

**Relatório do Trabalho Prático**

**Desenvolvimento de um Sistema Distribuído**

**1ª Fase - Planeamento**

Arquitetura e Integração de Sistemas

**Docente:** Bruno Lima

**Trabalho realizado por:**

14829 José Gomes

15708 Pedro Carvalho

15709 José Carreira

Ano letivo 2023/2024

Mestrado em Engenharia Informática

Barcelos, novembro 2023

Índice

[Índice 2](#_Toc150844991)

[1. Introdução 3](#_Toc150844992)

[2. Tema escolhido 4](#_Toc150844993)

[2.1. Problema Identificado 4](#_Toc150844994)

[2.2. Análise do problema 4](#_Toc150844995)

[2.3. Abordagem/Estratégia 5](#_Toc150844996)

[3. Requisitos 6](#_Toc150844997)

[3.1. Requisitos Funcionais 6](#_Toc150844998)

[3.2. Requisitos não funcionais 7](#_Toc150844999)

[4. Arquitetura do Sistema 8](#_Toc150845000)

[5. Pontos a salientar 9](#_Toc150845001)

[5.1. Elk Stack 9](#_Toc150845002)

[5.2. Ocelote Gateway 10](#_Toc150845003)

[5.3. xUnit – Teste unitários 10](#_Toc150845004)

[6. Conclusão 11](#_Toc150845005)

# Introdução

Este relatório é apresentado no âmbito do trabalho prático da unidade curricular de Arquitetura e Integração de Sistemas, do Mestrado de Engenharia Informática. Esta UC tem como objetivo dar a conhecer os fundamentos de arquiteturas de sistemas, baseadas em *microservices.*

Pretende-se efetuar o desenvolvimento de um sistema distribuído que permita gerir e agilizar um determinado processo de negócio, utilizando uma arquitetura de sistema baseada em *microservices.* Permitindo independência entre os diferentes *microservices,* tornando o sistema escalável, com flexibilidade na utilização de tecnologias diferentes e agilidade no processo de desenvolvimento.

# Tema escolhido

A aplicação a ser desenvolvida tem como objetivo agilizar a ida a eventos desportivos. Disponibilizando informação útil que torne uma ida a este tipo de eventos mais confortável para o utilizador da aplicação.

## Problema Identificado

Os adeptos de futebol normalmente gostam de assistir a jogos, sejam eles da equipa que apoiam ou outro qualquer jogo.

O problema identificado foi a complexidade na preparação de uma ida a um evento desportivo. A indisponibilidade demonstrada, desconhecimento de possibilidades e falta de tempo que as pessoas têm atualmente torna difícil responder ás seguintes questões:

* Onde estacionar;
* Onde almoçar/jantar;
* Como chegar lá.

Por estes motivos, uma experiência que deveria ser agradável, com momentos de alegria, entusiasmo e diversão, por vezes acaba por ser motivo de stress e desagradável.

## Análise do problema

Pretende-se desenvolver uma aplicação que ajude a resolver este problema, fornecendo informação ao utilizador sobre diferentes jogos, locais de estacionamento, estado da meteorologia, restaurantes/bares, direções até ao estádio, de forma rápida e assertiva.

Um utilizador pode verificar eventos desportivos do seu clube ou outros eventos que existam perto dele, dentro de um determinado período temporal e, para cada evento, obter um plano com informação relevante (direções, meteorologia, tempo de viagem, restaurantes/bares perto do recinto desportivo e locais de estacionamento).

## Abordagem/Estratégia

A estratégia assumida passa pela criação de um *microservice* para cada funcionalidade que se pretende disponibilizar na aplicação:

* Jogos
* Direções
* Lazer
* Tempo
* Estacionamento

Cada *microservice* tem a sua própria arquitetura e é independente dos restantes, podendo comunicar com *API’s* externas e/ou base de dados. Também será utilizado o *microservice* de autenticação (que funcionará com alojamento na *Cloud*) que será desenvolvido para o trabalho da unidade curricular Sistemas de Computação na *Cloud*, esse *microservice* será desenvolvido tendo em consideração a articulação com o restante sistema, garantindo a segurança e gestão de acessos.

# Requisitos

## Requisitos Funcionais

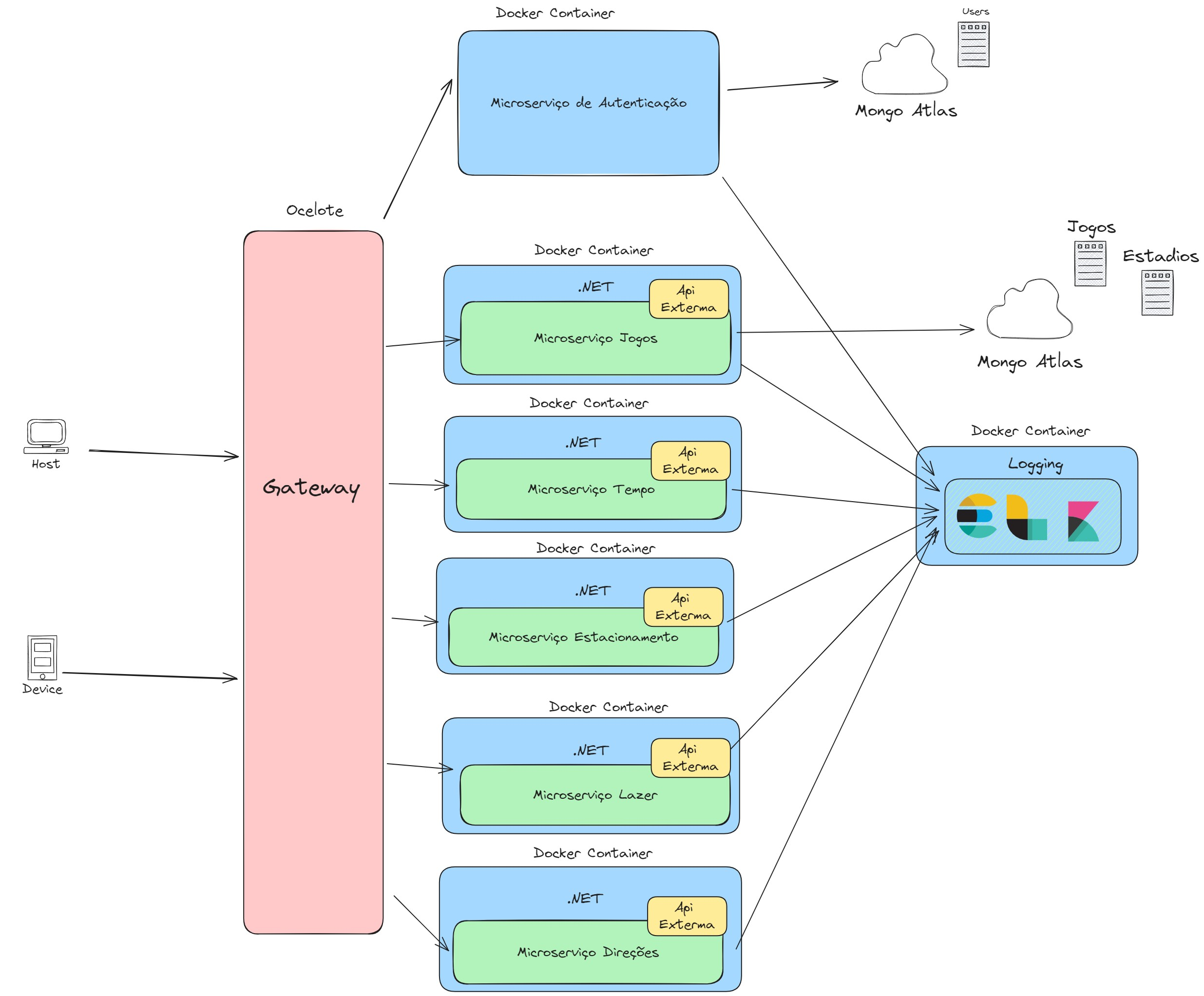
Efetuou-se o levantamento de requisitos funcionais para o *microservice* de autenticação:

|  |  |  |  |
| --- | --- | --- | --- |
| **Requisitos Funcionais** | | | |
| **ID** | **Requisito** | **Descrição** | **Prioridade** |
| RF01.1 | Gestão de Utilizadores | O sistema tem de permitir registar novos utilizadores. Campos obrigatórios:   * Login; * Password; * Nome; * Email; * Contacto; * Data Nascimento; | MUST |
| RF01.2 | O sistema deverá considerar os seguintes Tipos de Perfil:   * Utilizador; * Administrador; | MUST |
| RF01.3 | O sistema poderá permitir inativar utilizadores. | SHOULD |
| RF01.4 | O sistema poderá permitir atualizar a informação pessoal ao próprio utilizador:   * Nome; * Email; * Contacto; * Data Nascimento;   Alterações possíveis enquanto o utilizador tem a sessão ativa. | MUST |
| RF01.5 | O login deverá ser possível através de:   * Registo de utilizador, apresentando os campos:   + Login;   + Password; | MUST |
| RF01.6 | O sistema poderá disponibilizar a opção de recuperação de password por Email. | COULD |
| RF01.7 | O sistema poderá controlar o nº de tentativas de passwords erradas, 3 erros no mesmo dia e fica com o login bloqueado durante 1h. | COULD |
| RF01.8 | O sistema poderá permitir a funcionalidade de alterar password, onde deverá ser registado:   * Password Antiga; * Password Nova; * Confirmar Password; | SHOULD |
| RF01.9 | O sistema deverá ter "pré-registado" um Administrador, cuja login seja gerida pelo proprietário. | MUST |
| RF01.10 | O sistema poderá permitir associar/retirar funcionalidades do perfil. | COULD |

## Requisitos não funcionais

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requisitos Não Funcionais** | | | | |
| **ID** | **Requisito** | **Descrição** | **Prioridade** |
| RNF01 | Desempenho | O sistema deverá apresentar bom desempenho: respostas rápidas após clique e a carregar conteúdos. | MUST |
| RNF04 | Interoperabilidade | O sistema deverá comunicar com *API’s, MongoDB* *Atlas Cloud* e sistema externo de *logs*. | MUST |
| RNF05 | Confiabilidade | O sistema deverá correr sem problemas 99% do tempo, o serviço online é fundamental. | SHOULD |
| RNF06 | Idiomas | O sistema deverá suportar o idioma: Português. | MUST |
| RNF07 | Segurança e Privacidade | O sistema terá de garantir a segurança e privacidade dos dados dos utilizadores, pesquisas e ações efetuadas. | MUST |
| RNF08 | Compatibilidade | O sistema deverá ser capaz de correr nos browsers Chrome, IE e Mozilla, e dispositivos móveis. | SHOULD |
| RFN09 |  | O sistema deverá suportar a comunicação com *cloud* para consumir o *microservice* de autenticação. | SHOULD |
| RFN10 | Integração | O sistema deverá suportar a integração com as *API’s*:   * *Google Maps* para direções, lazer e estacionamento; * *API-FOOTBALL* para informações de jogos, equipas e estatísticas; * *Weather-API* para informação relativa a meteorologia. | SHOULD |

# Arquitetura do Sistema



A figura acima representa a arquitetura definida para suportar os *microservices* desenvolvidos.

A arquitetura tem como ponto de acesso uma *gateway,* quetem como objetivo:

* Servir de ponto de acesso único aos diferentes *microservices* desenvolvidos;
* Encaminhar pedidos para os *microservices* de forma correta.

Apenas o microservice de jogos utiliza uma base de dados (*MongoDB Atlas Cloud*)de forma a poder guardar informação relativa a jogos já decorridos assim como informação que não é atualizada constantemente (ex: moradas de estádios).

Todos os *microservices* irão correr dentro de *containers*, permitindo que a sua execução ocorra de forma isolada*.*

Os restantes *microservices* utilizam API’s para obter informação atualizada de acordo com o pedido do utilizador.

São utilizadas 3 API’s diferentes para disponibilizar informação aos serviços

* *API Google Maps* – Para fornecer informação relativa a direções até determinado estádio, informação sobre restaurantes/bares/cafés nas proximidades do estádio e estacionamento para o utilizador.
* *API Football* – Para fornecer detalhes das equipas, estatísticas e informação sobre o jogo a ver.
* *Weather API* – Fornece dados meteorológicos informativos ao utilizador relativamente ao momento atual e uma janela temporal até *x* horas depois.

# Pontos a salientar

## Elk Stack

É um conjunto de ferramentas que permite a análise de dados em tempo real e utilizada para gestão de *logs*. É constituída por três ferramentas principais:

* *Elastic Search*, permite armazenar, procurar e analisar grandes volumes de dados de forma rápida e em tempo real;
* *Logstash*, é uma ferramenta que obter, processar, enviar para armazenamento ou analisar, dados de várias fontes. Funciona como um intermediário que valida e garante que os dados estão formatados corretamente antes de serem enviados para armazenamento ou análise nas outras ferramentas *ELK*;
* *Kibana*, é um *UI* que permite visualizar e analisar os dados armazenados no *Elasticsearch*. Tem disponíveis vários *dashboards* e gráficos que facilitam a compreensão dos dados, análise de padrões e monitorização efetiva.

## Ocelote Gateway

Facilita a criação de *APIs* consistentes e seguras, permitindo uma gestão mais eficiente de vários pedidos e respostas entre os vários micro serviços. A inclusão de uma camada adicional entre o serviço e o cliente permite a implementação validações adicionais, controlo de acessos e ao mesmo tempo o cliente nunca comunica diretamente com os *endpoints*, facilitando assim a administração e segurança na comunicação entre diferentes micro serviços.

## xUnit – Teste unitários

São utilizados para validar individualmente partes do sistema. O objetivo é isolar e testar individualmente cada componente, de forma independente para garantir que cada parte do sistema funciona de acordo com o esperado.

# Conclusão

A realização da primeira fase deste trabalho foi importante para conhecer diferentes formas de definir a arquitetura de um sistema, o que nos levou a explorar algumas possibilidades e conceitos novos.

É um desafio muito interessante escolher a estrutura, de forma correta e seguindo os princípios fundamentais aprendidos em aula (arquitetura modular e escalável), que melhor se enquadra com o modelo de negócio que se pretende operacionalizar.

Esta primeira fase do trabalho também possibilitou a utilização de ferramentas com as quais o grupo não estava familiarizado (exemplo: *Docker, Open APIs, ELK Stack*). A utilização destas ferramentas foi um desafio que acabou por aumentar o nosso conhecimento, permitiu-nos organizar melhor o trabalho, sermos mais assertivos na resolução de problemas e agilizou o nosso processo de desenvolvimento.