

Universidade do Minho
Escola de Engenharia
Mestrado em Engenharia Informática

Unidade Curricular de Aplicações e Serviços de Computação em Nuvem

Ano Letivo de 2024/2025

1º Checkpoint - Airtrail



Miguel Gramoso
A100835



Luís Figueiredo
PG61532



Gustavo Barros
PG61527



Vasco Mota
PG58811



Enzo Vieira
PG61518

November 03, 2025

ASCN

Índice

1. Introdução	2
2. Arquitetura Geral	2
2.1. <i>Frontend</i>	2
2.2. <i>Backend</i>	3
2.3. Base de Dados	3
2.4. Deployment	3
3. Funcionalidades	3
4. APIs fornecidas pela aplicação	4
5. Potenciais gargalos de desempenho e pontos únicos de falha na instalação	5
6. Resposta às perguntas e reflexão e identificação de dificuldades na instalação da aplicação	6
6.1. Questão 1: Análise da Instalação Base	6
6.2. Questão 2: Otimizações Propostas	6
6.3. Reflexão e dificuldades na instalação da aplicação	7

1. Introdução

Este relatório serve de apoio ao desenvolvimento do projeto da UC de Aplicações e Serviços de Computação em Nuvem, mais concretamente, ao 1º *Checkpoint* do projeto.

Neste documento pretendemos fazer uma análise profunda da aplicação *Airtrail*, uma solução open-source e self-hosted que permite aos seus utilizadores registar, visualizar e analisar o seu histórico de viagens aéreas através de uma interface moderna e intuitiva.

A análise a esta aplicação pretende encontrar possíveis pontos de falha ou *bottlenecks* presentes que possam vir a dificultar o *deployment* em nuvém.

2. Arquitetura Geral

O *Airtrail* é uma aplicação desenvolvida focada em uma arquitetura monolítica full-stack em *SvelteKit* com o deployment feito com o auxilio de *Docker*. A aplicação foi desenhada de modo a utilizar um modelo cliente-servidor comum, onde o *Sveltekit* funciona tanto como *frontend* e *backend*.

2.1. Frontend

O *frontend* da aplicação foi construído com *Svelte*, resultando em uma *UI* apelativa aos utilizadores e de fácil compreensão. Os utilizadores conseguem observar um mapa com os seus voos

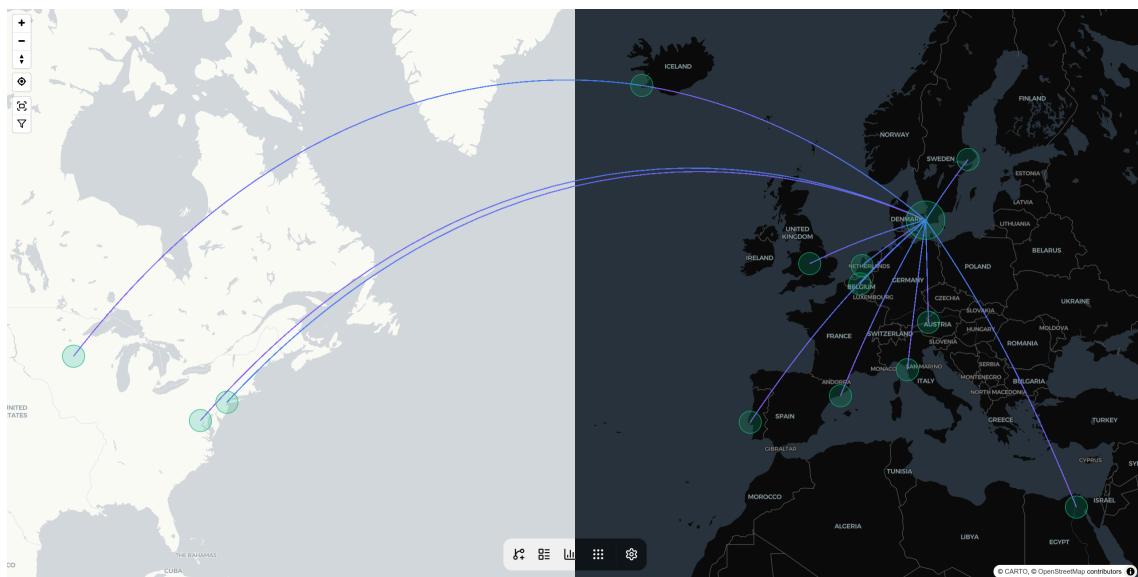


Figura 1: Exemplo de um mapa de voos

2.2. Backend

O *Backend* utiliza uma API RESTful através das *server routes* do *Sveltekit*. É responsável por todas as operações relacionadas com a lógica da aplicação, desde o processamento de voos até à validação de dados.

A autenticação é implementada através de OIDC (OpenID Connect), que permite a integração com múltiplos *providers*.

Um dos componentes a destacar são os múltiplos *parsers* para sites comuns de *track* de voos, como o MyFlightRadar24, App in the Air e JetLog.

2.3. Base de Dados

A base de dados utilizada pelo *AirTrail* é construída com *PostgreSQL*, um sistema de gestão de base de dados relacional. O *schema* da base de dados está organizado em tabelas principais que incluem registos de voos com as informações de origem, destino, datas, duração e companhia aérea, dados de utilizadores e as suas credenciais de autenticação, informações detalhadas sobre veículos utilizados, incluindo modelos e registos, catálogo de companhias aéreas, e base de dados de aeroportos com códigos identificadores e coordenadas geográficas.

2.4. Deployment

A aplicação já foi desenhada para a utilização de *Docker* no seu *deployment*, uma ferramenta que os alunos aprenderam a utilizar durante o decorrer da UC. É recomendado a utilização de *Docker Compose* para organizar tanto o *container* da aplicação quanto o *container* de *PostgreSQL*.

3. Funcionalidades

O *AirTrail* disponibiliza um conjunto abrangente de funcionalidades que permitem ao utilizador gerir e visualizar os seus voos de forma intuitiva. Entre as principais funcionalidades destacam-se:

- **Autenticação:** sistema de *login* com suporte a múltiplos utilizadores e configuração de OAuth.
- **Atualizações:** notificação automática quando existe uma nova versão da aplicação, com instruções de atualização.
- **Gestão de voos (CRUD):** criação, edição, remoção e consulta de voos, com detalhes como origem, destino, horários, companhia aérea e passageiros.

- **Mapa-mundo interativo:** visualização geográfica dos voos, distinguindo entre voos passados e futuros; inclui filtros, localização atual do utilizador (caso haja essa permissão), navegação livre, redefinição do norte e controlo de *zoom*.
- **Listagem de voos:** apresentação em formato de lista, com filtros por data e outros critérios.
- **Estatísticas:** cálculo e visualização de estatísticas relacionadas com os voos já realizados, com a possibilidade de expandir cada gráfico para ecrã completo e consultar detalhes adicionais sobre os dados apresentados.
- **Ferramentas adicionais:**
 - Consola SQL integrada para interação direta com a base de dados.
 - Função para deteção e remoção de voos duplicados.
- **Globo de países visitados:** representação 3D dos países visitados, vividos, visitados em escala ou adicionados à lista de desejos, com possibilidade de sincronizar automaticamente a informação com os voos registados.
- **Definições e gestão de dados:** opções gerais da aplicação, troca entre modo claro e modo escuro, partilha pública de voos (*shares*), importação de dados de outras plataformas (como *FlightRadar24*, *App in the Air*, etc.), exportação de dados em *JSON* ou *CSV*, atualização automática e adição manual de informações sobre aeroportos, companhias aéreas e aviões.
- **Integração externa:** ligação opcional ao serviço *AeroDataBox*, que fornece uma pesquisa de voos melhorada como forma de substituição do serviço *adsbdb*.
- **Gestão de utilizadores (CRUD):** criação, edição, remoção e consulta de utilizadores.
- **Configuração de permissões e autenticação:** ativação e gestão da autenticação via OAuth, que controla o acesso dos utilizadores à instância *AirTrail* e permite automatizar o processo de início de sessão ou de registo de utilizadores.

4. APIs fornecidas pela aplicação

O *AirTrail* disponibiliza uma API REST que permite a gestão completa dos voos registados pelo utilizador. As principais rotas incluem:

- `GET /flight/list` – devolve a lista de todos os voos do utilizador.
- `GET /flight/get/{id}` – obtém os detalhes de um voo específico.
- `POST /flight/save` – cria um novo voo ou atualiza um voo existente, consoante a presença do campo `id`.
- `POST /flight/delete` – remove um voo com base no seu identificador.

Estas rotas são utilizadas internamente pela aplicação para sincronizar os dados entre o *frontend* e o servidor.

Estas são as rotas documentadas oficialmente, responsáveis pela gestão de voos. Outras funcionalidades da aplicação são tratadas internamente e não possuem *endpoints* públicos descritos.

5. Potenciais gargalos de desempenho e pontos únicos de falha na instalação

No contexto da aplicação *AirTrail*, os potenciais gargalos de desempenho surgem quando determinados componentes limitam a escalabilidade ou aumentam a latência à medida que o número de utilizadores cresce. O servidor pode tornar-se um ponto crítico de estrangulamento caso o sistema não suporte escalamento horizontal, já que um único servidor sobrecarregado com múltiplos pedidos tende a provocar tempos de resposta elevados. A base de dados da *AirTrail*, se configurada como uma instância única sem mecanismos de replicação ou partição, constitui igualmente um fator limitante. Bloqueios simultâneos e excesso de conexões podem comprometer o desempenho global da aplicação. A infraestrutura de rede também representa um elemento sensível, pois a largura de banda insuficiente ou a elevada latência podem reduzir o throughput e afetar diretamente a experiência do utilizador.

Relativamente aos pontos únicos de falha na instalação da aplicação, estes englobam tanto os componentes críticos que podem afetar a disponibilidade da *AirTrail* como as dificuldades inerentes ao seu processo de instalação e automação. A inexistência de redundância no servidor pode levar à indisponibilidade total da aplicação em caso de falha. Da mesma forma, uma base de dados sem réplicas ou backups constitui um ponto único de falha, pois a sua indisponibilidade impede o acesso aos dados essenciais da *AirTrail*. A ausência de mecanismos para balancear a carga agrava este risco, uma vez que a falha de um único componente pode afetar todos os utilizadores. Adicionalmente, configurações ineficientes ou incorretas podem dificultar a recuperação automática após incidentes, comprometendo a resiliência e a estabilidade do sistema.

As dificuldades na instalação e automação da *AirTrail* estão frequentemente associadas a dependências de software mal geridas, incompatibilidades entre versões de pacotes e bibliotecas, ou configurações manuais que aumentam a probabilidade de erro humano. Problemas relacionados com a criação e configuração de containers, definição de redes, e permissões podem dificultar o processo de deployment. A complexidade do deployment da *AirTrail*, que exige coordenação entre vários componentes, representa também um desafio adicional, sobretudo na ausência de scripts ou pipelines bem definidos que garantam consistência e reproduzibilidade no processo de instalação.

6. Resposta às perguntas e reflexão e identificação de dificuldades na instalação da aplicação

6.1. Questão 1: Análise da Instalação Base

- a. Para um número crescente de clientes, que componentes da aplicação poderão constituir um gargalo de desempenho?**

O servidor caso o sistema não suporte escalamento horizontal, pois um único servidor pode ser saturado com pedidos, atrasando assim a resposta a todos eles. A base de dados única, sem replicação, devido a bloqueios e excesso de conexões. A rede é outro ponto crítico: largura de banda insuficiente ou latência elevada degradam o throughput e a experiência do utilizador.

- b. Qual o desempenho da aplicação perante diferentes números de clientes e cargas de trabalho?**

Como foi não feitos testes de desempenho formais nesta fase, a análise baseia-se em análises teóricas. Em cenários com poucos utilizadores, a aplicação deverá manter um desempenho aceitável, visto que os processamento das consultas à base de dados não terão grande carga. Contudo, com o aumento do número de clientes e requisições, é esperado que o tempo médio de espera cresça, por conta da falta de escalabilidade horizontal da instalação. Além disso, a falta de um balanceamento de carga traz um possível risco a falta dos serviços por conta da alta taxa de pedidos concorrentes.

- c. Que componentes da aplicação poderão constituir um ponto único de falha?**

O servidor único e a base de dados única, pois com a falha de um destes componentes, toda a aplicação fica inoperável.

6.2. Questão 2: Otimizações Propostas

- a. Que otimizações de distribuição/replicação de carga podem ser aplicadas à instalação base?**

Algumas técnicas podem ser exploradas para otimizar a distribuição/replicação de carga.

A primeira delas pode ser a escalabilidade horizontal do servidor web, com a criação de múltiplas instâncias do container web. Outra abordagem é a replicação e persistência da base de dados, já que isso aumenta a disponibilidade do serviço e protege a aplicação contra perda de dados em caso de falha de um nó. Por último, a monitorização, escalonamento automático e um *load balancer* estão planeados para uma adição futura, já que as duas primeiras técnicas ajustam dinamicamente o número de instâncias conforme a carga de trabalho e a última distribui igualmente a carga entre os diferentes nós disponíveis.

b. Qual o impacto das otimizações propostas no desempenho e/ou resiliência da aplicação?

O *load balancer* aumenta o *throughput* e melhora o escalonamento horizontal, permitindo que mais réplicas possam ser criadas e mais utilizadores possam utilizar a aplicação. A replicação da base de dados melhora a resiliência e a disponibilidade e a monitorização e escalonamento automático permitirão uma resposta rápida a diferentes cargas de forma transparente.

6.3. Reflexão e dificuldades na instalação da aplicação

Na instalação da aplicação foram detetados erros na configuração da URL da base de dados e a URL `ORIGIN`. Para resolver o primeiro, foi trocar o `DATABASE_HOST` de `db` para `airtrail-pg` que é o nome do container postgres na mesma network. Outro problema encontrado relacionado com a `ORIGIN` foi identificado ao efetuar o registo na aplicação, com a notificação do erro: “Cross-site POST form submissions are forbidden”. Este erro era causado porque a variável ambiente `ORIGIN` estava mal configurada. A resolução para esta problema foi alterar a variável ambiente para o IP da máquina virtual que está a executar a aplicação.

De notar ainda que a base de dados da aplicação necessita de ser criada manualmente, algo que não é referido no guia de instalação, pois este aborda apenas a instalação do servidor web. Este processo poderia ser automatizado.