Universidade do Minho

Mestrado em Cibersegurança
Mestrado em Engenharia Informática

Aplicações e Serviços de Computação em Nuvem Fase-1

Grupo nº33

Carlos Daniel Silva Fernandes
(PG59783)

Afonso Miguel da Silva Ribeiro
(PG60232)

Pedro Augusto Ennes de Martino Camargo
(PG59791)

Renato Pereira Garcia
(PG61542)

Luís Filipe Pinheiro Silva
(PG59790)

2 de novembro de 2025

Conteúdo

Capítulo 1

Contextualização

1.1 Visão Geral da Aplicação

Numa primeira fase deste projeto, é feito o estudo da aplicação de tracking de voos airtrail. Neste relatório estão presentes detalhes sobre a arquitetura da aplicação, as suas funcionalidades e APIs, bem como uma reflexão e discussão sobre potenciais pontos de falha da aplicação, gargalos de desempenho e possíveis dificuldades na instalação.

O principal objetivo da aplicação **AirTrail** é fornecer um sistema completo de **tracking de viagens aéreas**, permitindo que o utilizador registe e acompanhe todas as viagens realizadas num determinado período de tempo.

1.1.1 Funcionalidades Principais

- **Adição Manual de Viagens:** O utilizador pode adicionar uma viagem manualmente através de formulários disponibilizados pela própria aplicação.
- **Importação de Dados:** O **AirTrail** permite importar ficheiros provenientes de outras plataformas de gestão de voos, facilitando a migração e centralização de dados.
- **Exportação de Dados:** O utilizador pode exportar os seus registos de voo, permitindo criar cópias de segurança (*backups*) ou transferir os dados para outras aplicações compatíveis.
- **Estatísticas de Voo:** A aplicação gera automaticamente estatísticas e métricas baseadas nas viagens registadas, fornecendo ao utilizador uma visão geral do seu histórico de voos.

1.2 Principais Tecnologias

Base de Dados: *PostgreSQL*; **Web Server:** *SvelteKit + TypeScript*; **Backend:** Integrado no frontend via *SvelteKit*; **Acesso à BD:** *Kysely*; **Schema/Migrações:** *Prisma*.

1.3 Instalação da Aplicação

A aplicação *AirTrail* pode ser instalada de duas formas principais: manualmente ou através de *containers* utilizando *Docker*.

A **instalação manual** é um processo mais trabalhoso e pouco fiável para sistemas de maior escala, uma vez que a sua replicação e manutenção tornam-se mais complexas e propensas a erros humanos. Além disso, apresenta menor portabilidade entre diferentes ambientes.

Por outro lado, a **instalação containerizada** com *Docker* oferece maior portabilidade, facilita o processo de implantação e reduz significativamente o risco de inconsistências. Esta abordagem também permite uma escalabilidade mais simples e eficiente, sendo a mais recomendada para ambientes de produção.

1.4 Arquitetura e Componentes

A arquitetura adotada é do tipo **cliente-servidor**, em que o cliente interage com o servidor web que, por sua vez, comunica com a base de dados.

A infraestrutura é composta por dois containers Docker que comunicam entre si através de uma *bridge network* criada automaticamente pelo *docker compose*:

O web server e a base de dados são implementados nos containers airtrail e airtrail_db, respetivamente.

1.5 APIs fornecidas pela aplicação

1.5.1 API REST

A **API REST** da aplicação **AirTrail** fornece um conjunto de endpoints para gestão das viagens registadas por cada utilizador. Abaixo estão listadas as principais rotas:

```
GET  /flight/list       -> Lista todos os voos do utilizador.
GET  /flight/get/[id]   -> Detalhes de um voo específico.
POST /flight/save       -> Cria ou atualiza um voo existente.
POST /flight/delete     -> Remove um voo pelo seu ID.
```

1.5.2 APIs Externas

A aplicação **AirTrail** faz uso de serviços externos para obter e enriquecer informações sobre voos e aeronaves.

- **Adsbdb:** Serviço utilizado para realizar *fetch* de informações associadas a *flight numbers*.
- **AeroDataBox:** Alternativa à Adsbdb, permitindo que o utilizador opte por uma integração premium.

Capítulo 2

Análise de Riscos

2.1 Ponto Único de Falha

O principal risco na arquitetura da aplicação **AirTrail** é o **Web Server**. Caso o servidor falhe por qualquer motivo, todos os clientes conectados perdem imediatamente o acesso à aplicação.

Além disso, a utilização de APIs externas, representa outro ponto de falha potencial. Caso este serviço fique indisponível, o funcionamento do **AirTrail** poderá ser afetado.

Para mitigar o risco, recomenda-se a implementação de **redundância**, **replicação** e **failover**, garantindo alta disponibilidade do serviço.

2.2 Gargalos de Desempenho

À medida que o número de utilizadores cresce, o servidor pode passar por gargalos de desempenho, resultando em tempos de resposta mais lentos ou até falhas.

Com apenas uma instância de servidor web, o sistema não possui escalabilidade horizontal. Assim, em situações de aumento de tráfego, o servidor pode atingir o limite da sua capacidade, afetando o desempenho global da aplicação.

O comando `docker stats` foi utilizado para monitorizar o consumo de recursos dos containers em execução.

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
2acd2d28468c	airtrail	0.01%	488.5MiB / 13.45GiB	3.55%	30.7MB / 19.2MB	400MB / 523MB	60
3e60dfd972ce	airtrail_db	2.37%	59.02MiB / 13.45GiB	0.43%	133kB / 1.32MB	44MB / 1.15MB	6

Figura 2.1: Consumo de recursos dos containers.

Comparando com as especificações mínimas recomendadas pela documentação oficial do projeto (**2 CPU cores**, **2 GB de RAM** e **10 GB de espaço em disco**), observa-se que o consumo real é significativamente inferior aos limites estabelecidos. Isso indica que o AirTrail é uma aplicação leve e eficiente para pequenos ambientes de produção ou testes locais.