SISTEMA DE *STREAMING* DE DADOS EM TEMPO REAL

OBJETIVOS:

- Desenhar e implementar um sistema distribuído com arquitetura de microsserviços orquestrado em Kubernetes para suportar o processamento e visualização de dados em tempo real.
- Construir um fluxo de trabalho de integração contínua e entrega contínua (CI/CD) usando Github Actions e ArgoCD.
- Implementar monitorização dos serviços com Prometheus.

DESCRIÇÃO DO PROJETO PARA O CASO DE CORRIDAS (ADAPTAR PAR OUTRO TIPO DE DADOS):

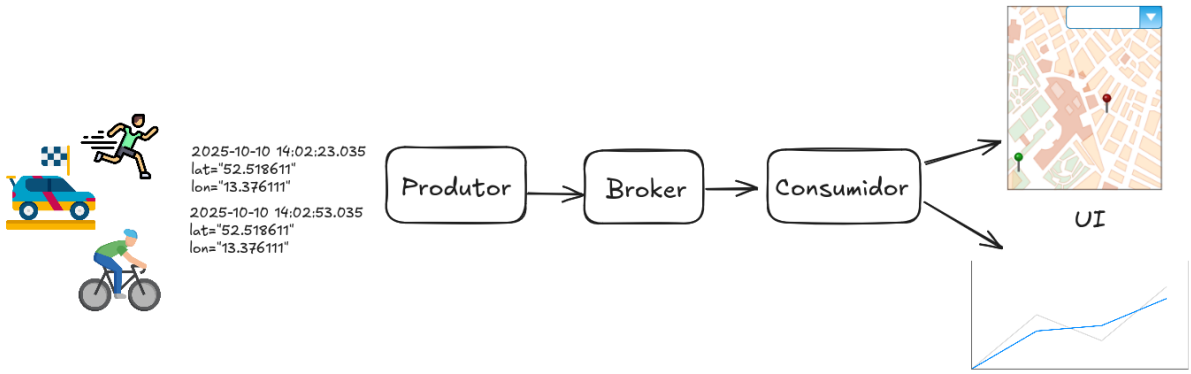

Figura 1. Esquema geral do sistema

Proposta para os requisitos funcionais

Produtor (Serviço que simula os dados em tempo real de uma ou várias corridas)

- Gerar continuamente as posições e velocidades de cada participante (o número de participantes, o número de corridas simultâneas, os percursos e o perfil dos participantes devem ser facilmente configuráveis).
- Publicar esses eventos num ou vários tópicos do broker em intervalos configuráveis

Consumidor (Serviço que fornece dados à UI)

- Lê um tópico do broker e processa/armazena/encaminha para outro servi (componente de ML, p.ex.)
- API REST

UI

- Mapa interativo com a posição de todos os participantes em tempo real
- Filtro para selecionar um subgrupo de participantes e mostrar apenas esses sobre o mapa.

- Tabela de classificação.

Nota de implementação: Explorar a possibilidade de usar Websocket para *stream* em tempo real de dados de um ou vários participantes. Testar os possíveis limites desta opção. Verificar se SSE faria mais sentido ou não.

A arquitetura do sistema deverá prever a existência de armazenamento dos dados em bruto e dos dados já processados. O tipo de armazenamento deverá ser escolhido pelo grupo de projeto. Todos os serviços devem incluir métricas de monitorização que devem ser enviadas para o Prometheus e posteriormente para uma ou várias *dashboards* Grafana.

Proposta para os requisitos não funcionais

- Alta disponibilidade
- Garantir o registo durável de todos os tempos simulados
- Latência ponto-a-ponto não deve exceder 500ms (verificar este valor)
- Segurança (autenticação, autorização e proteção contra DDoS)
- Redimensionamento

1ª FASE: 12/11/2025 CI/CD AUTOMÁTICO COM A APLICAÇÃO BASE (10%):

- Criar um repositório Github público e montar toda a pipeline de CI/CD para o cluster local no Docker Desktop
    - Configurar Github Actions para atualizações no código e envio de imagens para o DockerHub. *Atenção aos limites do DockerHub*[1].
    - Instalar e configurar Argo CD para publicação de novas versões no cluster local.

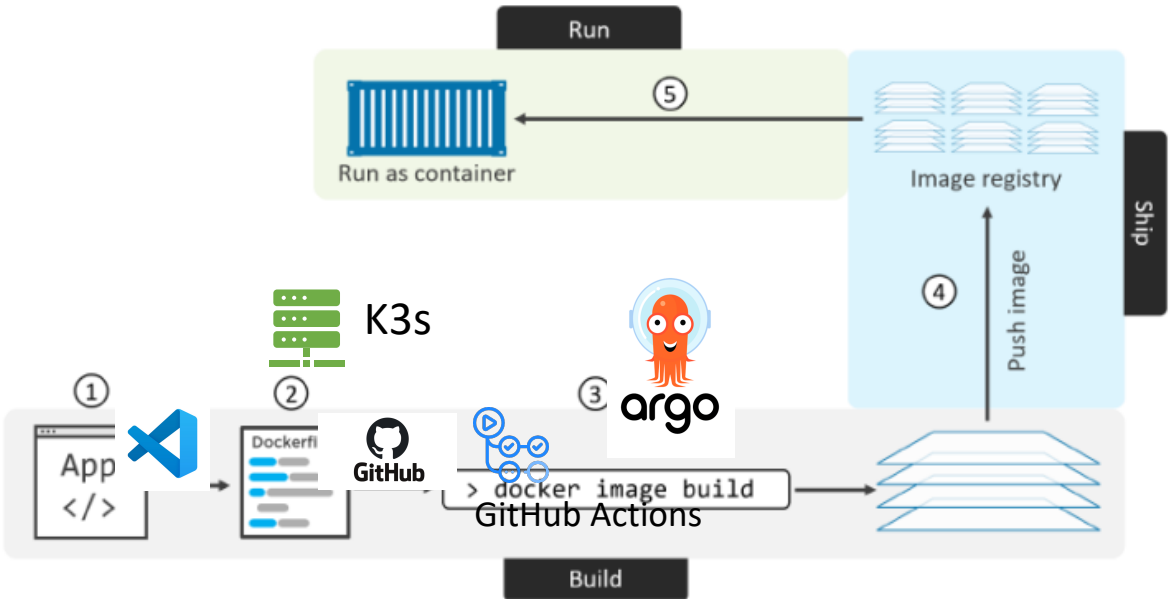

Figura 2. Pipeline CI/CD

[1] https://docs.docker.com/docker-hub/download-rate-limit/

- Simular uma corrida com poucos participantes com o broker instalado no cluster local.
- Aplicação web com mapa e posição dos atletas.

Demonstração a fazer na aula (12/11/2025):

- Demonstrar o funcionamento da aplicação web.
- Atualizar o código e automaticamente efetuar os passos necessários para a nova versão da app correr no cluster local.

2ª FASE: 05/01/2026 IMPLEMENTAÇÃO FINAL COM MONITORIZAÇÃO (50%)

- Simular uma ou várias corridas com número variável de participantes.
- Configurar o sistema (ou propor uma configuração) para alta disponibilidade e redimensionamento automático, baixa latência e alta resiliência.
- O sistema deve funcionar localmente e no cluster geral cujo acesso será disponibilizado aos grupos.
- Apresentar possíveis soluções para armazenamento.
- Refletir sobre mecanismos de comunicação usados.
- Propor mecanismos de segurança mínimos.
- Recolher e disponibilizar métricas relevantes no formato Prometheus.
- Disponibilizar as métricas numa *dashboard* Grafana.
- Funcionamento num cluster local e num cluster remoto (a disponibilizar)
- (opcional) Sugerir e desenhar serviços adicionais (p.ex., subscrições/notificações)
- (opcional) Explorar ferramenta de teste k6 e fazer um teste de carga.

Relatório (até 05/01/2026):

10-20 páginas; A4, Tamanho de letra 12, 1.5 espaçamento entre linhas; margens normais

Proposta de estrutura do relatório:

- Resumo (100-200 palavras)
- Introdução (1-2 pgs)
- Descrição da pipeline CI/CD (1-2 pgs)
- Descrição do sistema implementado (1-4 pgs)
  - Opções de implementação (linguagem, componentes, mecanismos de comunicação)
  - Reflexão sobre os mecanismos de comunicação usados
  - Métricas recolhidas e testes efetuados (2-3 pgs)
- Resultados (2-3 pgs)
- Conclusão e trabalho futuro (1-2 pgs)

Apresentação, demonstração e discussão (09/01/2025)

- Apresentação com os tópicos do relatório.
- Demonstração na sala.

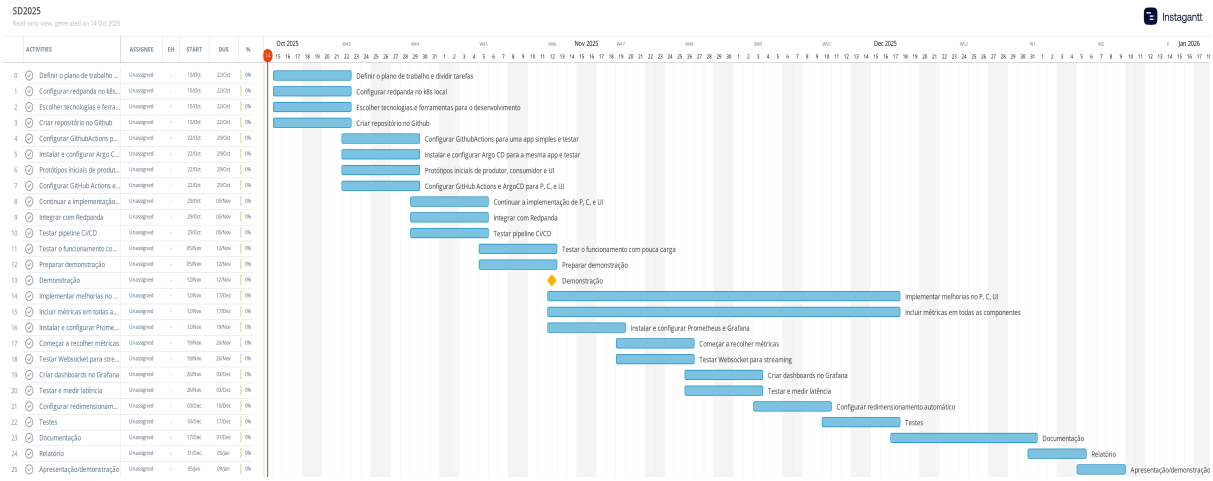

Figura 1 - Cronograma proposto

BIBLIOGRAFIA

Automatização do processo de lançamento da aplicação:

Materiais de apoio no moodle

Folhas de exercícios das aulas TP

Ligações recomendadas:

https://docs.docker.com

https://kubernetes.io/

https://www.redpanda.com

https://www.rabbitmq.com

https://argo-cd.readthedocs.io/en/stable/

https://github.com/features/actions

Techworld with Nana, Youtube.com

Livros:

N. Poulton, Docker Deep Dive, 2020

N. Poulton, P. Joglekar, The Kubernetes book, 2021

J. Domingus & J.Arundel, Cloud Native DevOps with Kubernetes

Sam Newman, Building Microservices, 2021

É permitida a utilização de ferramentas de IA, mas deve ser identificada a ferramenta e em que partes do projeto foi utilizada.

Este enunciado foi construído com a ajuda de Microsoft Copilot e a plataforma IAedu com o modelo OpenAI GPT 4o com acesso à web. A aplicação de demonstração disponibilizada no moodle foi criada usando IAedu com o modelo Open AI GPT 4o com acesso à web.