos utilizar operações CRUD, simplificando a gestão de recursos e melhorando a experiência de desenvolvimento da aplicação, bem como permitir a reutilização e a modularidade dos métodos utilizados para a comunicação.

A API é também utilizada para a nossa autenticação baseada em *tokens* (juntamente com o Keycloak), uma vez que permite a passagem de informação de forma segura.
Alguns exemplos das nossas operações CRUD e *endpoints* são demonstrados no capítulo anterior.

### 5.3.2 Carregamento de ficheiros estáticos

O projeto permite que sejam carregadas *frameworks*, controlos e políticas juntamente com as que já são fornecidas inicialmente. Para tal, o utilizador deve fornecer os ficheiros correspondentes (em formato JSON no caso dos controlos, ou em formato HTML para estruturas e políticas) e colocá-los na pasta apropriada do projeto (*about_frameworks*, *base_controls* ou *base_policies*).

Estes ficheiros são automaticamente carregados para a base de dados através da execução de um script Python no momento do arranque da aplicação.

## 5.4 Camada de dados

Uma vez que utilizámos essencialmente dois tipos de informação (um dinâmico de pequena dimensão e outro estático de grande dimensão), optámos por utilizar dois modelos de base de dados diferentes, cada um adequado às respetivas características dos dados.

O PostgreSQL, uma base de dados relacional, foi utilizado para armazenar os dados dinâmicos e transacionais que beneficiam das funcionalidades oferecidas pela linguagem SQL, como configurações de *Tenants*, metadados de *Projects* e referências do carregamento de ficheiros.

Por outro lado, a MongoDB, uma base de dados orientada a documentos, foi utilizada para armazenar os grandes conteúdos estáticos, como as *frameworks*, os controlos e subcontrolos e os textos das políticas. O armazenamento destes conteúdos como documentos simplifica as atualizações e permite matrizes aninhadas (subcontrolos dentro de controlos).

# Capítulo 6

# Gestão do Projeto

## 6.1 Roles da Equipa

A equipa de desenvolvimento do EasyReg foi composta por 5 membros, cada um com papéis bem definidos ao longo do projeto:

| | |
|---|---|
| Project Manager | Gonçalo Monteiro |
| Architect | Miguel Francisco |
| DevOps | Ana Loureiro, Filipe Oliveira, Miguel Pinto |
| Developer | Todos |

## 6.2 Calendário do projeto

O projeto foi dividido em fases semanais, com entregas iterativas e objetivos bem definidos para cada etapa:

| ID | Task | W01 | W02 | W03 | W04 | W05 | W06 | W07 | W08 | W09 | W10 | W11 | W12 | W13 | W14 | W15 | W16 | W17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.1 | Configure backlog | ✔ | | | | | | | | | | | | | | | | |
| 0.2 | State of the Art | ✔ | | | | | | | | | | | | | | | | |
| 0.3 | Develop microsite | ✔ | ✔ | | | | | | | | | | | | | | | |
| 0.4 | Draft Architecture | ✔ | ✔ | | | | | | | | | | | | | | | |
| 0.5 | Most important Mockups | ✔ | ✔ | | | | | | | | | | | | | | | |
| 0.6 | Define User Profiles | | ✔ | | | | | | | | | | | | | | | |
| 0.7 | Define Use Cases | | ✔ | | | | | | | | | | | | | | | |
| **M1: lifecycle objectives** | **Initial milestone achieved** | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 1.1 | Define Functional Requirements | | | ✔ | | | | | | | | | | | | | | |
| 1.2 | Define Non-Functional Requirements | | | ✔ | | | | | | | | | | | | | | |
| 1.3 | Finish Mockups | | | ✔ | ✔ | | | | | | | | | | | | | |
| 1.4 | Improve Architecture | | | | ✔ | | | | | | | | | | | | | |
| 1.5 | Define User Stories | | | | ✔ | | | | | | | | | | | | | |
| **M2: lifecycle architecture** | **Architecture milestone achieved** | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 2.1 | Setup containers | | | | | ✔ | ✔ | | | | | | | | | | | |
| 2.2 | Implement TDD | | | | | ✔ | ✔ | | | | | | | | | | | |
| 2.3 | Construct data domain | | | | | | ✔ | | | | | | | | | | | |
| 2.4 | Implement Keycloak | | | | | | ✔ | | | | | | | | | | | |
| 2.5 | Setup database models | | | | | | | ✔ | ✔ | ✔ | ✔ | | | | | | | |
| 2.6 | Frontend pages | | | | | | | ✔ | ✔ | ✔ | | | | | | | | |
| 2.7 | Define API endpoints | | | | | | | | ✔ | ✔ | | | | | | | | |
| 2.8 | Project DEMO | | | | | | | | | ✔ | | | | | | | | |
| **M3: prototype** | **Prototype milestone achieved** | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 3.1 | API Integration | | | | | | | | | | | ✔ | ✔ | ✔ | ✔ | ✔ | | |
| 3.2 | Database Completion | | | | | | | | | | | ✔ | ✔ | ✔ | ✔ | ✔ | | |
| 3.3 | Redoing project pages | | | | | | | | | | | | | | ✔ | | | |
| 3.4 | Role Specific Screens | | | | | | | | | | | | | | ✔ | | | |
| 3.5 | Documentation | | | | | | | | | | | | | | | ✔ | ✔ | |
| 3.6 | Technical Report | | | | | | | | | | | | | | | | ✔ | ✔ |
| **M4: final presentation** | **Final presentation milestone** | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |

Figura 6.1: Calendário do projeto

## 6.3 Plano de comunicação

A equipa utilizou diferentes canais para garantir uma comunicação eficaz:

**Discord -** Comunicação rápida e diária entre os membros e orientadores.

**GitHub** - Repositório de código e controlo de versões.

**WhatsApp** - Comunicação rápida e diária entre os membros.

**Reuniões Semanais** - Discussão de progresso, bloqueios e alinhamento de prioridades.

Este plano permitiu uma boa coordenação entre todos os membros e assegurou o cumprimento das tarefas dentro dos prazos estipulados.

## 6.4 Metodologia

A metodologia seguida no desenvolvimento do EasyReg foi inspirada em princípios da metodologia ágil, com foco na colaboração contínua, adaptação rápida e entregas incrementais.

Em vez de um modelo rígido como Scrum, optou-se por uma abordagem prática e flexível, ajustada ao contexto académico e às dinâmicas da equipa.

A coordenação do trabalho foi feita principalmente através do GitHub Projects, onde foram criadas tarefas, atribuídos responsáveis e acompanhados os progressos de cada componente do sistema. Este sistema permitiu manter uma visão clara das prioridades e distribuir o trabalho de forma equilibrada entre os membros.

A comunicação informal, mas constante entre os elementos da equipa, aliada ao uso de ferramentas como Whatsapp e Discord, permitiu tomar decisões em tempo útil e garantir o avanço contínuo do projeto. As funcionalidades foram implementadas por iteração, com revisões frequentes e integração contínua entre *backend*, *frontend* e autenticação.

# Capítulo 7

# Conclusão

A jornada de desenvolvimento da plataforma EasyReg revelou-se não só um exercício técnico, mas também uma resposta concreta a um problema recorrente no setor: a complexidade e fragmentação na gestão de conformidade. Ao longo deste projeto, procurámos construir mais do que um simples sistema - queríamos criar uma ferramenta útil, adaptável e, acima de tudo, alinhada com a forma como as equipas realmente trabalham.

Olhando para o que foi alcançado, destacam-se algumas conquistas que merecem ênfase. A arquitetura modular, implementada com recurso a *containers* Docker, ofereceu-nos a base para uma solução escalável e de fácil manutenção. Separar a informação transacional da documental, utilizando PostgreSQL e MongoDB respetivamente, não só melhorou a performance como tornou o sistema mais flexível perante diferentes tipos de dados.

Outro ponto crítico foi a gestão de utilizadores e permissões. A integração com o Keycloak permitiu uma abordagem sólida à autenticação e ao controlo de acessos, assegurando que cada tipo de utilizador - desde o Colaborador ao Auditor externo - tem exatamente as ferramentas e permissões que precisam. Isso traduziu-se numa experiência mais fluida e segura.

No campo da automatização, as melhorias foram evidentes. As auditorias, a submissão e revisão de evidências e a geração de relatórios passaram de tarefas demoradas e propensas a erros para processos bem estruturados e auditáveis. Isto, por si só, representa um avanço significativo face aos métodos manuais ainda comuns em muitas organizações.

Talvez uma das vitórias mais silenciosas, mas importantes, tenha sido a interface. Ao desenhá-la com base em princípios de usabilidade e acessibilidade, conseguimos reduzir a barreira de entrada para utilizadores com diferentes níveis de literacia digital — algo que, frequentemente, é ignorado nas soluções técnicas.

Assim, com o EasyReg não pretendemos reinventar o mercado, mas sim juntar e melhorar as opções existentes numa só plataforma. As limitações que existem hoje são, em grande parte, oportunidades para melhorias futuras: integrações adicionais, suporte multilingue mais refinado ou mecanismos de análise preditiva. O essencial, no entanto, está lá - uma plataforma funcional, testada, e pronta para evoluir com as necessidades de quem a utiliza.

# Referências

[1] The Process of Security – Bruce Schneier
URL: https://www.schneier.com/essays/archives/2000/04/the_process_of_secur.html
Accessed on June 15, 2025

[2] Gapps – Open-source security governance and compliance platform
URL: https://web-gapps.pages.dev
Accessed on June 17, 2025

[3] Jakob Nielsen's 10 general principles for interaction design.
URL: https://www.nngroup.com/articles/ten-usability-heuristics/
Accessed on June 17, 2025