

Declaração de Integridade

Declaro ter conduzido este trabalho académico com integridade.

Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração.

Portanto, o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim.

Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P.PORTO.

ISEP, Porto, 11 de junho de 2025

Dedicatória

Dedico este projeto a todos que me permitiram chegar a este capítulo da minha vida acadêmica.

Resumo

O resumo do relatório (que só deve ser escrito após o texto principal do relatório estar completo) é uma apresentação abreviada e precisa do trabalho, sem acrescento de interpretação ou crítica, escrita de forma impessoal, podendo ter, por exemplo, as seguintes três partes:

1. Um parágrafo inicial de introdução do contexto e do problema/objetivo do trabalho.
2. Resumo dos aspetos mais importantes do trabalho descrito no presente relatório, que por sua vez documenta abordagem adotada e sistematiza os aspetos relevantes do trabalho realizado durante o estágio. Deve mencionar tudo o que foi feito, por isso deve concentrar-se no que é realmente importante e ajudar o leitor a decidir se quer ou não consultar o restante do relatório.
3. Um parágrafo final com as conclusões do trabalho realizado.

Palavras-chave (Tema): Incluir 3 a 6 palavras/expressões chave que caracterizem o projeto do ponto de vista de tema/área de intervenção.

Palavras-chave (Tecnologias): Incluir 3 a 6 palavras/expressões chave que caracterizem o projeto do ponto de vista de tecnologias utilizadas.

(O Resumo só deve ocupar 1 página, cerca de 20 linhas.)

Palavras-chave: Keyword1, ..., Keyword6

Abstract

Here you put the abstract in the "other language": English, if the work is written in Portuguese; Portuguese, if the work is written in English.

Agradecimentos

Gostaria de iniciar esta secção do relatório expressando a minha profunda gratidão às pessoas que me acompanharam ao longo deste percurso e contribuíram para a conceção deste projeto.

Em primeiro lugar, um sincero agradecimento ao professor Paulo Proença, cujo papel como orientador foi fundamental. A sua disponibilidade para rever o relatório inúmeras vezes permitiu-me apresentar uma versão mais refinada e estruturada deste trabalho.

Agradeço também ao David Mota, supervisor dos estágios na Devscope, a quem tive o prazer de conhecer na edição de FallStack 2024. Foi ele quem me selecionou para este estágio e acolheu os meus interesses em áreas específicas da informática, que mais tarde se integrariam na minha proposta de estágio.

Um agradecimento especial ao meu buddy, André Reis, pela orientação técnica e apoio especializado ao longo do projeto.

A todos os estagiários da Devscope, expresso a minha gratidão por tornarem esta experiência mais acolhedora e enriquecedora.

Quero também agradecer aos meus colegas de universidade, Rita Barbosa, Ana Guterres e Afonso Santos, que sempre me incentivaram e apoiaram durante a licenciatura. A sua amizade e motivação foram essenciais para o meu desenvolvimento académico e profissional.

Um agradecimento à DGES, pelo apoio financeiro concedido através da bolsa de estudo durante os três anos da licenciatura, e à Câmara Municipal de Portimão, pelo apoio adicional durante dois anos. O contributo destas instituições foi crucial para que eu pudesse prosseguir os meus estudos.

Por fim, e com um carinho especial, quero agradecer aos meus pais, que, apesar dos desafios e dificuldades, sempre se esforçaram para que eu tivesse acesso ao ensino superior. Agradeço também ao meu namorado e à sua família, que têm sido um pilar fundamental de apoio para esta menina deslocada de casa.

Conteúdo

Agradecimentos	vii
Lista de Figuras	xiii
Lista de Tabelas	xv
Lista de Algoritmos	xv
Lista de Código	xv
Lista de Abreviações	xvii
1 Análise do Problema	1
1.1 Contexto	1
1.2 Domínio do problema	1
1.2.1 Visão Geral da Aplicação	4
1.3 Requisitos Funcionais e Não Funcionais	5
1.4 Requisitos Funcionais	5
1.4.1 UC-01 Visualização de KPI's	6
1.4.2 UC-02 Filtragem de Métricas	6
1.4.3 UC-03 Mapa de Materialidade	6
1.4.4 UC-04 Pontuação ESG	7
1.4.5 UC-05 Detalhes de Métricas	7
1.4.6 UC-06 Métricas Customizáveis	7
1.4.7 UC-07 Importe de Dados	7
1.5 Requisitos Não Funcionais	8
1.5.1 Usability (U)	8
1.5.2 Reliability (R)	9
1.5.3 Performance (P)	9
1.5.4 Supportability (S)	9
1.5.5 + (Outros)	9
2 Desenho da solução	11
2.1 Arquitetura do Sistema	11
2.1.1 <i>Clean Architecture</i>	11
2.1.2 Modelo C4	12
2.1.3 Modelo de Vistas 4+1	13
2.1.4 Vista de Processos	14
2.1.5 Vista de Cenários	18
2.1.6 Vista Lógica	19
2.1.7 Vista de Implementação	22

2.1.8	Vista Física	22
2.2	Alternativa Arquitetural: DDD com <i>Clean Architecture</i>	24
3	Implementação da Solução	25
3.1	Descrição da implementação	25
3.1.1	Tecnologias Usadas	25
	Tecnologias de Pré-desenvolvimento	25
	Tecnologias de Desenvolvimento	26
3.2	Funcionalidades	26
3.2.1	Home Page <i>Dashboard</i>	26
3.2.2	Página das Métricas	26
3.2.3	Mapa de Materialidade	26
3.2.4	Página dos Datasets	27
3.2.5	Página dos Objetivos	27
3.3	Testes	27
3.4	Avaliação da solução	27
4	Conclusões	29
4.1	Objetivos concretizados	29
4.2	Limitações e trabalho futuro	29
4.3	Apreciação final	29

Lista de Figuras

1.1	Modelo de Dominio	3
1.2	Mapa de Materialidade segundo o setor SASB da Devscope (<i>Software e serviços IT</i>)	4
2.1	Modelo Arquitural <i>Clean Architecture</i> (Martin2012)	12
2.2	Modelo Arquitural C4 (C4Model)	13
2.3	Modelo de Vistas "4+1"(Kruchten1995)	14
2.4	Vista de processos dos casos de uso de consulta/visualização e filtragem (Nível 1)	14
2.5	Vista de processos dos casos de uso de consulta/visualização e filtragem (Nível 2)	15
2.6	Vista de processos dos casos de uso de consulta (Nível 3)	15
2.7	Vista de processo do caso de uso de consulta com acesso ao armazenamento na nuvem (Nível 3)	16
2.8	Vista de processo do caso de uso de filtragem (Nível 3)	16
2.9	Vista de processo dos casos de uso de criação e importação (Nível 1)	17
2.10	Vista de processo dos casos de uso de criação e importação (Nível 2)	17
2.11	Vista de processo do caso de uso de criação de métricas (Nível 3)	18
2.12	Vista de processo dos casos de uso de importação de dados (Nível 3)	18
2.13	Vista de Cenários	19
2.14	Vista Lógica (Nível 1)	19
2.15	Vista Lógica (Nível 2)	20
2.16	Vista Lógica (Nível 3)	21
2.17	Vista de Implementação Nível 3	22
2.18	Diagrama de Nível 2 da vista física da solução	23
2.19	Diagrama de Nível 3 da vista lógica do <i>container Frontend</i> segundo uma arquitetura de camadas concêntricas	24

Lista de Tabelas

1.1	Glossário do domínio do problema	2
1.2	Lista de Casos de Uso	5
1.3	Categorias FURPS+	8

Lista de Abreviações

ESG	E nvironmental, S ocial and G overnance
SASB	S ustainability, A ccounting S tandards B oard
GRI	G lobal, R eporting I nitiative

Capítulo 1

Análise do Problema

Neste próximo capítulo aborda-se a análise do problema em questão. Inicia-se com a descrição do domínio do problema, através do glossário de termos usados, do modelo de domínio e do mapa de materialidade da Devscope de acordo com as métricas indicadas pela Sustainability Accounting Standards Board (SASB) para o setor de *software* e serviços IT. Por fim, serão delineados os requisitos funcionais e não funcionais do sistema.

1.1 Contexto

A fase de análise é uma das mais importantes no ciclo de vida do desenvolvimento de software, pois é nela que ocorre a recolha dos requisitos, funcionalidades e necessidades do cliente, bem como das especificações do sistema. O objetivo final é definir de forma clara os recursos e funcionalidades que compõem a solução a ser desenvolvida.

Qualquer informação incorreta ou não obtida pode comprometer o desenvolvimento do produto, tanto em termos de qualidade como de cumprimento dos prazos.

A proposta de desenvolvimento de uma plataforma ESG surgiu da necessidade identificada pela Devscope em centralizar e otimizar o controlo de diversas métricas relacionadas com os pilares ambiental, social e de governança (ESG). Esta necessidade prende-se com a crescente importância de monitorizar e reportar práticas sustentáveis, promovendo uma gestão mais transparente, eficiente e alinhada com os objetivos de responsabilidade corporativa da empresa. Ao centralizar estes dados numa única plataforma, a Devscope pretende não só melhorar a sua capacidade de análise e tomada de decisões, como também reforçar o seu compromisso com a sustentabilidade e o impacto positivo na sociedade.

1.2 Domínio do problema

A presente secção visa descrever o domínio do problema através de vários artefactos de variadas complexidades, nomeadamente conceitos e diagramas.

A Tabela 1.1 apresenta os conceitos introduzidos no domínio do problema e que serão usados ao longo do desenvolvimento.

Conceito (EN)	Conceito (PT)	Descrição
ESG	ESG (Ambiente, Social, Governança)	Conjunto de critérios que avaliam o desempenho ambiental, social e de governança de uma organização, com foco na sustentabilidade e responsabilidade corporativa.
Dashboard	Painel	Resumo gráfico de várias informações importantes, normalmente utilizado para dar uma visão geral de um negócio.
Materiality Map	Mapa de Materialidade	Ferramenta que destaca as métricas ESG mais relevantes para cada setor, segundo as normas da SASB.
Metric	Métrica	Unidade de medida usada para quantificar aspectos específicos do desempenho ESG, como emissões de carbono, diversidade na força de trabalho ou políticas anticorrupção.
PoC	Prova de Conceito	Implementação inicial e simplificada de um sistema ou funcionalidade com o objetivo de validar uma ideia, conceito ou abordagem técnica.
Dataset	Conjunto de Dados	Coleção estruturada de dados ESG, normalmente em formato tabular, que contem informações como categorias, datas e valores associados.
URL	URL (Localizador Uniforme de Recursos)	Endereço que identifica e localiza um recurso na internet, como páginas <i>web</i> , APIs ou arquivos.
API	API (Interface de Programação de Aplicações)	Conjunto de regras que permite a comunicação entre diferentes sistemas ou componentes de <i>software</i> . Utilizada para integrar funcionalidades ou aceder a dados.
KPI	KPI (Indicador-Chave de Desempenho)	Métrica usada para avaliar o sucesso de uma organização ou de uma atividade específica na conquista de objetivos estratégicos ou operacionais.
SPA	SPA (Aplicação de Página Única)	Tipo de aplicação <i>web</i> que carrega uma única página HTML e atualiza dinamicamente o conteúdo conforme o utilizador interage, sem recarregar completamente a página.
AWS S3	AWS S3 (Amazon Simple Storage Service)	Serviço de armazenamento em nuvem da <i>Amazon</i> utilizado para guardar arquivos como documentos, imagens ou exportações de dados de forma segura e escalável.

Tabela 1.1: Glossário do domínio do problema

A Figura 1.1 ilustra o modelo de domínio e relaciona os diferentes conceitos da solução.

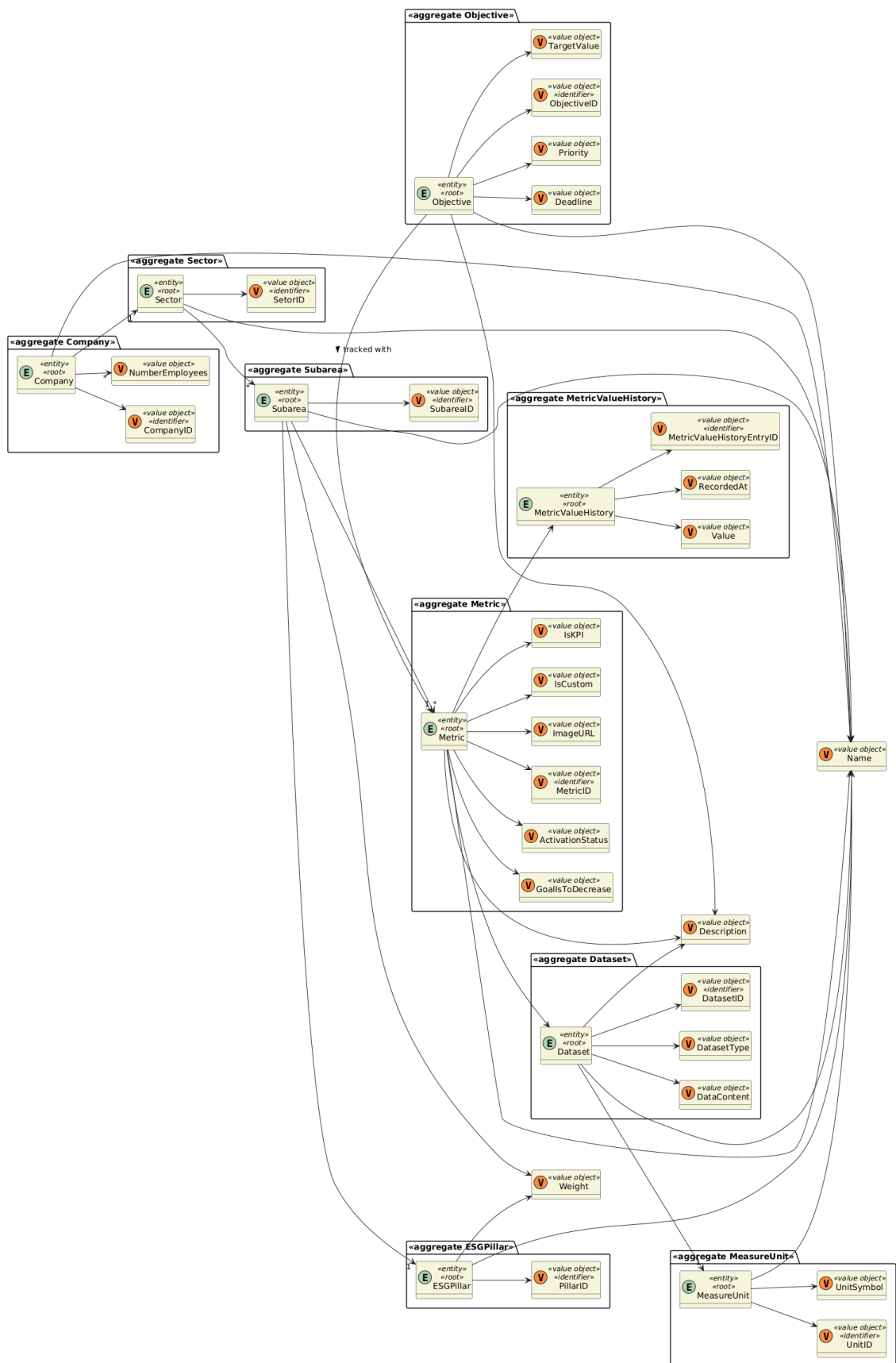


Figura 1.1: Modelo de Domínio

1.2.1 Visão Geral da Aplicação

O *dashboard* apresenta a pontuação ESG consolidada da empresa, oferecendo uma visão geral do seu desempenho em sustentabilidade. Além disso, destaca as métricas mais relevantes, assim como aquelas que apresentaram melhorias ou quedas em relação ao último registro.

Outra seção da plataforma disponibiliza uma visão detalhada de todas as métricas existentes: métricas customizáveis (criadas pelos próprios utilizadores) e métricas orientadas pelos padrões definidos pela SASB.

Cada métrica possui atributos específicos, incluindo o pilar e subárea ESG a que pertence, o seu código identificador, os conjunto de dados que lhe estão associados, a unidade de medida associada aos mesmos e a classificação do seu progresso.

De acordo com a SASB, adotado pela Devscope neste projeto, existem normas específicas para cada setor — como é o caso do setor de *Software* e Serviços de TI — que determinam quais as métricas devem ser reportadas (**SASBSector2025**). É com base nestas diretrizes que se constrói o Mapa de Materialidade, como indicado na Figura 1.2.

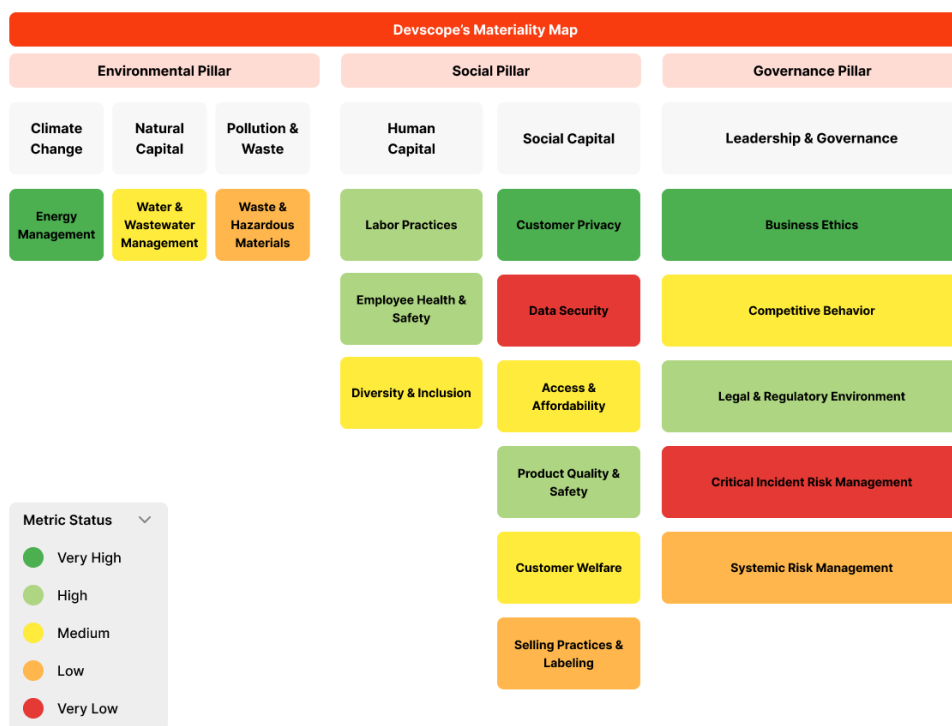


Figura 1.2: Mapa de Materialidade segundo o setor SASB da Devscope (*Software* e serviços IT)

Este mapa, um dos elementos centrais da plataforma, reflete as métricas ativas não só definidas pela SASB, consoante o setor da empresa, bem como as métricas personalizadas. Este mapa organiza as métricas segundo os pilares ESG e respectivas subáreas, e atribui uma cor correspondente ao seu progresso, proporcionando uma visão estruturada das prioridades de sustentabilidade da organização.

Por fim, outra página da plataforma é dedicada à gestão de objetivos, permitindo acompanhar os objetivos definidos, as métricas a eles associadas, o progresso na sua concretização ao saber se se encontram dentro ou fora do plano estabelecido.

1.3 Requisitos Funcionais e Não Funcionais

A seguinte seção compila os requisitos obtidos com o cliente e categoriza-os entre funcionais e não funcionais.

1.4 Requisitos Funcionais

Os requisitos funcionais correspondem a funcionalidades presentes no *software* a ser desenvolvido, normalmente representadas por casos de uso e acompanhados por diagramas. Os requisitos funcionais deste projeto estão organizados na Tabela 1.2, onde cada requisitos é considerado um caso de uso e identificado por um código alfa-numérico.

Código	Descrição Curta	Descrição
UC-01	Visualização de KPI's	Como utilizador da plataforma, quero visualizar rapidamente os principais KPIs ambientais, sociais e de governação, para poder avaliar o desempenho da organização.
UC-02	Filtragem de Métricas	Como utilizador da plataforma, quero poder filtrar os dados por pilar, de modo a facilitar a navegação.
UC-03	Mapa de Materialidade	Como utilizador da plataforma, quero visualizar a matriz de materialidade dividida por categorias ESG com um sistema de cores intuitivo, para identificar rapidamente os temas mais críticos e, ao clicar em cada indicador, aceder a detalhes explicativos que me ajudem a compreender o seu progresso e evolução.
UC-04	Pontuação ESG	Como utilizador da plataforma, quero aceder a uma pontuação ESG agregada, para ter uma visão geral do desempenho da empresa em responsabilidade social, ambiental e de governança.
UC-05	Detalhes de Métricas	Como utilizador da plataforma, quero clicar numa métrica e ver mais detalhes, para compreender a progressão de cada indicador.
UC-06	Métricas Customizáveis	Como utilizador da plataforma, quero poder criar métricas ESG específicas à realidade da empresa e definir a sua fonte de dados, para garantir que o painel reflete os indicadores mais relevantes para a nossa estratégia.
UC-07	Importe de Dados	Como utilizador da plataforma, quero poder importar dados para associar às métricas.

Tabela 1.2: Lista de Casos de Uso

1.4.1 UC-01 | Visualização de KPI's

Objetivo: Permitir ao utilizador consultar rapidamente os principais Key Performance Indicator (KPI) ESG da organização.

Ator Principal: Utilizador da plataforma

Descrição Geral: Este caso de uso disponibiliza uma visão geral de métricas dos três pilares (Ambiental, Social e de Governação). A interface mostra indicadores como emissões de CO₂, diversidade na força de trabalho, entre outros.

Fluxo Principal:

1. O utilizador acede ao painel principal da plataforma.
2. A aplicação apresenta os KPIs.
3. O utilizador pode visualizar valores atuais e tendências.

Resultado Esperado: O utilizador obtém uma perceção imediata do desempenho ESG da empresa.

1.4.2 UC-02 | Filtragem de Métricas

Objetivo: Permitir ao utilizador filtrar os dados por pilar ESG (*Environmental, Social* ou *Governance*).

Ator Principal: Utilizador da plataforma

Descrição Geral: Este caso de uso oferece um mecanismo de filtragem que facilita a navegação por métricas, de acordo com o pilar de interesse.

Fluxo Principal:

1. O utilizador seleciona o pilar desejado através de controlos de filtragem.
2. A plataforma atualiza os dados apresentados para refletir apenas as métricas relevantes.

Resultado Esperado: O utilizador consegue concentrar-se nos dados ESG mais relevantes ao seu objetivo de análise.

1.4.3 UC-03 | Mapa de Materialidade

Objetivo: Exibir uma matriz de materialidade com base no setor da empresa, permitindo a priorização de temas ESG.

Ator Principal: Utilizador da plataforma

Descrição Geral: A matriz destaca os temas ESG mais críticos segundo o modelo SASB, com categorização por pilar e subárea, e códigos de cor para facilitar a interpretação.

Fluxo Principal:

1. O utilizador acede à secção de materialidade.
2. É apresentada a matriz correspondente ao setor da empresa.
3. O utilizador pode clicar em indicadores para visualizar descrições detalhadas.

Resultado Esperado: Facilita a identificação de prioridades estratégicas ESG para o setor em análise.

1.4.4 UC-04 | Pontuação ESG

Objetivo: Fornecer uma pontuação ESG agregada para a empresa.

Ator Principal: Utilizador da plataforma

Descrição Geral: A aplicação calcula e exibe uma pontuação ESG com base nas métricas existentes e ativas, representada por cores e uma escalas numéricas.

Fluxo Principal:

1. O utilizador acede ao *dashboard* principal.
2. A plataforma mostra a pontuação global da empresa.
3. Podem existir dicas ou justificações para a pontuação.

Resultado Esperado: Oferece uma visão sintética do desempenho ESG.

1.4.5 UC-05 | Detalhes de Métricas

Objetivo: Permitir a exploração detalhada de cada métrica ESG.

Ator Principal: Utilizador da plataforma

Descrição Geral: O utilizador pode selecionar uma métrica para visualizar dados históricos, a classificação do seu progresso, o dataset de origem e o pilar e subárea a que pertence.

Fluxo Principal:

1. O utilizador clica numa métrica apresentada em *dashboards* ou gráficos.
2. A aplicação apresenta uma vista detalhada com gráficos temporais, dados e metadados.

Resultado Esperado: Maior compreensão do desempenho de cada indicador específico.

1.4.6 UC-06 | Métricas Customizáveis

Objetivo: Permitir ao utilizador definir novas métricas ESG alinhadas com a realidade da empresa.

Ator Principal: Utilizador da plataforma

Descrição Geral: Funcionalidade que permite criar métricas personalizadas, associando-as a fontes de dados e subáreas ESG específicas.

Fluxo Principal:

1. O utilizador acede à área de configuração de métricas.
2. Define os dados pedidos.
3. A métrica é validada e integrada nos *dashboards*.

Resultado Esperado: Flexibilidade para adaptar o sistema ESG a diferentes contextos empresariais.

1.4.7 UC-07 | Importe de Dados

Objetivo: Permitir ao utilizador importar datasets em formato CSV para análise ESG.

Ator Principal: Utilizador da plataforma

Descrição Geral: O utilizador pode carregar ficheiros de dados externos, que serão associados a métricas e utilizados em gráficos e cálculos.

Fluxo Principal:

1. O utilizador escolhe o ficheiro CSV a importar.
2. O sistema valida o conteúdo.
3. As informações passam a estar disponíveis para visualização e análise.

Resultado Esperado: Capacidade de integração de dados externos no ecossistema ESG da aplicação.

1.5 Requisitos Não Funcionais

Requisitos não funcionais correspondem a todas as propriedades do sistema que não se referem diretamente a funcionalidades específicas, mas sim a qualidades e restrições sobre como o sistema se deve comportar. Incluem aspetos como desempenho, usabilidade, confiabilidade, manutenibilidade e outros fatores de qualidade da aplicação.

Uma forma amplamente utilizada de categorizar estes requisitos é através do modelo **FURPS+**, detalhado na Tabela 1.3, que divide os requisitos não funcionais em cinco categorias principais, com um grupo adicional representado pelo símbolo +.

Letra	Categoria	Descrição
F	Functionality	Funcionalidades esperadas, segurança, interoperabilidade e adequação funcional
U	Usability	Facilidade de uso, aparência, acessibilidade, e interação com o utilizador
R	Reliability	Confiabilidade, tolerância a falhas, disponibilidade, e recuperação
P	Performance	Tempo de resposta, eficiência de recursos e escalabilidade
S	Supportability	Facilidade de manutenção, extensibilidade, modularidade, e testabilidade
+	Outros	Restrições de design, implementação, normas, tecnologias ou frameworks

Tabela 1.3: Categorias FURPS+

Os tópicos seguintes categorizam os requisitos não funcionais obtidos consoante o modelo FURPS+.

1.5.1 Usability (U)

- Interface clara, intuitiva e fácil de navegar.
- Feedback imediato ao utilizador através de alertas e *logs* indicando sucesso ou erro das ações realizadas.
- Aplicação disponível em inglês para melhor acessibilidade.

- Estilo visual coerente e consistente em todas as páginas e componentes.

1.5.2 Reliability (R)

- Tratamento robusto de erros, com mensagens claras sobre a origem dos problemas.
- Consistência e atualização correta dos dados apresentados ao utilizador.
- Implementação de testes unitários e de integração para garantir a confiabilidade das funcionalidades críticas.

1.5.3 Performance (P)

- As interações principais devem ser eficientes e não exceder 30 segundos por operação.
- O tempo total para utilizar a prova de conceito (incluindo *seeding* de dados e carregamento da interface) não deve ultrapassar 3 minutos.

1.5.4 Supportability (S)

- Código modular e bem estruturado, seguindo princípios de separação de responsabilidades.
- Facilidade em adicionar novas funcionalidades sem comprometer a usabilidade ou introduzir grandes alterações na base de código.
- Arquitetura baseada em boas práticas, como *Clean Architecture* e uso de camadas (*Controller, Service, Repository*).

1.5.5 + (Outros)

- A aplicação está totalmente em inglês.
- Estilo visual consistente e agradável.
- Seguir boas práticas de programação e arquitetura coesa.
- Utilizar fonte de dados simulada através de scripts de seeding.
- Conformidade com o padrão SASB para estruturação das métricas Environmental, Social, and Governance (ESG).
- Comunicação entre os diferentes módulos da aplicação através de REST APIs utilizando HTTP/S para garantir interoperabilidade e segurança.

Capítulo 2

Desenho da solução

Este capítulo aborda a fase de *design* no processo de desenvolvimento de *software*, onde, após a recolha de requisitos e análise das funcionalidades, são delineados os elementos do sistema através de diagramas que refletem a arquitetura e os modelos escolhidos, tendo em conta os requisitos e restrições definidos. Conforme descrito em **tsui2022essentials**, o *design* é dividido em duas etapas principais:

- **Design arquitetural** — uma visão macro e de alto Nível do sistema, onde são identificados os principais componentes, as suas propriedades externas e as relações entre eles. Esta fase é orientada pelos requisitos funcionais e não funcionais, bem como por considerações técnicas;
- **Design detalhado** — uma decomposição mais pormenorizada dos componentes, que detalha como cada módulo cumpre os requisitos funcionais definidos. Esta fase representa uma visão micro do sistema, guiada pela arquitetura estabelecida e pelos requisitos funcionais.

2.1 Arquitetura do Sistema

2.1.1 Clean Architecture

A *Clean Architecture* é um modelo arquitetural de *software* que privilegia a manutenibilidade e a escalabilidade do sistema, ao restringir as dependências entre os componentes de diferentes camadas. Neste modelo, diz-se que um componente A depende de um componente B se A conhece diretamente o nome de B, por exemplo, se o importa ou referencia. O objetivo da *Clean Architecture* é limitar estas dependências para que alterações em uma parte do sistema não causem efeitos colaterais noutras. Para isso, aplica-se o princípio da inversão de dependências: o componente A define uma interface, e o componente B implementa essa interface. Mesmo que A chame B em tempo de execução, B é quem depende de A, pois precisa de conhecer e implementar a interface definida por A (**Lano2023**).

Segundo **Lano2023**, este princípio está na base da chamada regra de dependência, que afirma que componentes dependentes de plataformas externas (como bases de dados, *frameworks* ou interfaces gráficas) podem depender da lógica de negócio, mas nunca o contrário. Assim, a lógica central do sistema permanece isolada e protegida de alterações tecnológicas periféricas, garantindo maior estabilidade e facilidade de evolução ao longo do tempo.

A Figura 2.1 ilustra o modelo *Clean Architecture* e as suas camadas concêntricas.

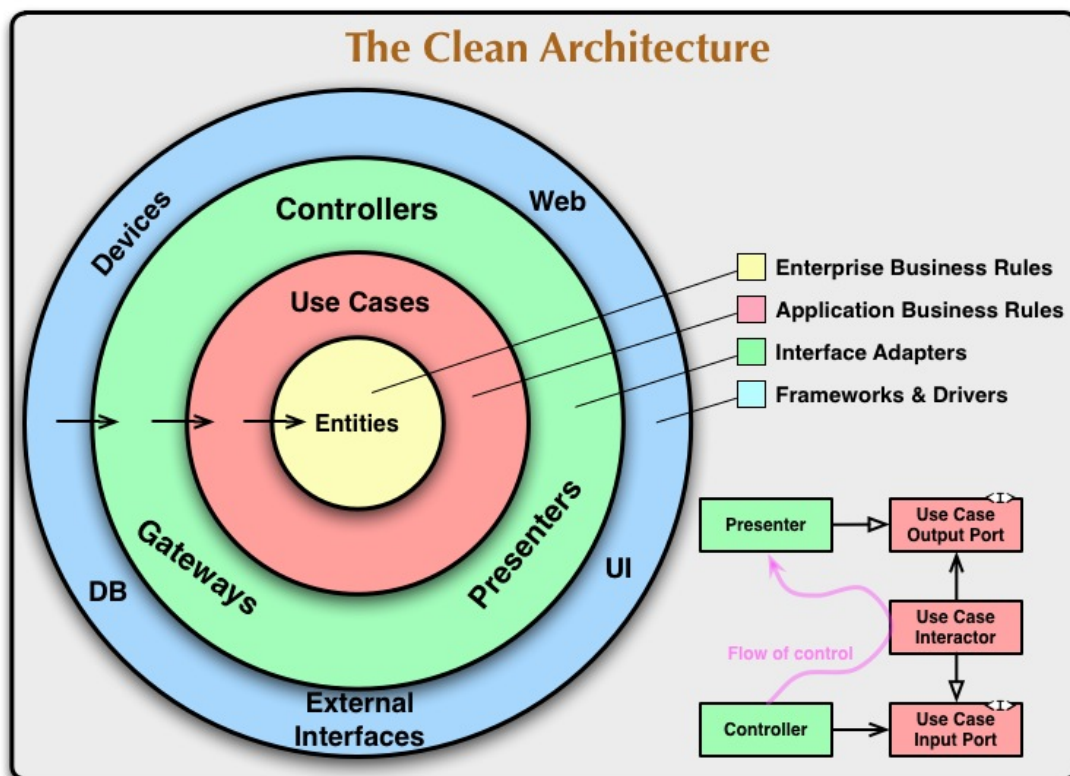


Figura 2.1: Modelo Arquitetural *Clean Architecture* (Martin2012)

Embora o projeto se inspire nos princípios da *Clean Architecture*, nomeadamente a separação clara entre as camadas de apresentação, lógica de negócio e acesso a dados, não foi realizada uma implementação rigorosa deste modelo (por exemplo, com injeção de dependências ou uso sistemático de interfaces). Esta decisão deve-se ao âmbito do projeto e à sua natureza prática, que não exigem este Nível de abstração.

Em alternativa à implementação deste modelo, foram aplicados os modelos C4 (Figura 2.1.2) e “4+1” (Figura 2.1.3) para representar graficamente a arquitetura da solução, proporcionando uma visão clara e compreensível da estrutura do sistema e das suas interações.

2.1.2 Modelo C4

O modelo C4 descreve a arquitetura de um *software* em quatro níveis de detalhe (*Context*, *containers*, *components* e *code*). De modo a criar os diferentes diagramas é necessário diferentes graus de abstração (Brown2018):

- **(Nível 1) System context diagram** - Dá uma vista geral do contexto em que o sistema se insere, isto é, quem usa o *software* criado e como é que outros sistemas inseridos no mesmo contexto comunicam com ele.
- **(Nível 2) Container diagram** - Detalha o sistema a um nível macro, decompondo-o em "*containers*", tais como aplicações (monolíticas ou microserviços), armazenamentos de dados, entre outros.

- **(Nível 3) Component diagram** - Decompõe um *container* em elementos que constituem abstrações reais no código a ser produzido.
- **(Nível 4) Code** - Nível de detalhe final, define a implementação de um componente.

A Figura 2.2 hierarquiza a granularidade pretendida no detalhe das diferentes visões do sistema numa representação com o modelo C4.

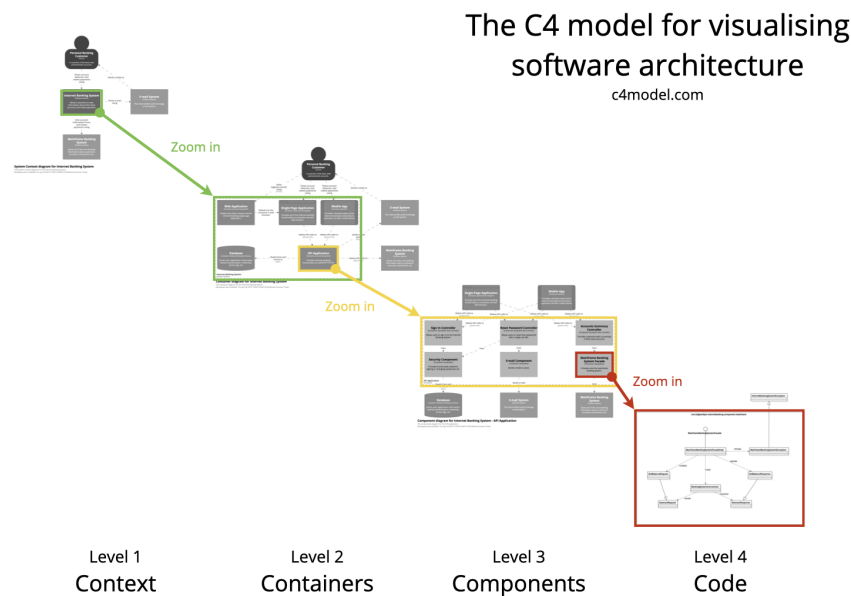


Figura 2.2: Modelo Arquitetural C4 (**C4Model**)

A granularidade do Nível 4 revela-se excessiva e retira o foco aos diagramas que melhor ilustram a arquitetura do sistema, portanto, não será abordado neste relatório.

2.1.3 Modelo de Vistas 4+1

Segundo **Kruchten1995**, o modelo arquitetural "4+1" descreve a arquitetura de um sistema de *software* a partir de cinco perspectivas complementares, cada uma focada em diferentes preocupações e públicos-alvo:

- **Vista Lógica** — Foca-se na funcionalidade do sistema, pois representa a estrutura dos principais componentes do domínio.
- **Vista de Processos** — Descreve os aspectos dinâmicos do sistema, como a comunicação entre processos concorrentes e a forma como as tarefas são sincronizadas e distribuídas.
- **Vista Física** — Mapeia os componentes de *software* para os recursos de *hardware* e ilustra como o sistema está distribuído.
- **Vista de Implementação/Desenvolvimento** — Mostra a estrutura estática do código, organização em módulos/pacotes e o ambiente de desenvolvimento.
- **Vista de Cenários** — Representa os principais casos de uso ou fluxos de interação do sistema.

A Figura 2.3 ilustra as diferentes vistas, o respetivo público-alvo para cada uma, e as suas interações.

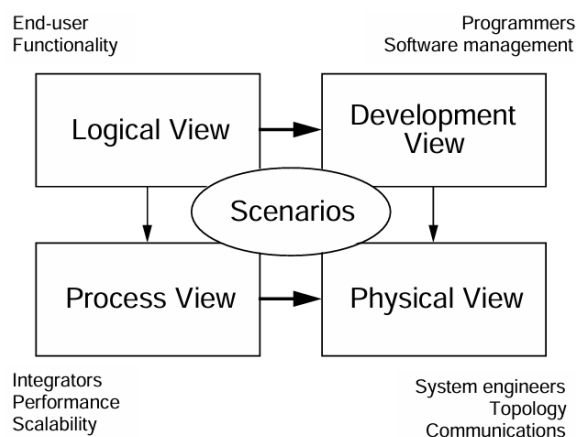


Figura 2.3: Modelo de Vistas "4+1"(Kruchten1995)

2.1.4 Vista de Processos

Alguns casos de uso partilham entre si a mesma estrutura de interação representada nos diagramas da vista de processos. Exemplos disso são os casos de uso UC-01, UC-02, UC-03, UC-04 e UC-05, que seguem a sequência ilustrada na Figura 2.4, variando apenas no conceito de negócio específico que cada um aborda.

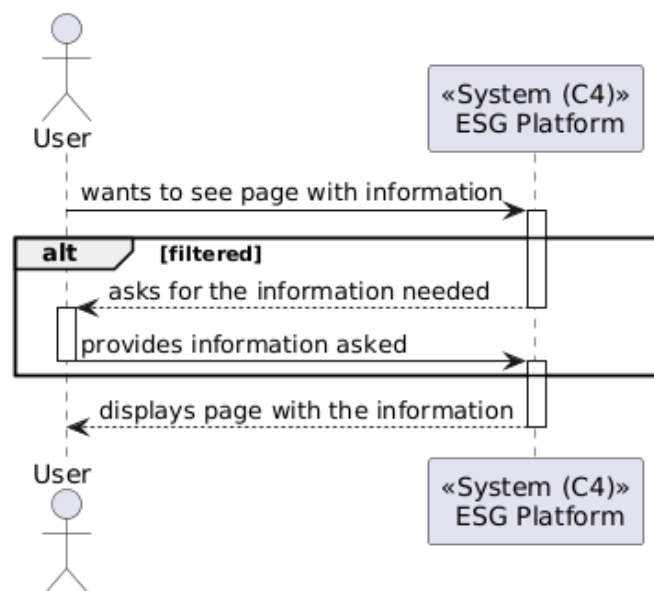


Figura 2.4: Vista de processos dos casos de uso de consulta/visualização e filtragem (Nível 1)

De igual forma, os respetivos diagramas de Nível 2 também partilham uma estrutura comum, diferenciando-se apenas na lógica de negócio subjacente. Esta semelhança é evidenciada na Figura 2.5, que exemplifica uma dessas variações mantendo a arquitetura processual de base.

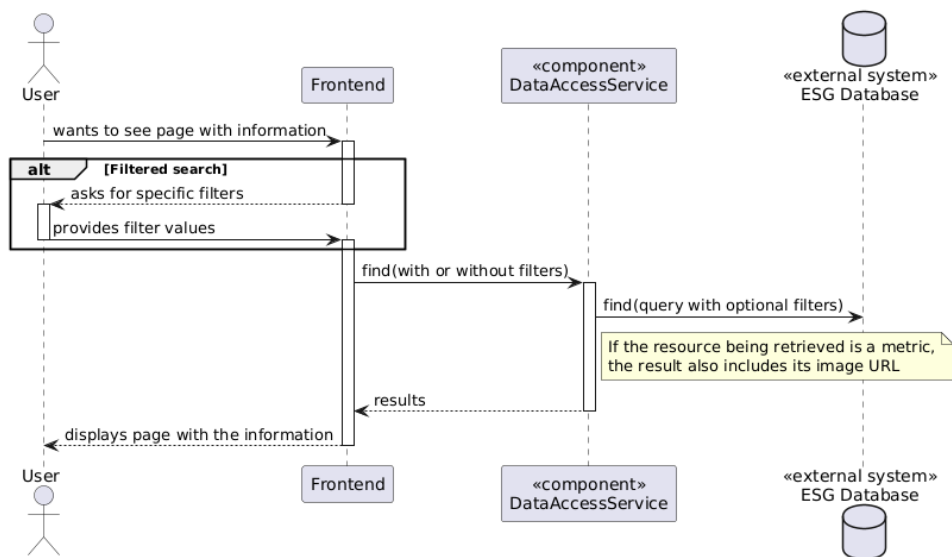


Figura 2.5: Vista de processos dos casos de uso de consulta/visualização e filtragem (Nível 2)

É no Nível 3 que se tornam evidentes as diferenças nos fluxos de interação, consoante os requisitos de cada funcionalidade. Como ilustrado na Figura 2.6, os casos de uso UC-01, UC-03 e UC-04 envolvem apenas o acesso à base de dados.

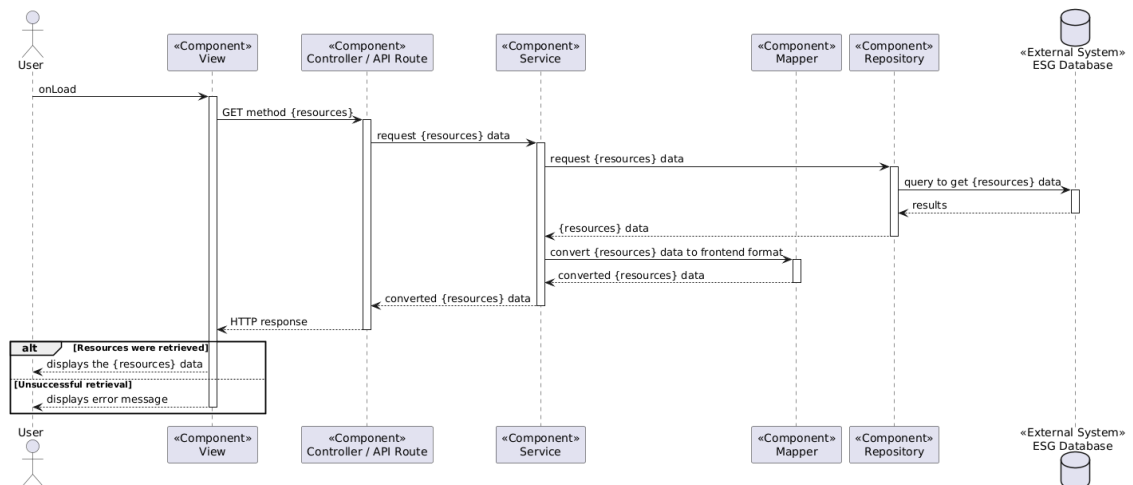


Figura 2.6: Vista de processos dos casos de uso de consulta (Nível 3)

Já o caso UC-05 (Figura 2.7) inclui também comunicação com o serviço de armazenamento na nuvem.

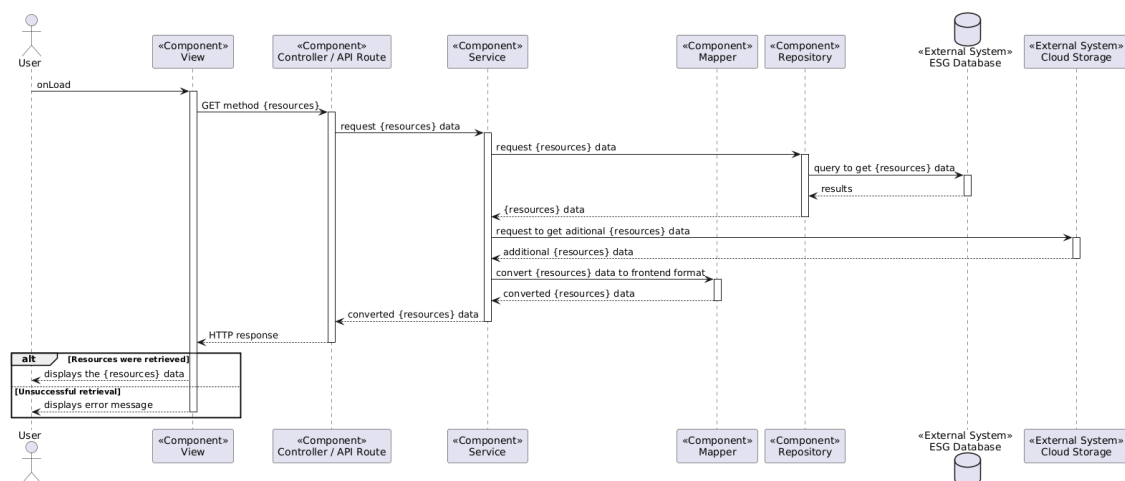


Figura 2.7: Vista de processo do caso de uso de consulta com acesso ao armazenamento na nuvem (Nível 3)

A UC-02 (Figura 2.8) distingue-se ainda por incluir uma etapa de filtragem adicional antes da obtenção dos dados.

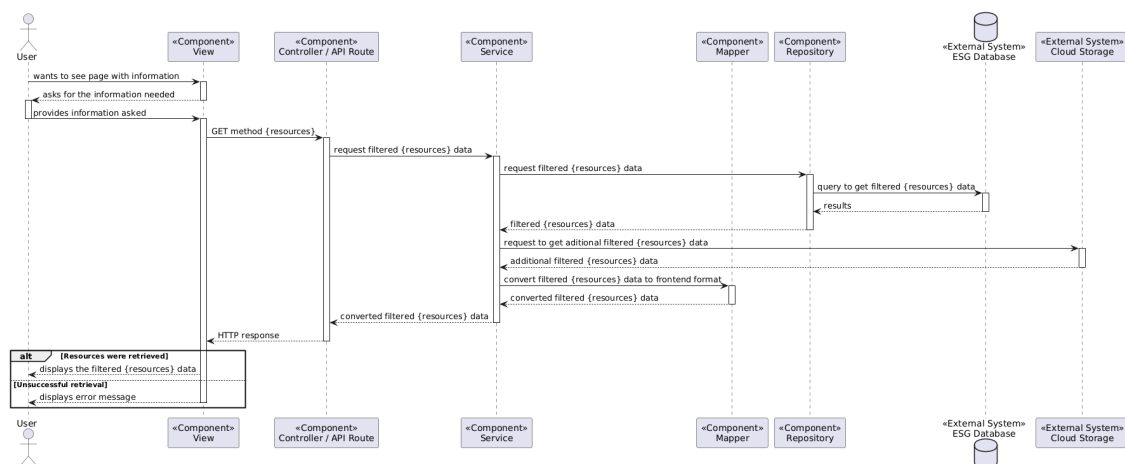


Figura 2.8: Vista de processo do caso de uso de filtragem (Nível 3)

Da mesma forma, no Nível 1, ambas os casos de uso UC-06 e UC-07 são bastante semelhantes, mudando apenas a lógica do negócio, como é ilustrado na Figura 2.9.

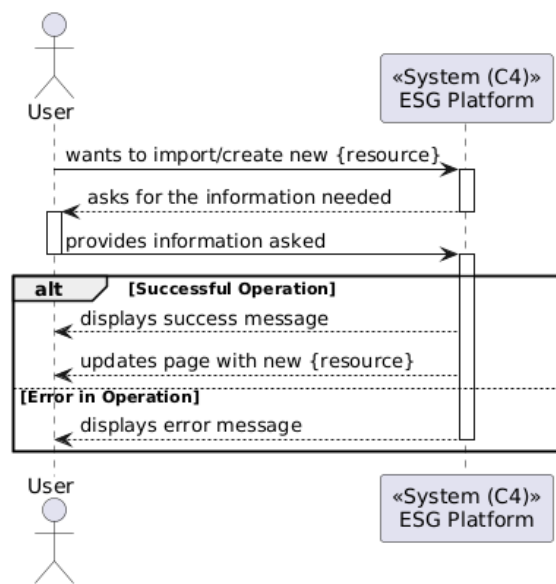


Figura 2.9: Vista de processo dos casos de uso de criação e importação (Nível 1)

A mesma tendência continua no nível seguinte, como é representado pela Figura 2.10.

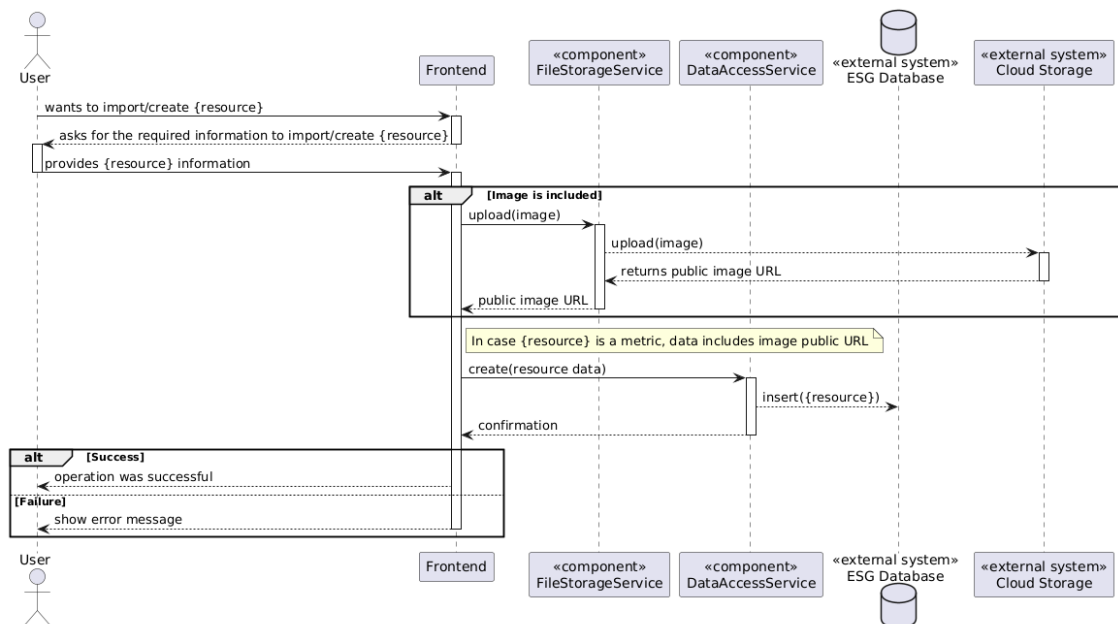


Figura 2.10: Vista de processo dos casos de uso de criação e importação (Nível 2)

A diferença no caso de uso UC-06 é realçada pelo diagrama de granularidade de Nível 3, como se pode ver na Figura 2.11.

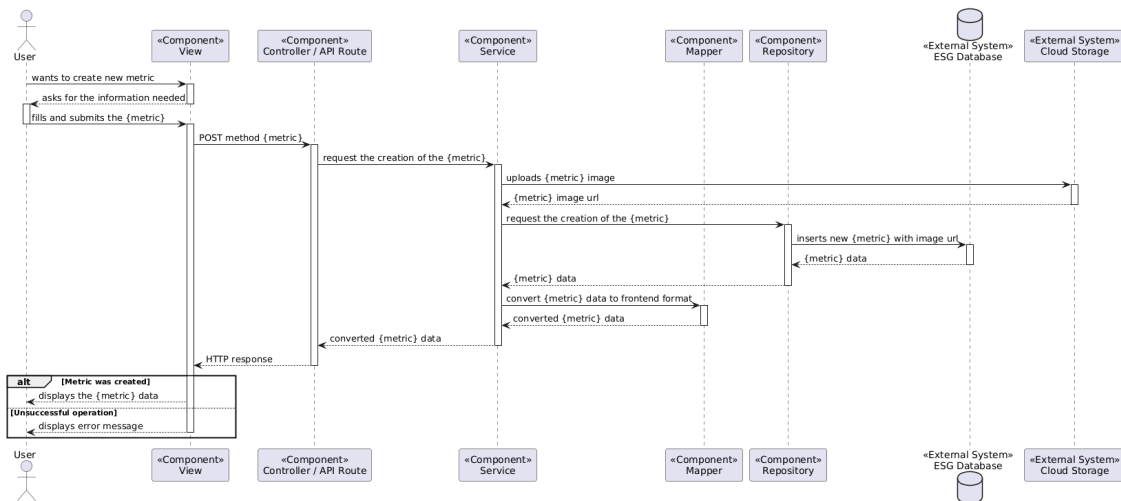


Figura 2.11: Vista de processo do caso de uso de criação de métricas (Nível 3)

De forma semelhante, o caso de uso UC-07 também apresenta essa distinção, evidenciada na Figura 2.12.

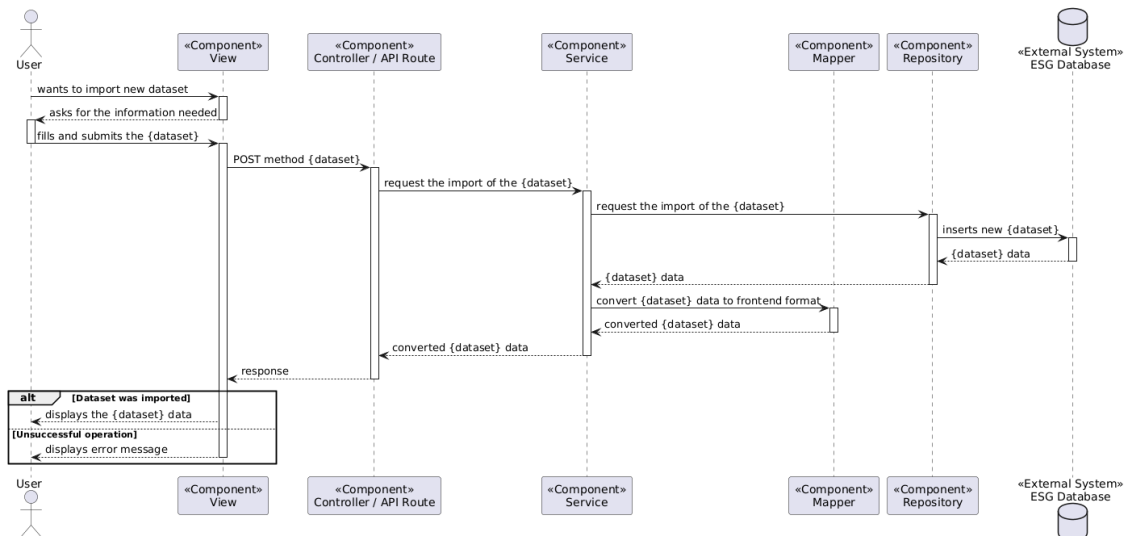


Figura 2.12: Vista de processo dos casos de uso de importação de dados (Nível 3)

2.1.5 Vista de Cenários

A vista de cenários, ilustrada pela Figura 2.13, compila os caso de usos principais. Esta vista é redundante, daí o "+1" na designação do modelo arquitetural (Kruchten1995).

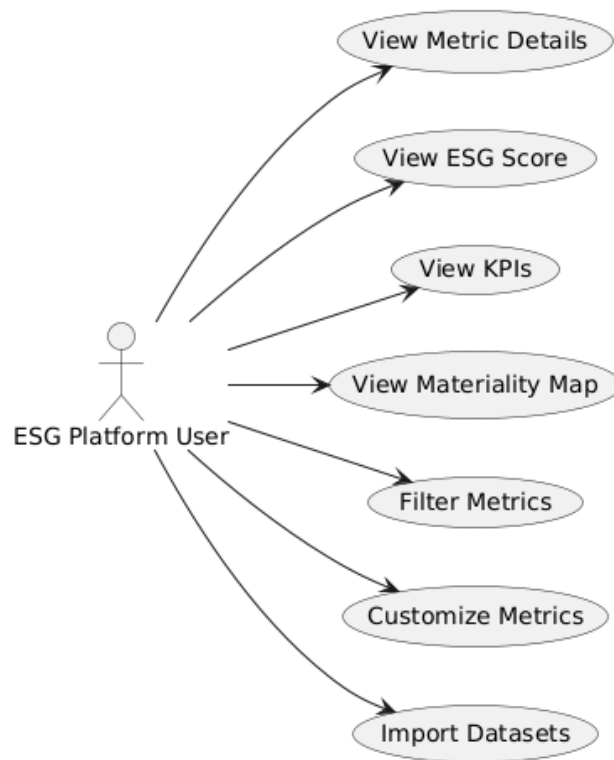


Figura 2.13: Vista de Cenários

2.1.6 Vista Lógica

Esta vista será abordada por vários diagramas correspondentes a sucessivos níveis de detalhe na representação lógica do sistema, iniciando-se pelo Nível 1, com a Figura 2.14, que permite entender o que o sistema disponibiliza, assim como que APIs consome.

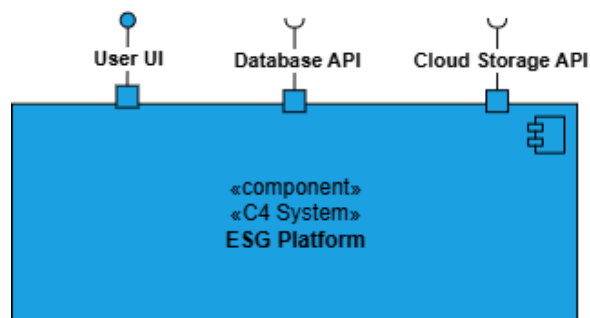


Figura 2.14: Vista Lógica (Nível 1)

A Figura 2.15 apresenta o Nível 2, oferecendo uma visão mais detalhada do sistema. Este nível inclui o contendor *Visualization*, responsável por disponibilizar a *User UI*, através da qual os utilizadores interagem com a aplicação. Este contendor consome duas APIs, a *Database API* e a *Cloud Storage API*, que, respetivamente, comunicam com a base de dados e com o serviço de armazenamento na nuvem.

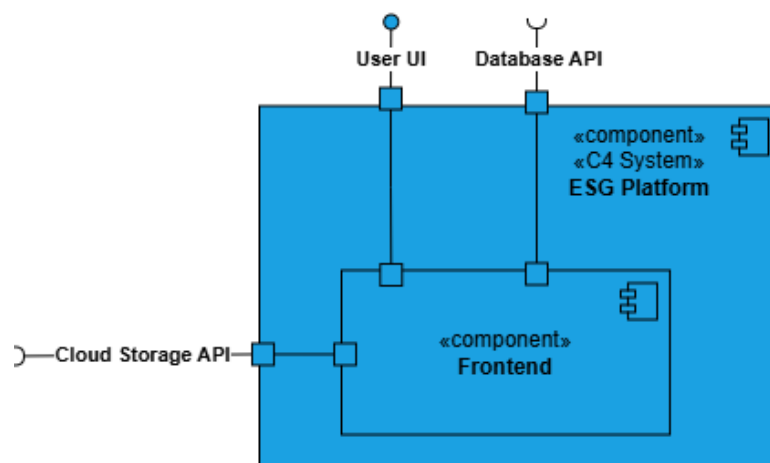


Figura 2.15: Vista Lógica (Nível 2)

Já o Nível 3 (Figura 2.16) representa os componentes que compõem os contentores do sistema, tornando mais claras as interações entre eles.

Observa-se que o contentor *Visualization* consome duas APIs: uma disponibilizada pelo *Database Provider* e outra pelo *Cloud Storage Provider*.

O *Cloud Storage Provider* é utilizado para armazenar ficheiros estáticos (como imagens) e gerar URLs de acesso público. Estes URLs são posteriormente armazenados na base de dados.

O *Database Provider*, por sua vez, utiliza uma base de dados relacional. A responsabilidade de persistir os dados, incluindo os URLs gerados, cabe ao contentor *Visualization*, que interage diretamente com ambas as APIs.

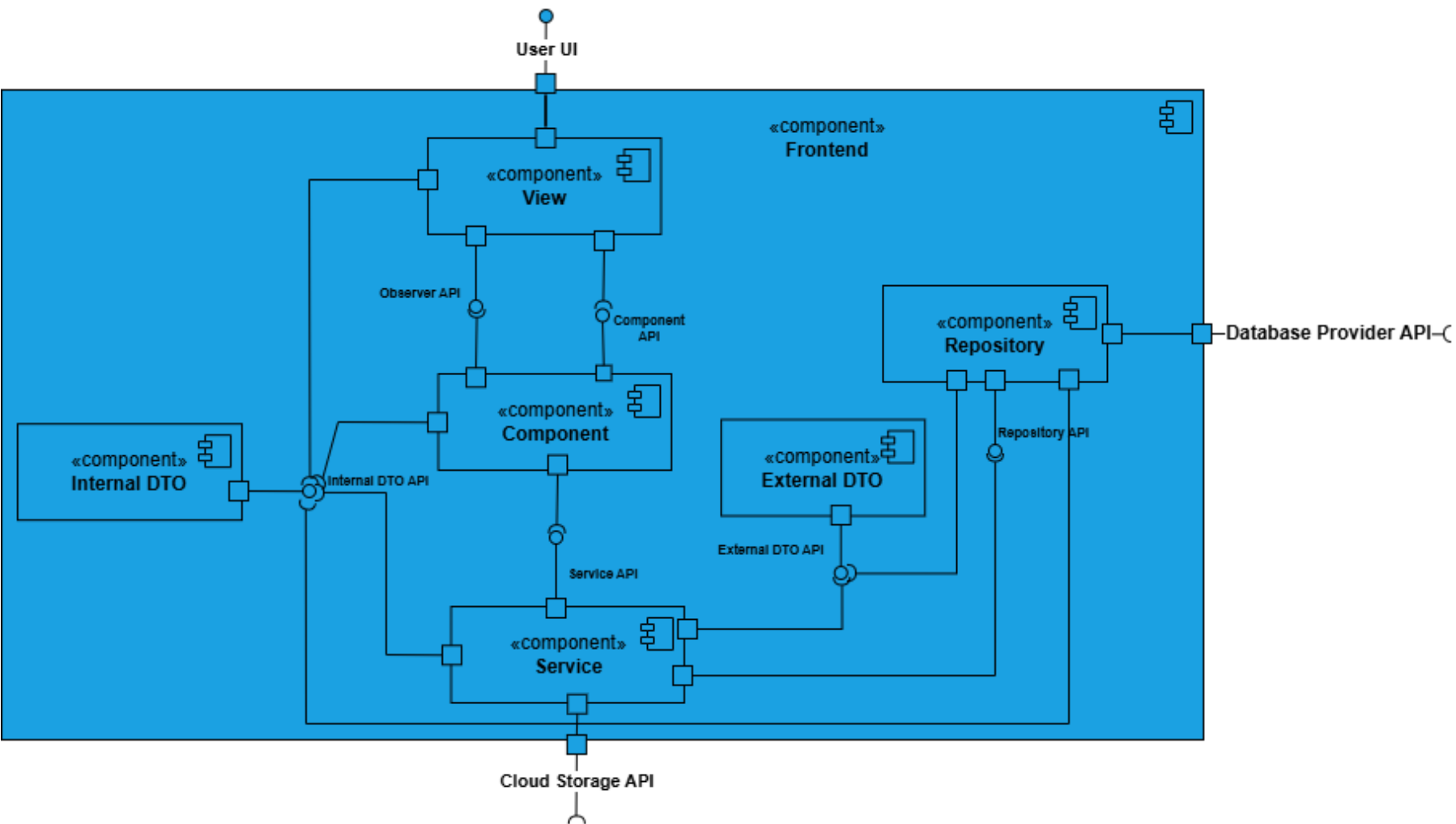


Figura 2.16: Vista Lógica (Nível 3)

2.1.7 Vista de Implementação

Semelhante à subseção anterior, esta vista será detalhada somente com um diagrama de Nível 3, representado pela Figura 2.17.

O Nível 1 não consta neste relatório por se tratar de uma representação demasiado abstrata, que pouco acrescentaria ao entendimento do sistema em questão. O Nível 2 não oferece nenhuma informação acrescida comparativamente ao diagrama do mesmo nível da vista lógica (Figura 2.15).

O Nível 3, de granularidade mais fina, elabora conteúdo do módulo *Frontend*, dando um maior entendimento não só na sua constituição mas também nas diferentes interações entre os seus elementos

A aplicação Plataforma ESG é constituída por uma React Single Page Application (SPA) para o *Frontend*, construída para suportar a geração atual de *browsers* para *desktop*.

Todas as ações realizadas pelos utilizadores têm origem no *Frontend*, que comunica com o *Database Provider* para obter ou inserir os dados necessários. No caso específico da criação de uma nova métrica, o *Frontend* interage adicionalmente com o *Cloud Storage Provider*, onde ocorre o *upload* da imagem representativa da respetiva métrica.

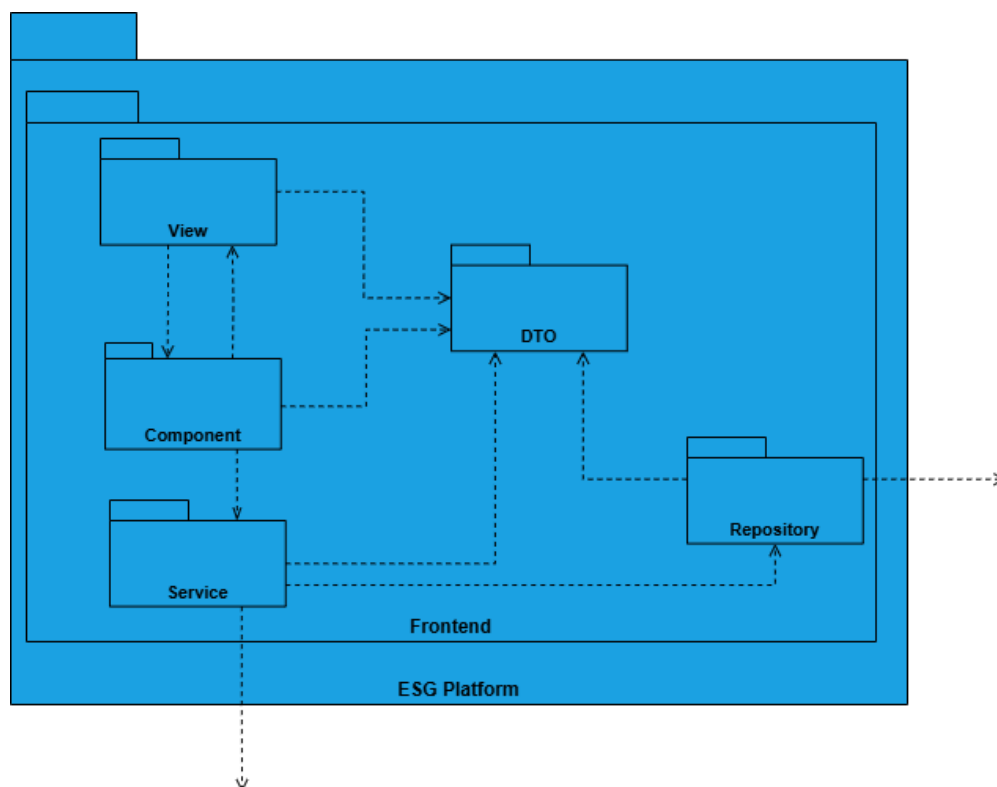


Figura 2.17: Vista de Implementação Nível 3

2.1.8 Vista Física

A vista física tem como objetivo mapear o *software* para o *hardware*, com ênfase nos requisitos não funcionais do sistema, como disponibilidade, confiabilidade (tolerância a falhas),

desempenho e escalabilidade (**Kruchten1995**). A Figura 2.18 apresenta o diagrama de Nível 2 da vista física da solução desenvolvida.

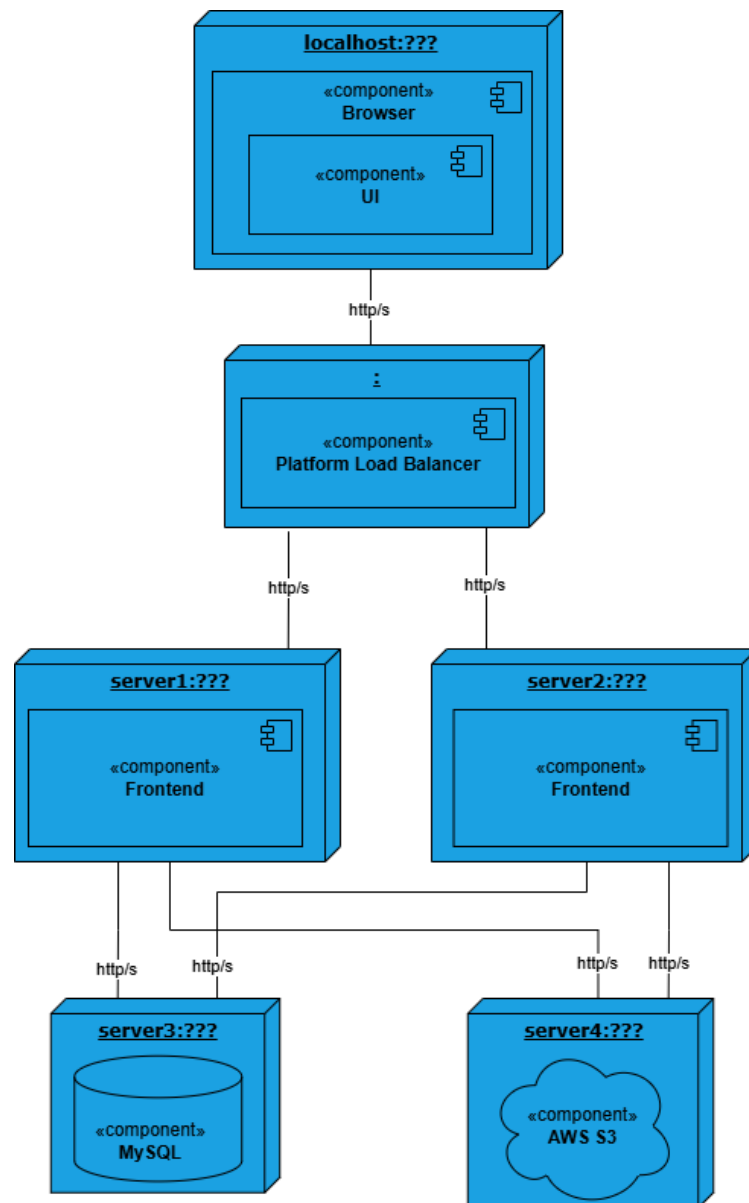


Figura 2.18: Diagrama de Nível 2 da vista física da solução

Os Load Balancers são responsáveis por distribuir o tráfego de forma dinâmica entre os servidores que compõem as camadas de *Visualization (frontend)* e *Data Management (backend)* (com base de dados). Estes atuam como intermediários entre os utilizadores e os servidores, assegurando uma distribuição equilibrada da carga, especialmente durante períodos de maior tráfego. Com isso, aumentam a disponibilidade e a resiliência do sistema: caso um servidor falhe, o tráfego é automaticamente redirecionado para outro servidor ativo (tolerância a falhas). Esta abordagem contribui ainda para a redução da latência e para evitar respostas inconsistentes da aplicação (**F52025**).

As comunicações entre os vários componentes são realizadas através de protocolo HTTP ou HTTPS.

2.2 Alternativa Arquitetural: DDD com Clean Architecture

Uma alternativa à arquitetura baseada nos modelos C4/4+1 seria o uso de *Domain-Driven Design* (DDD) combinado com os princípios da *Clean Architecture*. Esta abordagem também permite uma estrutura modular e escalável, alinhada aos limites naturais do domínio e preparada para futuras expansões.

Como mostra a Figura 2.19, o sistema é dividido em módulos independentes, cada um representando um conceito central da aplicação ESG: *Dataset*, *Metric*, *Objective*, *Pillar* e *Subarea*. Cada módulo encapsula as suas entidades, regras de negócio e casos de uso, comunicando-se com o exterior por meio de interfaces (*ports*), enquanto os adaptadores (como base de dados ou APIs) implementam essas interfaces.

A estrutura segue o modelo em anéis concêntricos da *Clean Architecture*, isolando o núcleo de domínio das dependências externas. Os casos de uso coordenam a lógica na camada de aplicação, e as camadas mais externas lidam com frameworks como *Prisma* ou *Next.js*.

Esta organização garante separação clara de responsabilidades, menor acoplamento, maior testabilidade e independência tecnológica — fatores que contribuem para um código mais limpo, coeso e sustentável.

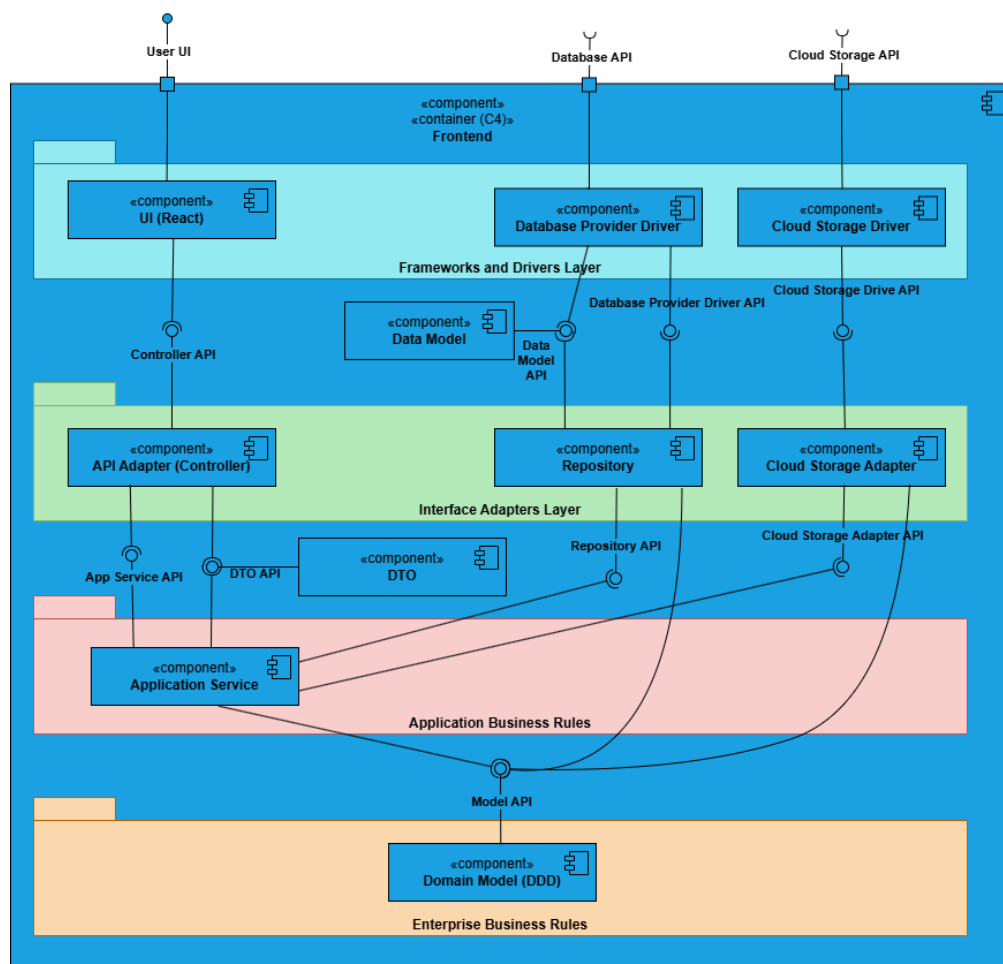


Figura 2.19: Diagrama de Nível 3 da vista lógica do *container Frontend* segundo uma arquitetura de camadas concêntricas

Capítulo 3

Implementação da Solução

Este capítulo deve ser dedicado à apresentação de detalhes relacionados com o enquadramento e implementação das soluções preconizadas no capítulo anterior. Note-se, no entanto, que detalhes desnecessários à compreensão do trabalho devem ser remetidos para anexos. Dependendo do volume, a avaliação do trabalho pode ser incluída neste capítulo ou pode constituir um capítulo separado.

3.1 Descrição da implementação

Descrever a implementação da solução proposta no capítulo anterior, podendo ser dadas explicações e evidências de soluções intercalares. Devem também ser descritas as tecnologias e metodologias utilizadas (software, sistemas de operação, linguagens, dispositivos ou outras ferramentas) e perspectiva crítica sobre as mesmas.

Esta secção descreve a implementação da solução proposta no capítulo anterior.

Alguns dos diagramas referidos na secção anterior podem aparecer nesta secção. Por exemplo, diagramas de classes ou diagramas de módulos, sendo detalhadas as operações de cada classe ou as funções de cada módulo (diagramas de atividades).

Devem também ser descritas as especificidades de implementação de acordo com o ambiente de desenvolvimento, plataforma e linguagem escolhida para o desenvolvimento e deve ser claro que o desenho apresentado anteriormente foi, de facto, adotado na implementação

Descrição da fase de implementação no processo de Engenharia de Software

3.1.1 Tecnologias Usadas

Menção da divisão das tecnologias em duas categorias: pré-desenvolvimento e desenvolvimento

Tecnologias de Pré-desenvolvimento

- Plantuml
- VSCode
- Draw.io
- Figma (anexo com imagens dos mockups)
- Github Projects (mostrar setup do projecto e dos cartões) + Repository

Site do *Figma*: <https://www.figma.com/>

Site do *Github*: <https://github.com/>

Site do *Drawio*: <https://www.drawio.com/>

Site do *Plantuml*: <https://plantuml.com/>

Site do *Visual Studio Code*: <https://code.visualstudio.com/>

Tecnologias de Desenvolvimento

- Visual Studio Code
- React
- Next.js
- Tailwind CSS
- http requests
- Vuexy template
- AWS S3
- Prisma - Railway com MySQL database

Site do *React*: <https://react.dev/>

Site do *Next.js*: <https://nextjs.org/>

Site do *Tailwind CSS*: <https://tailwindcss.com/>

Site do *Vuexy*: <https://pixinvent.com/vuexy-bootstrap-html-admin-template/>

Site do *AWS S3*: <https://aws.amazon.com/s3/>

Site do *Prisma*: <https://www.prisma.io/>

Site do *Railway*: <https://railway.com/>

3.2 Funcionalidades

3.2.1 Home Page | Dashboard

- KPIs
- ESG Score (+ formula)
- Best metrics since last record
- Worse metrics since last record

3.2.2 Página das Métricas

- Display
- Cards
- Filtering
- Creation of Metrics

3.2.3 Mapa de Materialidade

- Mapa Display
- mention that when clicked it will open a generic metric card

3.2.4 Página dos Datasets

- Display
- Card
- Show Filtering
- Show import
- Show deletion (maybe)

3.2.5 Página dos Objetivos

mention its a complementary part tha the company showed interest in implementing

- Display
- Creation of objectives

3.3 Testes

- Teste unitarios aos componentes - Alguns de integraçao

mostrar apenas print das cenas a passarem + 1 exemplo de unitario, 1 exemplo de integraçao (maybe E2E ???)

A descrição dos testes efetuados (e.g. unitários, funcionais, de integração, de sistema) pode ser feita nesta secção ou, caso não se justifique, na secção anterior.

3.4 Avaliação da solução

Nesta secção deve ser descrita a abordagem seguida para avaliar a solução, ou parte da solução, nomeadamente um ou mais requisitos de qualidade (e.g. desempenho, usabilidade).

São descritas experiências efetuadas e apresentados os dados/modelos utilizados, bem como os resultados obtidos.

Devem ser descritos e avaliados os resultados obtidos.

Deve ser feita uma discussão sobre a adequação dos resultados obtidos relativamente aos planeados anteriormente.

Esta secção poderá não existir em alguns relatórios de projeto/estágio, mas nesse caso deverá ser dada uma justificação para tal.

Capítulo 4

Conclusões

O capítulo de conclusões é um dos mais importantes do relatório, devendo ser apresentado um resumo dos resultados do trabalho efetivamente desenvolvido. **As conclusões devem basear-se nos resultados realmente obtidos.** Devem enquadrar-se os resultados obtidos com os objetivos enunciados e procurar extrair conclusões mais gerais, eventualmente sugeridas pelos resultados. Podem acompanhar as conclusões incluindo recomendações apropriadas resultantes do trabalho, nomeadamente sugerindo e justificando eventuais extensões e modificações futuras.

4.1 Objetivos concretizados

Nesta secção devem ser repetidos os objetivos apresentados no capítulo de introdução e para cada um deles deve ser descrito o seu grau de realização. Recomenda-se o uso de uma lista ou tabela, dado que facilita a leitura e compreensão.

4.2 Limitações e trabalho futuro

Nesta secção devem ser identificadas as limitações do trabalho realizado, fazendo uma análise autocrítica do trabalho realizado, bem como extrapolar eventuais direções de desenvolvimento futuro.

4.3 Apreciação final

Esta secção deve fornecer uma opinião pessoal sobre o trabalho desenvolvido.