



VinhaTech

Integração de Sistemas de Informação

Aluno: 23010 – Hugo Cruz

Aluno: 23016 – Dani Cruz

Professor: Óscar Ribeiro

Licenciatura em Engenharia de Sistemas Informáticos

Índice

1	Introdução.....	5
2	Arquitetura de Integração e Tecnologias	6
2.1	Visão Global da Integração	6
2.2	Diagrama ER.....	7
3	Implementação de Serviços.....	8
3.1	API Gateway (RESTful).....	9
3.2	Serviço SOAP Interno	10
3.3	Serviço SOAP Externo	10
3.4	Módulo IoT e Tempo Real	11
3.5	Base de Dados.....	11
4	Cenários de Teste e Validação	13
4.1	Cenário A: Monitorização de Alertas	13
4.2	Cenário B: Fluxo de Compra e Integração de Pagamentos	14
4.3	Cenário C: Gestão de Acessos	14
5	Conclusão.....	15

Índice de Figuras

Figura 1 - Arquitetura Orientada a Serviços (SOA)	6
Figura 2 - Diagrama ER	7
Figura 3 - Infraestrutura de Alojamento.....	8
Figura 4 - Interface Swagger.....	9
Figura 5 - Integração com SOAP Interno	10
Figura 6 - Interface de descrição do serviço SOAP externo.....	10
Figura 7 - Fluxo de dados IoT e propagação de alertas	11
Figura 8 - Painel de administração da plataforma Auth0.....	12
Figura 9 - Dashboard dos Valores do Sensor	13
Figura 10 - Simulação de envio de uma leitura crítica.....	13
Figura 11 - Receção do Alerta	13
Figura 12 - Stock Inicial	14
Figura 13 - Simulação do Pagamento.....	14
Figura 14 - Stock Final.....	14
Figura 16 - Conta "Owner"	14
Figura 15 - Conta "Cliente"	14

1 Introdução

Este relatório apresenta a segunda fase do projeto desenvolvido no âmbito da Unidade Curricular de Integração de Sistemas de Informação. Enquanto a etapa inicial se focou na definição do problema e no desenho da arquitetura conceitual, esta dedica-se assim à implementação técnica e ao funcionamento dos serviços que compõem a nossa solução.

Sabendo que a preservação de vinhos exige condições ambientais rigorosamente controladas, uma vez que fatores como a **temperatura**, a **humidade** e a **luminosidade** influenciam diretamente a sua qualidade e longevidade. Neste contexto, surgiu a necessidade de desenvolver um sistema capaz de monitorizar continuamente estas variáveis, garantindo que o vinho é armazenado em condições adequadas ao longo de todo o seu ciclo de vida.

Ao longo deste documento, demonstramos como articulamos uma **API REST** desenvolvida em **ASP.NET Core**, um **Frontend em React**, serviços **SOAP** (tanto internos como externos) e componentes **IoT**.

Serão detalhados os desafios encontrados durante a implementação, a lógica de comunicação estabelecida entre as diferentes camadas e, por fim, os cenários de teste que validam a robustez da solução.

2 Arquitetura de Integração e Tecnologias

2.1 Visão Global da Integração

A arquitetura da solução segue um modelo orientado a serviços (**SOA**) híbrido, desenhado para maximizar a segurança e a capacidade de integração. Uma das decisões estruturais mais importantes foi a **separação** entre as interfaces públicas (Frontend e API) e o acesso aos dados sensíveis.

A integração do sistema ocorre em três níveis fundamentais. Primeiramente, a comunicação entre o Frontend (React) ou a Aplicação IoT e a API REST é realizada através de **JSON sobre HTTP**.

Em segundo lugar, para garantir o isolamento da base de dados, a API não acede diretamente aos dados. Em vez disso, comunica com o **SOAP interno (ASMX)** via **XML**. Por último, o sistema consome um serviço **SOAP externo**, disponibilizado por terceiros, para efetuar a validação de cartões de crédito.

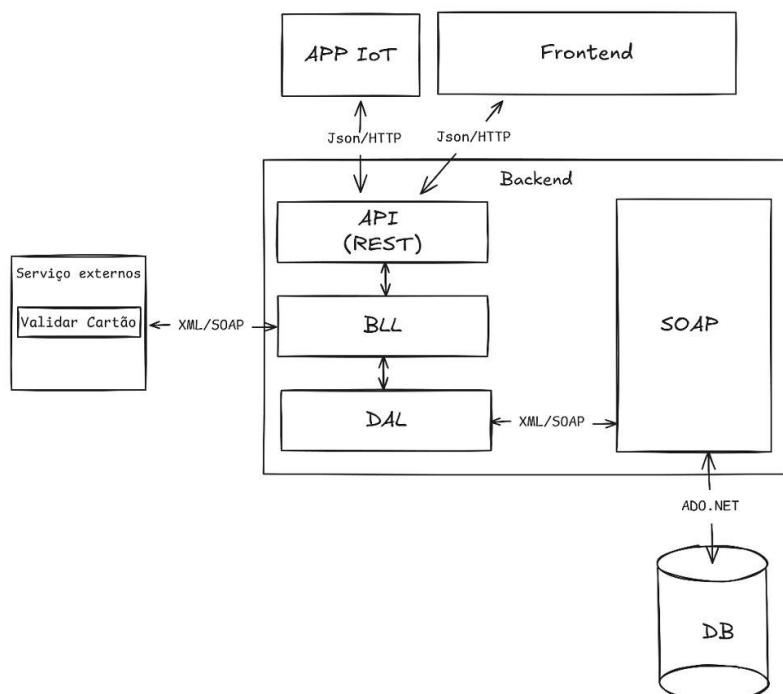


Figura 1 - Arquitetura Orientada a Serviços (SOA)

2.2 Diagrama ER

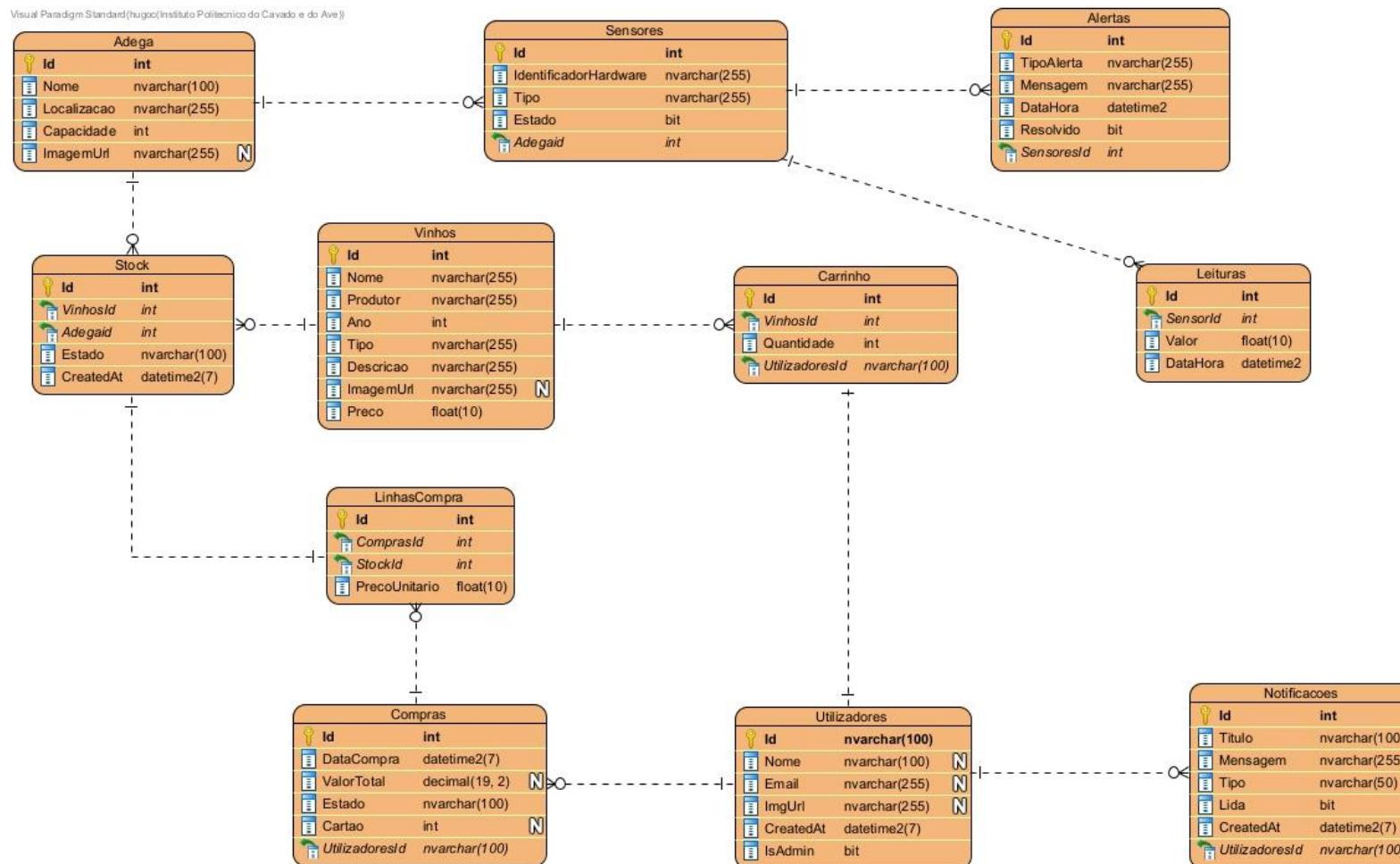


Figura 2 - Diagrama ER

3 Implementação de Serviços

3.1 Infraestrutura de Alojamento

A arquitetura de implantação da solução, ilustrada na **Figura 3**, foi desenhada sobre a plataforma **Microsoft Azure** com foco na segurança e isolamento de recursos.

O diagrama evidencia a separação entre a **Zona Pública**, onde residem o Frontend (*Static Web App*) e a API (*App Service*) e a **Zona Privada**. Nesta última, o Serviço SOAP e a Base de Dados SQL encontram-se protegidos numa rede virtual (**VNet**), acessíveis apenas internamente através de **Private Endpoints**.

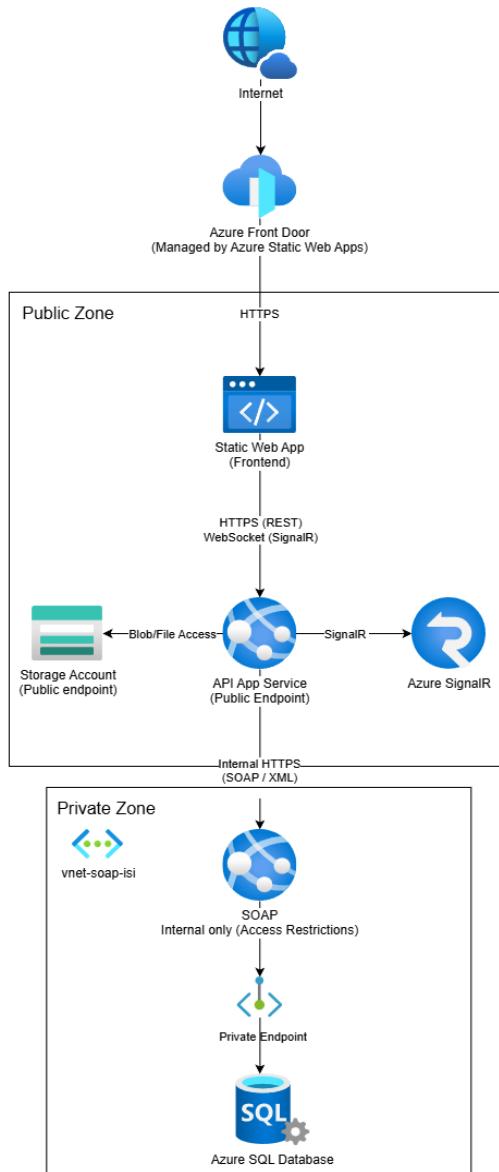


Figura 3 - Infraestrutura de Alojamento

3.2 API Gateway (RESTful)

A API, desenvolvida em **ASP.NET Core**, que funciona como ponto central de comunicação entre o frontend e o SOAP. Este componente não contém lógica direta de acesso à base de dados, focando-se exclusivamente na validação dos dados recebidos, na autenticação dos utilizadores e no encaminhamento dos pedidos para os serviços adequados.

Os endpoints encontram-se organizados por áreas funcionais, nomeadamente gestão de vinhos, adegas, sensores, stock, compras, utilizadores e notificações. Todas as rotas da API estão devidamente documentadas através da ferramenta **Swagger (OpenAPI)**, permitindo a sua consulta e teste de forma interativa.

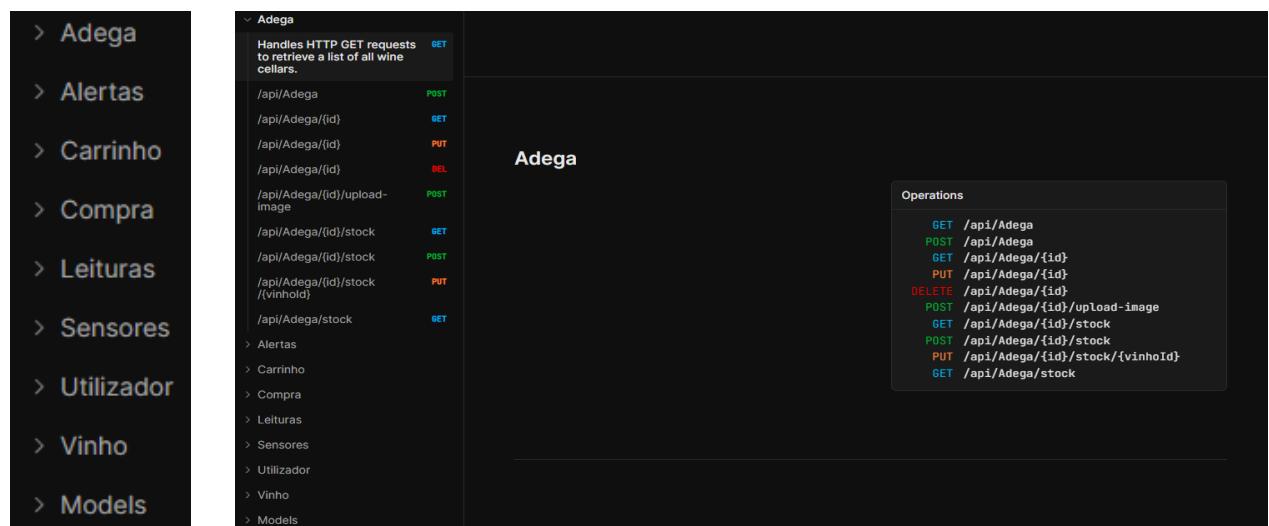


Figura 4 - Interface Swagger

3.3 Serviço SOAP Interno

O sistema utiliza serviços SOAP em dois contextos distintos.

O **serviço SOAP interno** é responsável exclusivamente pelo acesso à base de dados, encapsulando todas as operações de persistência e consulta de informação. Este serviço garante o **isolamento da camada de dados** e reforça a segurança do sistema, sendo consumido apenas pela API RESTful.

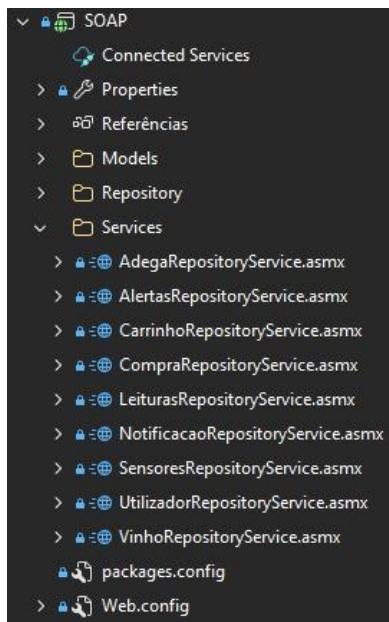


Figura 5 - Integração com SOAP Interno

3.4 Serviço SOAP Externo

Adicionalmente, o sistema integra um serviço **SOAP externo** para validação de cartões de crédito, utilizado durante o processo de pagamento no módulo de e-commerce. Esta integração permite simular um cenário real de transações eletrónicas, assegurando que apenas pagamentos válidos são processados. A nossa validação apenas utiliza o “GetCardType”, pois estávamos com problemas para a utilização dos outros.

CreditCardValidator

Credit Card Validation and Card Type Web Service

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [GetCardType](#)
Returns the card issuer Visa, MasterCard, AMEX, etc., based on the card number.

Figura 6 - Interface de descrição do serviço SOAP externo

3.5 Módulo IoT e Tempo Real

A monitorização ambiental constitui o núcleo funcional do projeto. O fluxo de informação foi desenhado para ser **contínuo** e **imediato**. Assim que um sensor envia uma leitura via **POST** para a API, a camada de lógica de negócio analisa se o valor recebido excede os limites definidos para aquela adega específica.

Caso se verifique uma anomalia, o sistema gera um alerta que é difundido instantaneamente via **WebSockets** para o Frontend. Isto permite que o proprietário da adega visualize o problema no dashboard sem necessidade de recarregar a página, garantindo uma capacidade de resposta rápida.

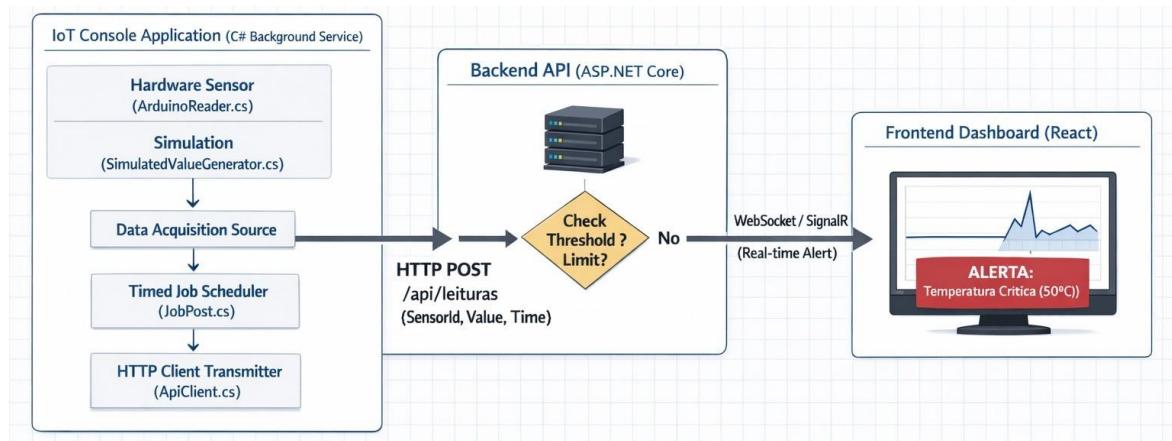


Figura 7 - Fluxo de dados IoT e propagação de alertas

3.6 Base de Dados

A base de dados utilizada é de natureza relacional e armazena toda a informação necessária ao funcionamento do sistema. Entre os principais dados armazenados encontram-se as adegas, sensores, leituras ambientais, vinhos, stock, utilizadores, compras e notificações.

O acesso à base de dados é realizado **exclusivamente através do serviço SOAP interno**, não sendo permitido qualquer acesso direto por parte da API REST ou do frontend. Esta abordagem contribui para uma maior integridade e segurança da informação.

3.7 Autenticação Auth0

Para garantir um nível elevado de segurança e evitar a complexidade de gestão de credenciais sensíveis na base de dados local, a solução não recorre ao método tradicional de ASP.NET Identity (via Entity Framework). Em sua substituição, foi integrado o **Auth0**, uma plataforma dedicada à gestão de identidades.

Desta forma, a autenticação é **realizada diretamente** na plataforma Auth0. Quando um utilizador inicia sessão no Frontend (React), é redirecionado para a interface de login do Auth0. Após o sucesso da operação, o sistema devolve um **token** seguro (**JWT - JSON Web Token**).

Assim, a API REST limita-se a validar a assinatura desse *token* a cada pedido, extraíndo o Auth0UserId para identificar o utilizador e o seu respetivo perfil (**Owner** ou **Cliente**), sem que haja qualquer armazenamento ou processamento local de palavras-passe.

The screenshot shows the Auth0 Applications dashboard. At the top, there's a header with the Auth0 logo and a 'Create Application' button. Below the header, there's a sub-header: 'Setup a mobile, web or IoT application to use Auth0 for Authentication. Show more >'. The main area displays three application entries:

- backed (Test Application)**: Machine to Machine. Client ID: ccUGUqK1jSreQfxKK03WVsTe1ZzMyCB. Action buttons: three dots and a copy icon.
- frontend**: Single Page Application. Client ID: ArSx6AIrbsH02JJdQX4bLbzicwqpmrXR. Action buttons: three dots and a copy icon.
- Machine**: Machine to Machine. Client ID: Zd7sei0G3l1PJJMnSHk7M48CEpbWnnoc. Action buttons: three dots and a copy icon.

Figura 8 - Painel de administração da plataforma Auth0

4 Cenários de Teste e Validação

Nesta secção, apresentamos as evidências do funcionamento do sistema através de cenários de utilização reais, que comprovam a correta integração entre as diversas camadas da solução.

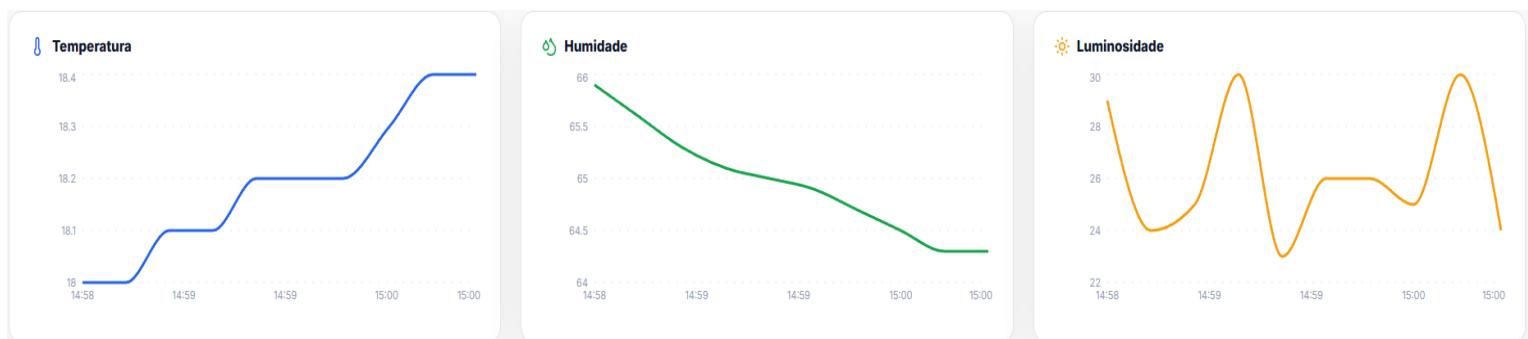


Figura 9 - Dashboard dos Valores do Sensor

4.1 Cenário A: Monitorização de Alertas

O primeiro cenário teve como objetivo verificar a eficácia do sistema de **alertas em tempo real**. Para tal, manipulamos os valores de Luminosidade para estes valores não passarem nas restrições e enviarem um alerta em relação a esse mesmo sensor.



Figura 10 - Simulação de envio de uma leitura crítica



Figura 11 - Receção do Alerta

4.2 Cenário B: Fluxo de Compra e Integração de Pagamentos

Neste teste, validámos a integração com o serviço SOAP externo e a consequente atualização de stock via SOAP interno. O fluxo iniciou-se com o utilizador a adicionar vinhos ao carrinho e a proceder ao pagamento, momento em que o sistema contactou o validador de cartões.

	Quantidade
Serra Licoroso	Licoroso
2019	Stock Baixo
2	

Figura 12 - Stock Inicial

Após a confirmação de sucesso no pagamento, verificámos na base de dados que a quantidade de stock do produto adquirido foi decrementada automaticamente, garantindo a consistência do inventário.

The screenshot shows a payment simulation interface. On the left, there's a 'Pagamento' section with a 'Método de Pagamento' dropdown set to 'Cartão de Crédito'. Below it are fields for 'Nome no Cartão' (Dani Cruz) and 'Número do Cartão' (4183600903848202). At the bottom are 'Validade (MM)' (12) and 'Validade (AA)' (25) fields. On the right, a 'Resumo do Pedido' box shows a subtotal of €18.00, taxes of 0, and a total of €18.00. A red 'Comprar' button is at the bottom. To the right of the summary is a success message: 'Compra concluída com sucesso' with a green checkmark, 'A sua encomenda foi processada com sucesso. Obrigado pela sua preferência.', and a link 'Clique para ver'.

Figura 13 - Simulação do Pagamento

	Quantidade
Serra Licoroso	Licoroso
2019	Stock Baixo
1	

Figura 14 - Stock Final

4.3 Cenário C: Gestão de Acessos

Por fim, testámos os mecanismos de segurança baseados em perfis. Ao aceder à plataforma com um utilizador de perfil "Client", confirmámos que as funcionalidades administrativas, como adicionar novas adegas ou gerir sensores que não se encontram acessíveis, validando assim a proteção das rotas sensíveis.



Figura 15 - Conta "Owner"



Figura 16 - Conta "Cliente"

5 Conclusão

A concretização da segunda fase do projeto VinhaTech permitiu consolidar os conhecimentos adquiridos na unidade curricular, demonstrando a viabilidade técnica da arquitetura proposta.

Concluímos com sucesso a integração de componentes tecnologicamente distintos, destacando-se a comunicação fluida entre a API REST e o serviço de dados SOAP legado, bem como a incorporação de serviços externos que aproximam o projeto de um contexto empresarial real.

Adicionalmente, a implementação de WebSockets conferiu ao módulo IoT a capacidade de resposta em tempo real exigida pelo problema.

Desta forma, o projeto cumpre integralmente os requisitos funcionais e técnicos propostos, resultando numa ferramenta robusta e funcional para a gestão inteligente de adegas.