# Relatório de “SGRAI”

## “Relatório Final”

**Turma 3DJ _ Grupo 03**
1231274_Jorge Ubaldo
1230564_Francisco Santos
1230839_Emanuel Almeida
1230444_Romeu Xu

**Data: 10/12/2025**

# Índice

# Parte I – Introdução e estrutura do trabalho

## I.1 Introdução

A presente componente integra-se na **ThPA Port Management Platform**, um sistema académico desenvolvido no âmbito do projeto integrado do 5.º semestre da Licenciatura em Engenharia Informática (ISEP), focado na gestão de operações no Porto de Salónica.

O objetivo desta componente é disponibilizar um **visualizador 3D interativo** do porto, totalmente integrado com os dados da Web API (.NET), permitindo ao utilizador observar, de forma intuitiva, a distribuição de recursos físicos e logísticos: docas, navios, áreas de armazenamento, contentores, parques de estacionamento, equipamentos e estruturas de apoio.

Mais do que ser apenas "decorativo", o viewer 3D pretende ser uma **ferramenta de apoio à decisão**, facilitando a leitura do estado do sistema:

- Que navios estão presentes e onde estão atracados;
- Como estão distribuídas as áreas de armazenamento e contentores;
- Que recursos físicos existem em cada zona;
- Que vias de circulação, parques e elementos complementares existem no porto.

Este relatório descreve de forma sucinta o trabalho realizado, as principais opções técnicas, a interação com o utilizador (rato/teclado) e algumas particularidades relevantes da implementação. Termina com uma conclusão e com um conjunto de propostas de melhoria futura.

## I.2 Enquadramento e estrutura global do trabalho

O trabalho desenvolvido pode ser organizado em três grandes blocos:

1. Integração de dados com o backend
    - Foram criados serviços específicos para carregar, a partir da Web API, informação sobre docas, navios aceites (VVN "Accepted"), áreas de armazenamento, contentores e recursos físicos.
    - Estes dados são convertidos em DTOs específicos do viewer 3D (DockDto, VesselDto, StorageAreaDto, ContainerDto, PhysicalResourceDTO, agrupados em SceneData).

- Ainda no lado do cliente, é aplicada uma primeira camada de normalização (por exemplo, cálculo de TEU, largura/altura aproximada de áreas, valores por defeito, filtragem de VVN, etc.).

2. Cálculo de layout sobre uma grelha lógica (portGrids)

   - O porto é dividido em zonas A, B e C, cada uma subdividida em células (A.1, A.2, B.1, B.2, C.1…C.10), representadas por retângulos (Rect) e, nalguns casos, por grelhas internas.
   - Foi criado um "layout engine" (layoutEngine.ts) que, com base nesta grelha, decide onde e como posicionar:
     - Áreas de armazenamento em B;
     - Contentores reais em A.2;
     - Docas físicas à volta da Zona C;
     - Navios na água, alinhados com as docas;
     - Estruturas decorativas (storage áreas decorativas e gruas decorativas) em Zona C;
     - Recursos físicos sob a cobertura de A.1.
   - Este layout é calculado de forma automática, garantindo que não existem sobreposições óbvias e respeitando margens, folgas de estrada, distâncias ao mar, etc.

3. Construção da cena 3D e interação com o utilizador

   - Com o layout calculado, a cena 3D é construída usando Three.js, com um conjunto de serviços de placement e de criação de objetos (GLBs normalizados, geometrias básicas e materiais PBR simples).
   - A interação é feita através de um componente React (PortViewer) que cria o canvas WebGL, inicializa a câmara, luzes, controlos de órbita, picking com o rato e ligação ao painel de informação no lado direito / header de estatísticas.
   - A cena inclui também elementos de contexto: plano de base do porto, água, cidade de fundo, parques de estacionamento, tráfego rodoviário, toldos modernos, oficinas, armazéns decorativos, etc., contribuindo para uma leitura mais realista do ambiente.

Esta estrutura modular permite separar responsabilidades: o backend fornece dados “lógicos”, o layout engine traduz esses dados para o espaço 3D e os módulos Three.js constroem a representação visual final.

# Parte II – Desenvolvimento

## II.1 Descrição da tarefa e requisitos do enunciado

A tarefa proposta para esta componente consistia, em essência, na construção de uma **cena 3D interativa** que representasse o porto e os principais recursos geridos pela plataforma ThPA Port Management Platform. Não se tratava apenas de "desenhar" um cenário genérico, mas sim de **materializar em 3D a informação de domínio** já existente no backend: docas, navios, áreas de armazenamento, contentores e recursos físicos.

Um dos requisitos centrais era precisamente a **integração com dados reais provenientes da Web API**, descartando a abordagem simplista de criar objetos "hard-coded" no cliente. Assim, a cena 3D teria de refletir, tanto quanto possível, o estado atual do sistema, usando os dados expostos pelos serviços REST. Esta opção aproxima a solução de um caso real de aplicação industrial, em que o front-end é um consumidor de dados de negócio e não um gerador de cenários artificiais.

Do ponto de vista da experiência de utilização, o enunciado exigia também a implementação de **interação com o utilizador**, recorrendo ao rato e, quando fizesse sentido, ao teclado. Isso incluía a navegação pela cena (controlo da câmara, zoom, rotação e movimento lateral) e a **seleção de elementos 3D** para consulta de informação mais detalhada. Desta forma, o viewer não se limita a ser uma "imagem bonita", mas torna-se um **componente explorável**, em que o utilizador pode inspeccionar entidades e compreender melhor a situação operacional do porto.

Por fim, a entrega tinha de ser acompanhada por um **relatório em PDF**, onde se descrevesse de forma sucinta o trabalho desenvolvido: uma breve descrição da tarefa, a identificação das tecnologias utilizadas, a explicação do modelo de interação (teclas e ações de rato) e a referência a quaisquer particularidades relevantes da implementação.

A solução desenvolvida não só cumpre estes requisitos como vai um pouco mais longe, ao introduzir uma **lógica de layout relativamente avançada**. Em vez de colocar os elementos em posições arbitrárias, são aplicados algoritmos de distribuição para **posicionar automaticamente contentores, navios, parques de estacionamento, gruas decorativas, edifícios de apoio e áreas decorativas**, garantindo coerência espacial com a grelha lógica do porto. O resultado é uma cena que, embora simplificada, funciona como um reflexo razoavelmente consistente da configuração real do sistema, respeitando zonas, distâncias, margens e relações entre objetos.

## II.2 Tecnologias utilizadas

A implementação da componente baseia-se num conjunto de tecnologias modernas que, em conjunto, permitem criar uma aplicação **full-stack** com visualização 3D integrada.

No **front-end** foi utilizada a combinação **React + TypeScript** para estruturar a SPA (Single Page Application). React é responsável pela construção dos componentes de interface, gestão de estado e ciclo de vida, enquanto TypeScript garante tipagem estática e maior segurança ao manipular os DTOs vindos da API e as estruturas internas do viewer 3D. O componente principal do porto é responsável por **carregar os dados da Web API**, armazená-los no estado local, desencadear o cálculo de layout e, finalmente, inicializar e atualizar a cena Three.js. É também em React que se fazem as ligações ao UI "tradicional", como o cabeçalho do viewer, os botões de filtro de layers, o painel de informação e a lógica de seleção de entidades.

Para a visualização 3D propriamente dita foi utilizada a biblioteca **Three.js**, que fornece as primitivas para criação de **cena, câmara, luzes, sombras e materiais**. Através do Three.js é possível carregar **modelos GLB** (navios, contentores, veículos, edifícios, cidade de fundo, etc.), combiná-los com geometrias procedurais (planos, cilindros, caixas) e aplicar materiais com propriedades adequadas à cena (metalness, roughness, opacidade, etc.). A navegação da câmara é feita com **OrbitControls**, permitindo ao utilizador orbitar o porto, fazer zoom e deslocar a vista. O **picking de objetos** é feito por raycasting: um raio é lançado a partir da posição do rato no ecrã para dentro da cena 3D e, quando interseta um objeto "clicável", são lidos metadados associados a esse objeto para descobrir que entidade lógica foi selecionada (por exemplo, um navio ou uma doca).

Em cima do core de Three.js foram desenvolvidos vários **módulos específicos de layout e placement**. O módulo portGrids define a malha lógica do porto, dividindo-o em zonas (A, B, C) e subzonas (A.1, A.2, B.1, B.2, C.1…C.10), cada uma com um retângulo associado. O **motor de layout** (layoutEngine.ts) centraliza o cálculo de posições e rotações, delegando em módulos especializados tarefas como: criar parques de estacionamento em C.1/C.2 (addAngleParkingInC), distribuir oficinas e edifícios complementares em C.3/C.4 (addWorkshopsInC34 e addExtrasRowInC34), construir zonas de pilhas de contentores decorativos junto à água em C.7–C.10 (addContainerYardsInC78910), gerar toldos modernos em A.1 (addModernAwningsInA1), adicionar tráfego rodoviário nas estradas principais e internas (addRoadTraffic) e posicionar contentores ou storage areas em células específicas (placeResourcesUnderA1, placeStorageAreasInB, placeContainersA2_Max2PerSlot, entre outros).

Do lado do **backend**, a componente comunica com uma **Web API em .NET**, que expõe endpoints para aceder aos dados de Dock, StorageArea, Vessel, Container e PhysicalResource, bem como às VesselVisitNotification em estado Accepted. O acesso a estes dados é feito através de um cliente HTTP (webApi), que obtém as respostas JSON, converte os campos para tipos adequados e aplica algumas normalizações (por exemplo, cálculo de largura/altura a partir de TEU, saneamento de datas e estados, escolha da VVN mais recente por IMO, etc.).

Esta pilha tecnológica – React/TypeScript, Three.js, módulos de layout dedicados e Web API .NET – permite que a camada visual 3D funcione como **projeção direta da informação de domínio**, beneficiando da robustez do backend e da flexibilidade gráfica do front-end.

## II.3 Interação com o utilizador

A interação com o utilizador assenta sobretudo no **rato**, refletindo o padrão habitual de visualizadores 3D técnicos, nos quais a manipulação da câmara e a inspeção dos objetos é feita de forma direta e intuitiva.

No que diz respeito às **ações de rato**, um clique simples com o botão esquerdo sobre um elemento da cena desencadeia um processo de **seleção**, suportado por raycasting. Cada objeto "interessante" do ponto de vista de negócio (docas, navios, áreas de armazenamento, contentores, recursos físicos, gruas decorativas, etc.) é criado com metadados associados que incluem, pelo menos, um identificador e um tipo. Quando o utilizador clica, o sistema traduz o resultado do raycast num "pick" lógico do tipo {type, id, label}.

Essa pequena estrutura é então entregue à função mapPickedToSelection, que a utiliza para localizar o **DTO completo** na estrutura SceneData. A partir daí, o front-end consegue mostrar, na interface, informação detalhada sobre a entidade selecionada: por exemplo, nome e código do navio, comprimento e calado, doca associada, ou características de uma storage area. Assim, o utilizador tem a sensação de estar a "clicar" em objetos 3D, mas o sistema está, na prática, a trabalhar com entidades de domínio.

Ainda com o rato, o utilizador pode **arrastar** para controlar a câmara. Com OrbitControls, o arrastar (com o botão esquerdo ou direito, conforme configuração) permite orbitar em torno do porto e fazer pan, deslocando o enquadramento horizontal e verticalmente. Isto facilita a inspeção de zonas específicas, como as docas da Zona C, os parques de estacionamento ou as áreas de contentores, sem perder a noção global da cena. A **roda do rato** é usada para zoom, aproximando ou afastando a câmara em relação ao

centro de interesse corrente. Estão definidos limites para evitar que a câmara atravesse o plano da água ou se afaste tanto que o porto se torne demasiado pequeno ou irrelevante no ecrã.

Quanto ao **teclado**, a versão atual da componente opta por uma abordagem minimalista. Não foram definidos atalhos de navegação específicos (como esquemas WASD para mover a câmara), precisamente para simplificar o modelo de interação e não sobrecarregar o utilizador com combinações de teclas. Na prática, o teclado é utilizado quase exclusivamente pelas próprias funcionalidades do browser (por exemplo, F11 para fullscreen global) e por eventuais campos de texto presentes noutras partes da UI.

Esta decisão liberta a aprendizagem do utilizador, que apenas precisa de compreender as ações de rato para explorar o porto, e deixa espaço para, numa fase posterior, introduzir **atalhos de teclado bem documentados**: por exemplo, teclas para focar automaticamente em docas ocupadas, centrar a câmara na Zona C ou alternar rapidamente a visibilidade de certas camadas (tráfego, cidade, elementos decorativos, etc.).

## II.4 Peculiaridades e aspetos relevantes da implementação

Apesar de a componente se basear em conceitos relativamente clássicos de visualização 3D, existem vários aspetos específicos da implementação que acrescentam profundidade e complexidade ao trabalho.

Um primeiro ponto importante é a **normalização de modelos GLB**. Em vez de assumir que todos os modelos externos têm origem, escala e orientação "perfeitas", foi criado o utilitário loadGLBNormalized, responsável por: recentrar os modelos em XZ, assentar a base em Y = 0, calcular uma Box3 para medir o tamanho real do objeto e aplicar, quando necessário, uma escala que o faça caber em **células definidas pelo layout**. Este processo é usado de forma consistente para vários tipos de elementos – oficinas, edifícios complementares, contentores decorativos, veículos estacionados, carros e camiões em movimento, etc. – reduzindo discrepâncias visuais e permitindo que a cena mantenha proporções coerentes independentemente das origens dos ficheiros GLB.

Outro aspeto relevante é o **relacionamento entre navios e docas**. A função placeDocksC não coloca as docas ao acaso: ela calcula até oito docas físicas encostadas às bordas da Zona C (topo, esquerda, direita), com comprimento ajustado à dimensão das células respetivas e orientação consistente com a linha de costa. Em seguida, placeVesselsOnWater recebe a lista de navios e alinha cada um com uma doca, aplicando escalas e deslocamentos para garantir que o navio não "atravessa" o cais. O comprimento do navio é ajustado para não ultrapassar a dock, a largura é utilizada para calcular uma distância de

segurança em relação ao muro, e é introduzido um **offset lateral** que posiciona o navio na água, mantendo uma folga parametrizável. Existem ainda parâmetros de ajuste fino (como lengthScale, addLengthM, widthScale, entre outros) que permitem calibrar visualmente o encaixe dos modelos.

A **modelação de parques de estacionamento e tráfego rodoviário** constitui outro exemplo de pormenor. O módulo addAngleParkingInC calcula vagas de estacionamento oblíquas a 60°, desenhando as marcas no chão com pequenos planos e usando parâmetros como largura da vaga, comprimento e folga entre filas. Com base numa probabilidade configurável, algumas destas vagas são ocupadas por veículos GLB (carros ligeiros e camiões), posicionados e rodados de acordo com a geometria do parque. Em paralelo, addRoadTraffic define um conjunto de segmentos de estrada (vias principais, rua horizontal entre A e B, malha interna de Zona C) e distribui, ao longo desses segmentos, veículos em movimento "congelado", criando a sensação de **porto ativo**. A quantidade de veículos por segmento é proporcional ao comprimento da via, garantindo uma densidade visual consistente.

Na zona A.1, os **toldos modernos** criados por addModernAwningsInA1 representam uma solução de geometria paramétrica. Em vez de carregar um modelo estático, a função constrói programaticamente postes, vigas, barras de acabamento e um "tecido" translúcido com cantos suavemente arredondados. A altura dos beirais, a inclinação da cobertura e a extensão das saliências nas extremidades podem ser ajustadas via parâmetros. Sob esta cobertura, o módulo placeResourcesUnderA1organiza os recursos físicos (gruas, equipamentos de movimentação, etc.) numa grelha, calculando automaticamente o número de colunas e linhas, bem como as folgas necessárias, de modo a evitar sobreposições e a manter um aspeto de arrumação coerente.

Na Zona C, o tratamento das áreas de contentores e dos elementos decorativos também foi pensado com algum detalhe. O módulo addContainerYardsInC78910 é responsável por criar **pilhas de contentores decorativos** em C.7 a C.10, usando um viés em X que aproxima as pilhas do lado da água, sem invadir as faixas reservadas a circulação. Para cada zona, é calculado um sub-retângulo "encostado" ao mar, dentro do qual são geradas colunas e linhas de pilhas com altura máxima configurável. Ao mesmo tempo, placeDecorativeStorageAreasZoneC insere "faixas" de áreas de armazenamento decorativas tanto nas laterais (C.7, C.8, C.9, C.10) como no topo (C.5, C.6), com dimensões uniformizadas e orientações consistentes (+X, −X ou +Z), reforçando a leitura de que a Zona C é um espaço intensivo de operações logísticas.

Por fim, o módulo placeDecorativeCranesZoneC trata do posicionamento das **gruas decorativas** ao longo das docas. Para cada doca é calculado um retângulo representando o comprimento e a profundidade do cais e, com base nisso, é determinada a largura, profundidade e altura da grua. A posição da base é escolhida de forma a encostar a estrutura ao lado da água, mas com uma folga para não "invadir" a linha de cais. Para além da orientação do chassis, é calculada também a **orientação do braço (boom)** em função da célula C.* em que a doca se encontra: por exemplo, em C.5–C.8 o boom aponta para +X, enquanto em C.9 pode apontar para −Z, de forma a sugerir que a grua está a operar na direção do mar ou dos contentores. Este cuidado faz com que as gruas não sejam apenas objetos decorativos, mas também **sugestões visuais da direção de operação**.

No conjunto, estas peculiaridades contribuem para um porto que, apesar de simplificado, apresenta um **nível de detalhe e coerência espacial** superior ao de uma simples maquete. A ligação entre a grelha lógica (portGrids), os módulos de layout e os modelos 3D concretos é o que permite transformar dados de domínio em uma representação visual rica e informativa.

# Parte III – Conclusão

## III.1 Conclusão

A componente desenvolvida atinge o objetivo principal definido no enunciado: disponibilizar uma **visualização 3D interativa do porto**, ligada a dados reais do sistema e capaz de representar, de forma coerente, os principais recursos da plataforma (docas, navios, contentores, áreas de armazenamento e recursos físicos). Ao longo do desenvolvimento, a preocupação central foi sempre evitar um "demo" puramente decorativo e aproximar a cena de um **cenário operacional plausível**, no qual a informação vem do backend e é tratada de forma sistemática, tanto na camada de dados como na camada de layout.

Do ponto de vista técnico, a solução demonstra a **integração eficaz entre uma Web API em .NET e um front-end moderno em React + TypeScript com Three.js**, incluindo a normalização de DTOs, o cálculo de posições e rotações com base numa grelha lógica (portGrids) e a utilização consistente de módulos de placement especializados. A forma como navios são acoplados às docas, como as pilhas de contentores decorativos são encostadas à água sem interferir com as estradas, como os parques de estacionamento respeitam margens e ângulos, ou como os recursos físicos surgem organizados sob coberturas em A.1, revela uma preocupação com **regras de composição espacial** em vez de simples coordenadas fixas.

Em termos de interação, a escolha de centrar a experiência no **rato e no modelo de OrbitControls** permite que o utilizador explore o porto de forma intuitiva, orbitando, aproximando e afastando a câmara, enquanto pode **selecionar entidades específicas** para ver detalhes no UI. O mecanismo de picking, apoiado em metadados e num mapeamento posterior para DTOs completos, garante que o clique do utilizador tem sempre tradução no domínio da aplicação, o que é essencial para que o viewer 3D não seja um "mundo à parte", mas sim uma extensão visual do sistema de informação.

Naturalmente, a solução apresenta também **limitações**: a cena, embora coerente, continua a ser uma abstração do porto real; muitos elementos são decorativos e não representam, ainda, estados operacionais dinâmicos em tempo real; o comportamento dos veículos é estático; e a própria interface poderia expor mais funcionalidades de filtragem, destaque e navegação orientada a tarefas. Apesar disso, para o contexto desta componente, o resultado é consistente com os requisitos e constitui uma **base sólida para evoluções futuras**, tanto no plano visual como no plano funcional.

Do ponto de vista pedagógico, o desenvolvimento desta componente permitiu consolidar competências em **integração full-stack**, estruturação de código em módulos reutilizáveis (layout, placement, carregamento de modelos, mapeamento de picks), bem como uma melhor compreensão dos desafios específicos de trabalhar com **gráficos 3D em aplicações de negócios**: escalas, orientação, desempenho, usabilidade e relação entre dados abstratos e representações visuais concretas.