Integração de Sistemas de Informação

Mario Portilho (a27989@alunos.ipca.pt)

Prof. Luis Ferreira

Índice

1. Enquadramento

O presente relatório técnico é elaborado no âmbito da Unidade Curricular de Integração de Sistemas de Informação (ISI), do curso de Engenharia de Sistemas Informáticos. O objetivo central deste trabalho prático é a conceção e implementação de uma plataforma de serviços para uma Smart City, demonstrando a capacidade de integrar sistemas heterogéneos e aplicar padrões de arquitetura modernos. A crescente complexidade dos ambientes urbanos modernos exige soluções de software que sejam não só robustas e escaláveis, mas também altamente interoperáveis. A plataforma desenvolvida, visa responder a esta necessidade através de uma arquitetura de microserviços, que permite a gestão eficiente de diversos domínios, como utilizadores, veículos, viagens, IoT e pagamentos.

2. Problema e Motivação

O desafio fundamental abordado por este projeto é a criação de uma infraestrutura digital unificada para uma Smart City. Esta infraestrutura deve ser capaz de gerir e coordenar múltiplos serviços de forma independente, garantindo a comunicação eficaz entre eles e com aplicações externas.A arquitetura de microserviços foi adotada como solução para superar as limitações de um sistema monolítico, permitindo:

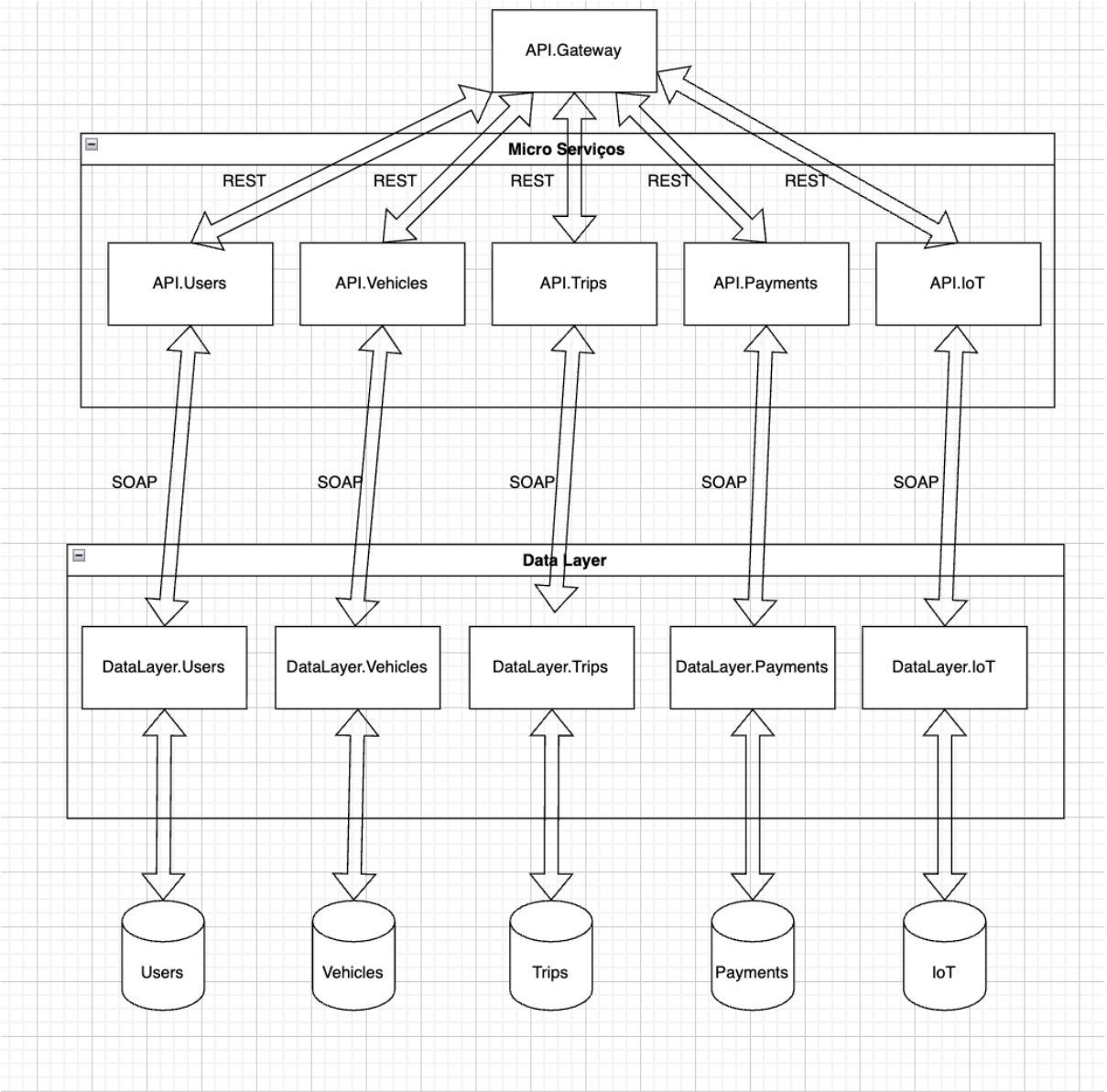
1. **Escalabilidade Independente**: Cada serviço pode ser escalado de acordo com a sua carga específica
2. **Interoperabilidade**: A utilização de diferentes protocolos (REST e SOAP) demonstra a capacidade de integrar tanto sistemas modernos como sistemas legados.
3. **Resiliência**: A falha num serviço não compromete a totalidade da plataforma.

3. Estratégia Utilizada e Arquitetura

A estratégia de desenvolvimento baseou-se na adoção de uma Arquitetura Orientada a Serviços (SOA), materializada através de microserviços. Esta abordagem é crucial para a gestão de ambientes complexos como uma Smart City, onde a agilidade e a capacidade de evolução são prioritárias.

3.1 Arquitetura de Microserviços

A plataforma Smart City está estruturada em torno de domínios de negócio bem definidos, cada um representado por um microserviço. A estrutura do repositório GitHub revela a seguinte decomposição:



Domínio de Serviço	Localização no Código	Protocolo Principal	Funcionalidade Chave
Gestão de Utilizadores	src/Services/UserManagement	RESTful	Autenticação, Registo, Gestão de Perfis.
Gestão de Viagens	src/Services/TripManagement	RESTful	Criação e monitorização de viagens.
Gestão de IoT	src/Services/IotService	RESTful	Recolha e processamento de dados de sensores.
Gestão de	src/Services/Payme	RESTful	Processamento de

Pagamentos	ntService		transações.
Dados de Veículos	src/DataLayer/SmartCity.DataLayer.VehicleService	SOAP	Contratos de serviço para dados de veículos.

Esta separação permite que cada serviço seja desenvolvido, implementado e mantido de forma autónoma, utilizando contentorização (Docker) para facilitar a implantação em ambientes de cloud.

3.2 API Gateway e Segurança

Para gerir o tráfego de entrada e centralizar a segurança, o projeto utiliza um API Gateway, localizado em src/Services/Gateway/SmartCity.ApiGateway. O Gateway desempenha funções essenciais:

- **Roteamento**: Encaminha os pedidos dos clientes para o microserviço apropriado.
- **Agregação**: Combina respostas de múltiplos serviços, se necessário.
- **Segurança Centralizada**: Aplica políticas de autenticação e autorização antes de o pedido chegar aos serviços internos.

A segurança é implementada através do controlo de execução com credenciais, utilizando o padrão OAuth 2.0 e JSON Web Tokens (JWT). O serviço UserManagement é responsável pela emissão dos tokens, que são validados pelo API Gateway, garantindo que apenas pedidos autenticados e autorizados acedem aos microserviços.

3.3 Dualidade de Protocolos: REST e SOAP

Um dos requisitos cruciais do trabalho é a demonstração da integração de sistemas com diferentes protocolos. O projeto Smart City utiliza uma abordagem híbrida:

Protocolo	Serviço Exemplo	Justificação
RESTful	UserManagement, TripManagement	Ideal para serviços modernos, públicos e com alta frequência de acesso. Utiliza o protocolo HTTP de forma leve e sem estado, sendo facilmente consumível por aplicações web e móveis.
SOAP	SmartCity.DataLayer.VehicleService	Utilizado na camada de dados (DataLayer), que simula a integração com um

		sistema legada ou um serviço de dados mais formal.

Esta dualidade assegura que a plataforma é capaz de se integrar tanto com as APIs mais recentes (REST) como com sistemas empresariais mais antigos que dependem de contratos de serviço estritos (SOAP).

4. Implementação e Detalhes Técnicos

A implementação do projeto foi realizada em C# utilizando a framework .NET 8, uma escolha que garante alto desempenho e um ecossistema robusto para o desenvolvimento de microserviços.

4.1 Serviços RESTful (User Management)

O serviço de gestão de utilizadores (SmartCity.UserManagement.API) é um exemplo claro de um serviço RESTful bem estruturado:

- **Controladores**: O ficheiro UsersController.cs
expõe endpoints HTTP para operações CRUD sobre recursos de utilizadores (ex: GET /api/users, POST /api/users).
- **Autenticação**: O AuthController.cs
lida com o processo de login e a emissão de tokens JWT, essenciais para o controlo de acesso.
- **Documentação**: A utilização de frameworks como o Swagger (OpenAPI) fundamental para documentar automaticamente os endpoints RESTful, facilitando o consumo por parte das aplicações cliente.

4.2 Serviços SOAP (Vehicle Data Service)

O serviço de dados de veículos (SmartCity.DataLayer.VehicleService) demonstra a implementação de um serviço SOAP.

- **Contrato de Serviço**: A interface IVehicleDataService.cs
define o contrato de serviço com métodos como GetVehicleByIdAsync e UpdateVehicleLocationAsync, decorados com o atributo [OperationContract].
- **Comunicação**: A utilização de System.ServiceModel indica que a comunicação é baseada em mensagens XML, conforme o padrão SOAP, o que é ideal para transações que exigem garantias de entrega e contratos de serviço formais.

4.3 Integração de Serviços Externos

Foram integradas APIs externos como Open-Meteo para fornecimento de dados meteorológicos em tempo real

Deploy

Para dar deploy basta seguir estes comandos:

1. git clone https://github.com/portilho13/smartcity
2. cd smartcity/scripts
3. docker-compose up –build -d

5. Conclusão

O Trabalho Prático II de Integração de Sistemas de Informação resultou na conceção e implementação de uma plataforma de serviços para uma Smart City, cumprindo integralmente os requisitos do enunciado. A adoção da arquitetura de microserviços em .NET 8, combinada com a utilização estratégica de um API Gateway, provou ser uma solução robusta e escalável.

A capacidade de integrar diferentes protocolos, nomeadamente RESTful para a maioria dos serviços de negócio e SOAP para a camada de dados de veículos e outros serviços, demonstra uma compreensão aprofundada dos desafios de interoperabilidade em ambientes empresariais. A segurança, garantida pela autenticação OAuth/JWT, e a documentação via Swagger asseguram que a plataforma está pronta para ser consumida por aplicações cliente e para ser implementada em ambiente de cloud.

O projeto serve como uma demonstração prática e abrangente dos conceitos de integração de sistemas, arquitetura de software e desenvolvimento de serviços modernos.

6. Bibliografia

portilho13. (2025). smartcity. GitHub. Disponível em: `
[2] ESI-ISI2025-26-TP2-enunciado.pdf. (2025). Enunciado do Trabalho Prático II.
[3] Microsoft. (2025). .NET Documentation. Disponível em: `
[4] W3C. (2007). SOAP Version 1.2 Part 1: Messaging Framework. Disponível em: `
[5] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine].