INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

28-12-2025

Trabalho Prático II

Integração de Sistemas de Informação

José Miguel Oliveira Pires
27968 – LESI

Índice

Índice de Ilustrações

Introdução

Este trabalho prático, realizado no âmbito da unidade curricular de Integração de Sistemas de Informação, tem como objetivo principal o desenvolvimento de processos de interoperabilidade entre sistemas baseados em serviços web.

Projeto: "SmartEstate – Plataforma Integrada de Gestão Imobiliária"

O projeto consiste no desenvolvimento de uma solução de integração de sistemas para o setor Imobiliário. O sistema visa modernizar a gestão de mediação (imóveis, proprietários, clientes e visitas), substituindo processos manuais por um ecossistema digital interoperável.

A solução focar-se-á na exposição de uma API de serviços robusta que permita a comunicação fluida entre o repositório central da imobiliária, aplicações clientes e serviços externos de geo-localização e conversão financeira.

Enquadramento

O setor imobiliário encontra-se numa fase acelerada de transformação digital, onde a rapidez e a precisão da informação são fatores críticos de sucesso. Atualmente, o mercado caracteriza-se pela fragmentação de ferramentas: as agências utilizam sistemas de gestão interna (CRM) que raramente comunicam com as aplicações de produtividade pessoal (agendas, mapas, conversores) utilizadas por clientes e mediadores.

Esta falta de integração resulta em "silos de informação", obrigando à duplicação manual de dados e aumentando o risco de erros, como falhas em compromissos ou inconsistências na localização de ativos. Neste contexto, o conceito de "Smart Real Estate" evolui da mera domótica para a Gestão Inteligente de Dados. O foco desloca-se para a capacidade de um sistema fornecer contexto geográfico imediato (Location Intelligence) e sincronização temporal universal.

Este projeto enquadra-se na necessidade técnica de garantir a interoperabilidade total entre estas diferentes plataformas. Através de uma arquitetura orientada a serviços (SOA) e alojamento na Cloud, pretende-se criar um ecossistema onde o repositório central da imobiliária atue não apenas como um arquivo, mas como um hub de serviços que alimenta automaticamente mapas interativos e agendas digitais, utilizando normas standard de mercado como SOAP e REST.

.

Constituição do Grupo

Em conformidade com o modelo de funcionamento estipulado para a Unidade Curricular, este trabalho prático será realizado em regime individual.

Nome: Miguel Pires

Número de aluno: 27968

Curso: Licenciatura em Engenharia de Sistemas Informáticos

Descrição do Problema

No setor imobiliário atual, as agências enfrentam dificuldades operacionais que resultam em perda de oportunidades de venda. Identificam-se três problemas críticos que este projeto visa resolver

1. **Ineficiência no Agendamento:** A falta de padronização na marcação de visitas resulta em falhas de comunicação e ausência de sincronização com as agendas pessoais (smartphones) de clientes e mediadores.
2. **Falta de Contexto Geográfico:** A maioria dos sistemas regista apenas a morada em texto, dificultando a visualização espacial da carteira de imóveis e o cálculo de distâncias para os clientes.
3. **Barreiras Internacionais:** A rigidez na apresentação de preços (apenas na moeda local) cria atrito na comunicação com investidores estrangeiros.

Este projeto propõe resolver estes problemas através de uma arquitetura orientada a serviços que permite:

- Gerir o ciclo de vida da venda (Entidades: Proprietário, Imóvel, Visita, Venda, Cliente) de forma centralizada.
- **Promover a interoperabilidade de agendamentos:** Implementação de exportação standard .ics para integração automática em calendários (Outlook, Google, iOS).
- **Enriquecimento de Dados:** Utilização de serviços externos para geocoding automático e conversão monetária em tempo real.

Arquitetura prevista para a solução

A solução será desenvolvida sobre a plataforma .NET (C#), adotando uma arquitetura SOA (Service-Oriented Architecture) e será alojada na Cloud (Microsoft Azure).

A estrutura lógica divide-se nas seguintes camadas:

A. Data Layer (Serviços SOAP)

- Implementação de serviços SOAP para a camada de acesso a dados (*Data Access Layer*).
- Esta camada é responsável pelas transações ACID e gestão das entidades relacionais críticas (Owner, Property, Client, Visit, Sale). O uso de SOAP assegura contratos formais para a integridade do repositório.

B. Integration Layer (Arquitetura de Microserviços REST) A lógica de negócio será exposta via **REST**, seguindo uma abordagem de microserviços para cumprir os requisitos de modularidade e segurança:

1. **Microserviço de Autenticação (Auth Service):** Serviço isolado responsável pela gestão de identidade. Valida credenciais e emite Tokens JWT (JSON Web Tokens), garantindo que a segurança está desacoplada da lógica de negócio.
2. **Microserviço Core (SmartEstate API):** API principal que gere o negócio imobiliário. Valida os tokens JWT e disponibiliza as operações CRUD e o endpoint de exportação .ics (iCalendar) para a interoperabilidade de agendas.
3. **Documentação:** Toda a API será documentada utilizando o standard Open API (Swagger).

C. Integração de Serviços Externos O sistema consumirá APIs externas para enriquecimento de dados e valorização do serviço:

1. **OpenStreetMap API:** Consumido no *backend* para serviços de *Geocoding*. Ao registar um imóvel, o sistema converte a morada em coordenadas (Latitude/Longitude) para armazenamento estruturado e posterior visualização em mapas.
2. **Exchange Rates API:** Para conversão de preços de venda (FinalPrice) em tempo real, permitindo apresentar valores em múltiplas moedas a clientes internacionais.

D. Aplicações Cliente e Infraestrutura

- Desenvolvimento de uma Aplicação Web (ASP.NET Core MVC). Esta escolha permite a fácil integração de bibliotecas de mapas (LeafletJS) para visualização dos imóveis georreferenciados
- O repositório de dados será alojado na Cloud, utilizando Microsoft SQL Server (Azure SQL Database).

IPCA INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE ESCOLA SUPERIOR DE TECNOLOGIA

Arquitetura prevista – Diagrama de Componentes (continuação)

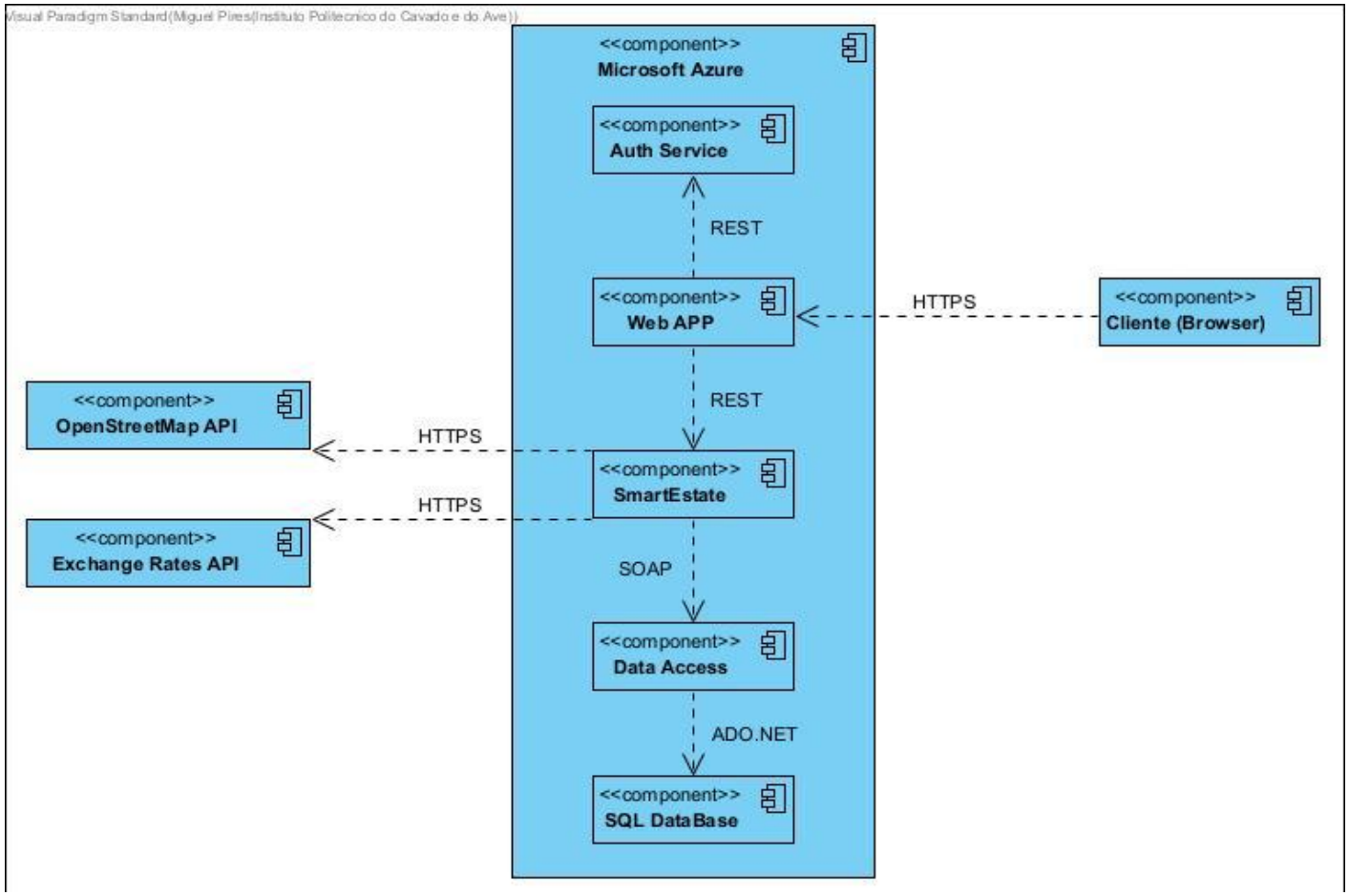

Ilustração 1 - Diagrama de Componentes

Arquitetura Prevista - Diagrama Entidade Relação (ER)

Ilustração 2 - Diagrama ER

Implementação e Integração na Cloud

A. Data Layer (Serviços SOAP)

A base da solução assenta numa arquitetura de dados robusta, desenhada para suportar a interoperabilidade entre sistemas. A implementação foi dividida em duas componentes fundamentais: a definição de estruturas de dados transportáveis (*Models*) e a formalização dos comportamentos (*Contracts*).

1. Modelação de Dados e Serialização (Models)

Foram criadas interfaces específicas para cada entidade de negócio, decoradas com atributos WCF (*Windows Communication Foundation*) para garantir a interoperabilidade e a possibilidade de exposição futura via SOAP.

- **Entidades do sistema**: Owner, Client, Property, Visit, Sale, User
- **Data Contracts:** Todas as classes principais foram decoradas com o atributo [DataContract], e as suas propriedades com [DataMember]. Isto garante que os objetos podem ser serializados (convertidos) automaticamente para XML ou JSON, permitindo que a API envie dados complexos para o Frontend ou para outros serviços sem problemas de compatibilidade.

2. Definição Formal de Serviços (Contracts)

Para garantir o desacoplamento entre a definição do serviço e a sua implementação lógica, foram utilizadas Interfaces C#.

- **Service Contracts:** Foram criadas interfaces granulares (IProperty, IUser, etc.) decoradas com [ServiceContract], que expõem os métodos disponíveis (ex: AddProperty, Login) como operações de serviço ([OperationContract]).

- **Padrão de Agregação:** Implementou-se a interface IImobiliario que herda de todas as outras interfaces. Esta abordagem permite instanciar um único ponto de acesso à base de dados na API, mantendo o código organizado e modular.

B. Integration Layer (Arquitetura de Microserviços REST)

A lógica de negócio foi exposta através de uma API RESTful desenvolvida em **ASP.NET Web API**, atuando como a ponte entre a camada de dados e as aplicações clientes.

1. Arquitetura REST e Controladores

A API foi estruturada em Controladores (Controllers) que agupam operações por entidade. Tomando como exemplo o PropertyController, a implementação segue estritamente os princípios REST:

- **Mapeamento de Verbos HTTP:** As operações CRUD da camada de dados foram mapeadas para os verbos HTTP adequados: GET (Leitura), POST (Criação), PUT (Atualização) e DELETE (Remoção).

- **Consumo da Data Layer:** Os controladores instanciam diretamente a classe Imobiliario da Data Layer, garantindo que toda a lógica de acesso a dados permanece encapsulada e não é duplicada na API.

2. Documentação automática (Swagger)

Para cumprir o requisito de documentação e facilitar os testes, foi integrada a biblioteca **Swashbuckle**.

- Esta ferramenta gera automaticamente uma interface interativa (Swagger UI) ao ler os controladores e modelos da API.
- Isto permitiu testar os *endpoints* diretamente no browser (via /swagger) sem necessidade de escrever código de Frontend numa fase inicial, acelerando o ciclo de desenvolvimento e depuração.

C. Integração de Serviços Externos

Para enriquecer a experiência do utilizador e cumprir o requisito de *Location Intelligence*, o sistema integra serviços de terceiros para geocodificação e visualização de mapas.

1. Serviço de Geocodificação (Backend)

Foi implementado um serviço dedicado (ToolsController) que consome a API do **Nominatim (OpenStreetMap)**. O objetivo é converter moradas textuais em coordenadas geográficas (Latitude e Longitude) de forma automática.

- **Implementação Técnica:** A integração foi realizada via HttpClient na plataforma .NET.
- **Fluxo de Dados:** O serviço recebe uma *string* (ex: "Avenida da Liberdade, Barcelos"), contacta a API externa e processa a resposta JSON (JArray), extraindo apenas as coordenadas necessárias para persistência na base de dados.

D. Aplicações Cliente e Infraestrutura

A aplicação cliente foi desenvolvida seguindo uma abordagem de *Single Page Application* (SPA) simplificada, utilizando HTML5, CSS3 e JavaScript. Esta camada é totalmente independente da lógica de negócio, comunicando exclusivamente através de pedidos HTTP assíncronos.

1. Comunicação Assíncrona e Fetch API

Toda a interação com o Backend é realizada via fetch(). O código JavaScript foi estruturado para suportar a migração para a Cloud:

- **Configuração de Endpoints:** Foi definida uma constante global API_URL apontando para o serviço Azure (https://imobiliario-api-pires...), permitindo que o Frontend (que corre no browser do cliente) consuma dados da API alojada na nuvem.
- **Tratamento de Dados:** As respostas JSON da API são processadas assincronamente (async/await) para atualizar o DOM (Document Object Model) sem recarregar a página, proporcionando uma experiência de navegação fluida.

2. Segurança e Gestão de Sessão no Cliente

Embora o Frontend não consiga "proteger" dados (isso é responsabilidade da API), ele implementa a gestão da sessão do utilizador:

- **Armazenamento de Token:** Após o login, o *Token JWT* recebido é armazenado em localStorage.
- **Injeção de Cabeçalhos:** Em cada pedido protegido (como CreateProperty), o JavaScript recupera este token e injeta-o automaticamente no cabeçalho Authorization: Bearer, permitindo à API validar a identidade do utilizador.
- **Interface Adaptativa:** O sistema verifica a *role* do utilizador (Cliente ou Proprietário) guardada localmente e ajusta a interface, melhorando a usabilidade e segurança visual.

E. Publicação e Configuração do Ambiente de Produção (Azure)

A etapa final do projeto consistiu na disponibilização da solução completa na Cloud, utilizando o ecossistema **Microsoft Azure**. O processo de *deployment* foi estruturado para garantir que tanto a base de dados, como a API e o Frontend ficassem imediatamente acessíveis e funcionais.

1. Infraestrutura de Dados (Azure SQL Server)

Para suportar a persistência de dados na nuvem, foi criada uma infraestrutura dedicada:

- **Criação do Servidor:** Foi provisionado um **Servidor SQL Lógico** no Azure (imobiliarioserver-pires...) para centralizar a gestão de acessos e segurança.
- **Base de Dados:** A base de dados relacional foi implantada neste servidor. A *Connection String* da API foi atualizada para apontar para esta nova instância, garantindo que todas as operações (registos, logins, visitas) ficam gravadas na nuvem e não na máquina local.

2. Publicação da API

A camada de Backend foi publicada num **Azure App Service** independente.

- **Exposição Imediata:** A configuração foi ajustada para que a documentação **Swagger UI** ficasse disponível logo no arranque. Isto permite que qualquer utilizador (ou o professor) aceda ao URL da API e veja imediatamente a lista de *endpoints* disponíveis, facilitando testes rápidos.

3. Aplicação Web e Experiência de Utilizador

O Frontend foi publicado num serviço web distinto, com foco na usabilidade imediata:

- **Encaminhamento para Login:** Foi configurado o documento predefinido (*Default Document*) do servidor web. Desta forma, quando se acede ao link principal do site, o sistema encaminha o utilizador automaticamente para a página de **Login** (Login.html). Isto elimina a necessidade de o utilizador conhecer o caminho específico dos ficheiros, oferecendo uma experiência de entrada fluida e profissional assim que o site é aberto.

Postman – API Teste

Para garantir a robustez e o correto funcionamento da Web API desenvolvida, foi realizado testes automatizados utilizando o *Collection Runner* do Postman. O objetivo foi simular um fluxo completo de utilização real do sistema, desde o registo do utilizador até à concretização de uma venda.

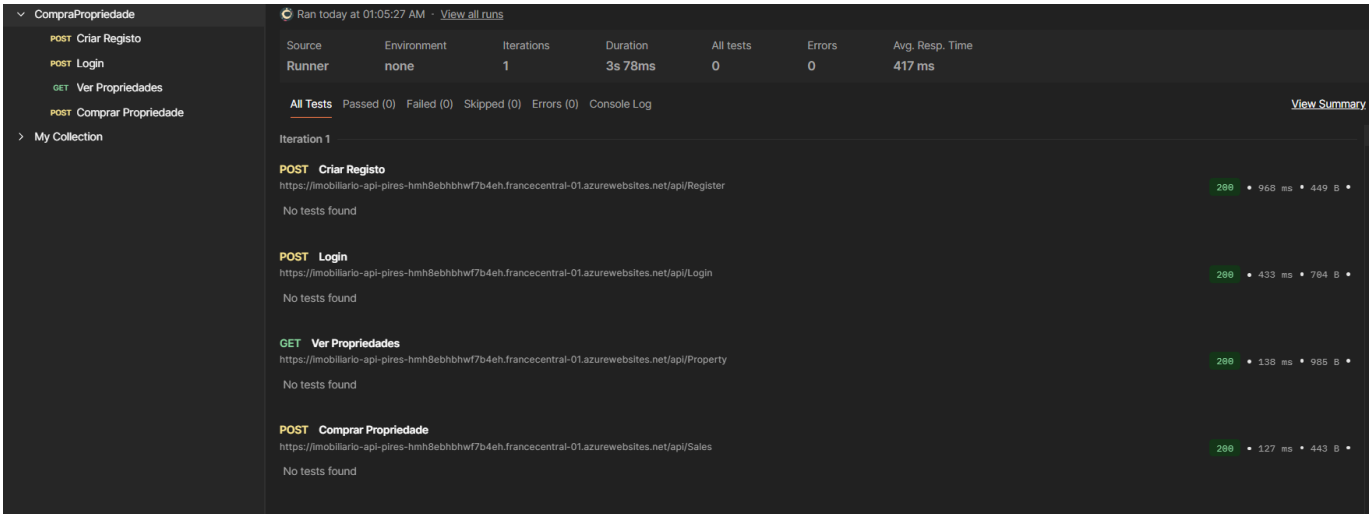

Ilustração 3 - Testes Postman

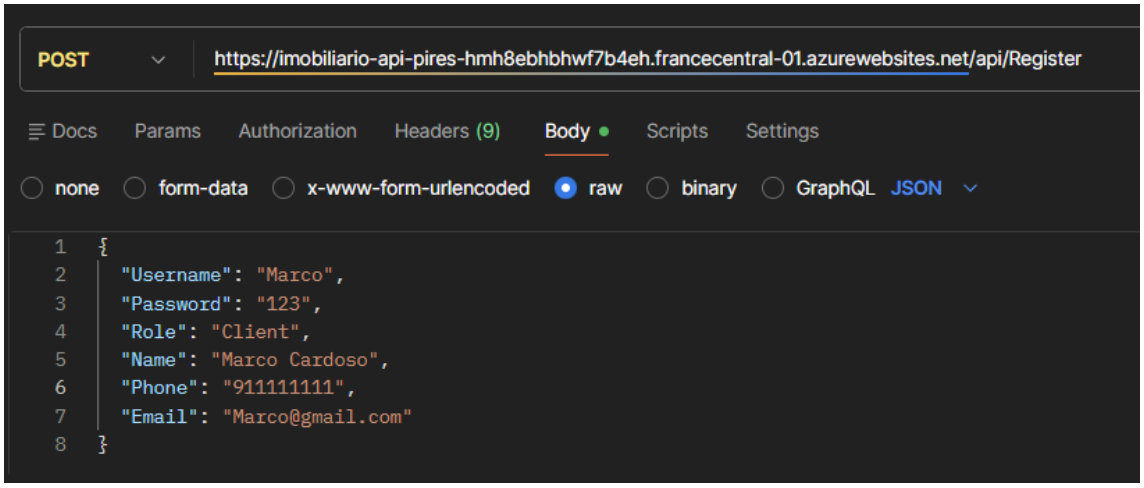

Ilustração 4 - Postman Registar

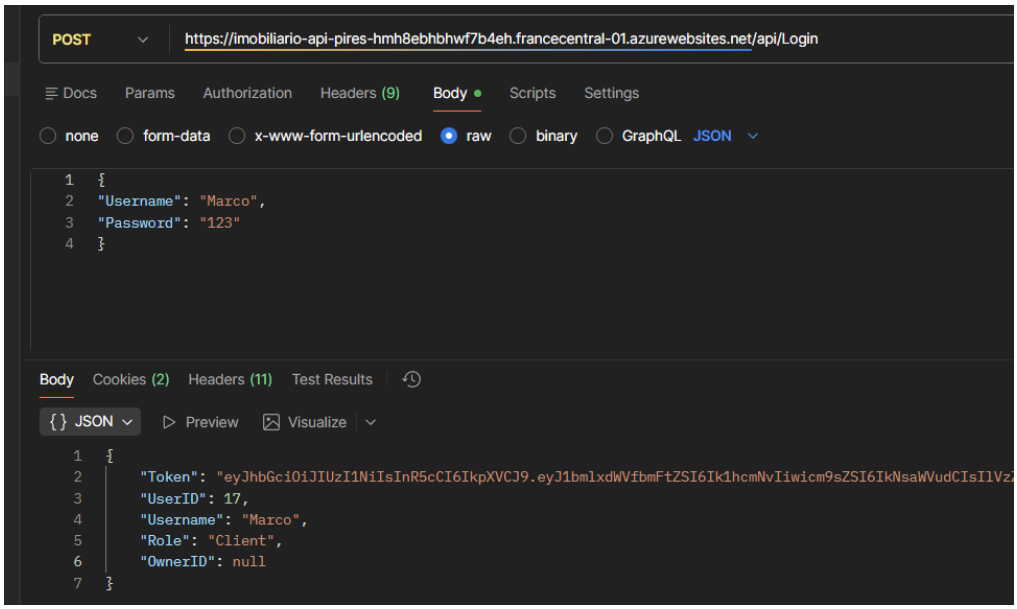

Ilustração 5 - Postman Login

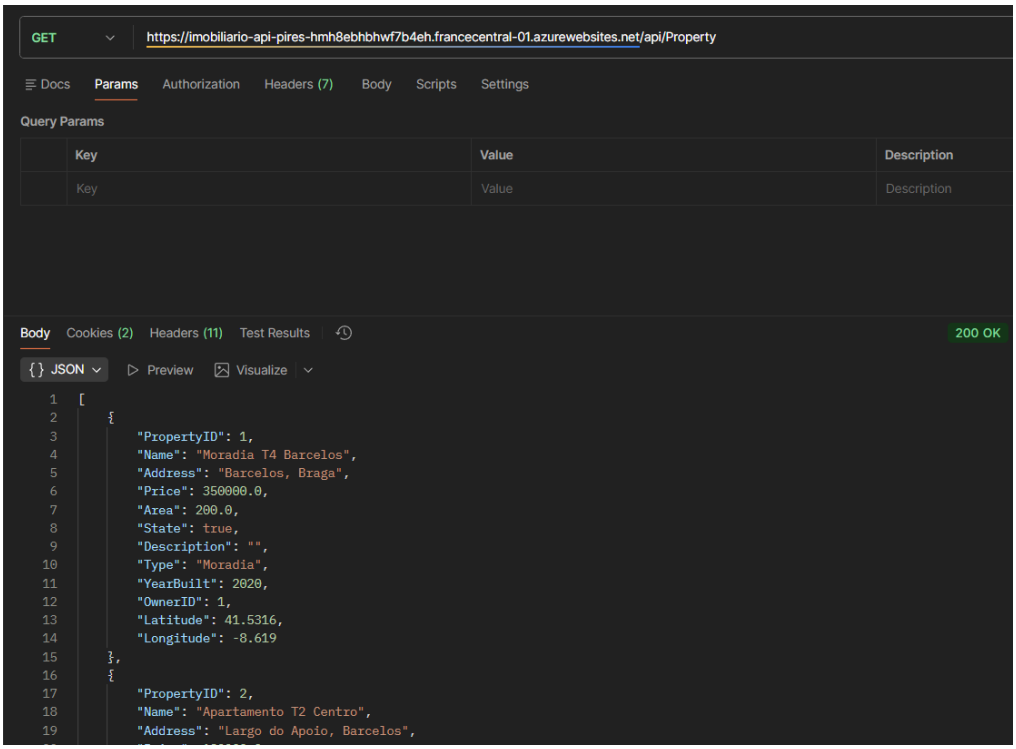

Ilustração 6 - Postman Ver Propriedades

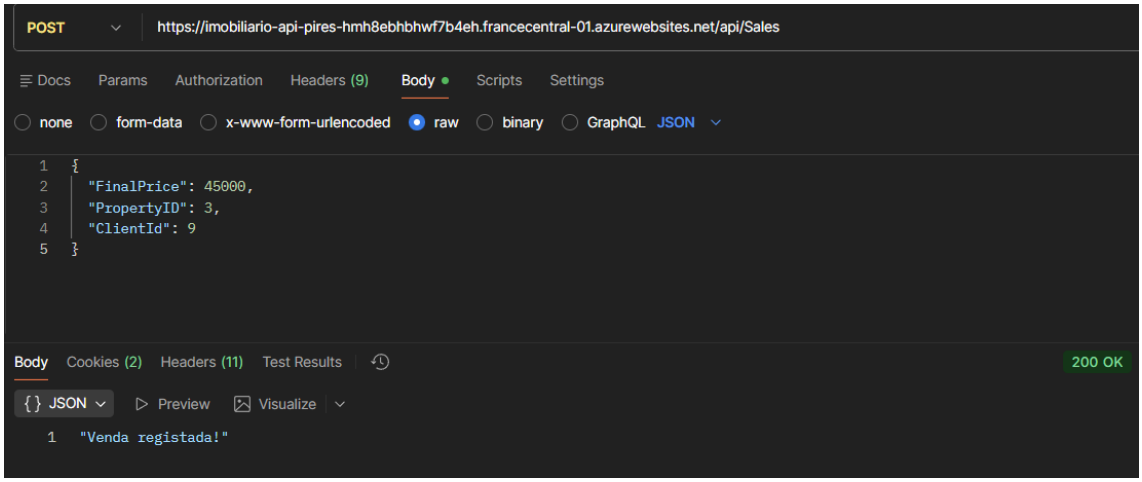

Ilustração 7 - Postman Comprar Propriedade

Projeto - Links

A solução desenvolvida adota uma arquitetura distribuída e encontra-se integralmente alojada na infraestrutura *cloud* do **Microsoft Azure**. Esta abordagem elimina a necessidade de configurações locais por parte do utilizador e garante a disponibilidade contínua dos serviços.

O sistema está dividido em dois componentes principais, acessíveis publicamente através dos seguintes endereços:

- **Aplicação Web (Frontend):** Interface gráfica para gestão de clientes e vendas.
  - https://imobiliario-web-pires-drf2ayamdvfea4bg.francecentral-01.azurewebsites.net
- **Web API (Backend):** Serviço de dados RESTful ligado à base de dados SQL Azure.
  - https://imobiliario-api-pires-hmh8ebhbhwf7b4eh.francecentral-01.azurewebsites.net

Conclusão

O desenvolvimento do projeto "SmartEstate" permitiu alcançar com sucesso os objetivos propostos para a Unidade Curricular, resultando na implementação de um ecossistema de gestão imobiliária funcional, interoperável e alojado na Cloud.

A adoção de uma **Arquitetura Orientada a Serviços (SOA)** provou ser fundamental para o sucesso da solução. A separação estrita entre a Camada de Dados (suportada por contratos formais e *Data Members*), a API REST (documentada automaticamente via Swagger) e o Frontend (SPA assíncrona) garantiu a modularidade do sistema. A integração de APIs externas, especificamente o serviço de geocodificação *Nominatim*, valorizou o produto final, permitindo transformar dados estáticos de moradas em inteligência de localização visual através de mapas interativos.

A fase de migração para o **Microsoft Azure** constituiu o desafio técnico mais enriquecedor do projeto. A configuração de um ambiente de produção real — que envolveu a criação de servidores SQL lógicos, a gestão de regras de segurança na Firewall e a correta parametrização dos *App Services* — permitiu consolidar competências práticas de *Cloud* . O resultado é um sistema distribuído onde a API e a Base de Dados comunicam de forma segura na nuvem, e onde a aplicação Web oferece uma experiência de acesso imediato e intuitivo ao utilizador final.

Em suma, o projeto cumpriu os requisitos de integração de sistemas, entregando uma plataforma capaz de gerir o ciclo de vida imobiliário (do registo à venda) num ambiente web moderno e acessível globalmente.