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

ENGENHARIA EM SISTEMAS INFORMÁTICOS

**Relatório Integração de Sistemas de Informação**

No. 23502 – Manuel Fernandes

Índice

[**Relatório Integração de Sistemas de Informação** 1](#_Toc186800917)

[Índice de Figuras 2](#_Toc186800918)

[Introdução 4](#_Toc186800919)

[Base de Dados 5](#_Toc186800920)

[API 6](#_Toc186800921)

[RESTful API 7](#_Toc186800922)

[*Appsettings.json* 7](#_Toc186800923)

[Program.cs 8](#_Toc186800924)

[Services 9](#_Toc186800925)

[Models 11](#_Toc186800926)

[Controllers 12](#_Toc186800927)

[JWT Authentication 16](#_Toc186800928)

[Integração Externa – *Stripe* 19](#_Toc186800929)

[SOAP API 21](#_Toc186800930)

[Models 22](#_Toc186800931)

[Service 24](#_Toc186800932)

[Testes 27](#_Toc186800933)

[Insomnia 27](#_Toc186800934)

[Cloud 30](#_Toc186800935)

[Conclusão 31](#_Toc186800936)

[Referências 32](#_Toc186800937)

# Índice de Figuras

[Figura 1 - Base de Dados 5](#_Toc186800938)

[Figura 2 - Estrutura da API 6](#_Toc186800939)

[Figura 3 - appsetings.json 7](#_Toc186800940)

[Figura 4 - Stripe Program.cs 8](#_Toc186800941)

[Figura 5 - Swagger Documentation Program.cs 8](#_Toc186800942)

[Figura 6 - JWT Token Program.cs 8](#_Toc186800943)

[Figura 7 – JwtServices 9](#_Toc186800944)

[Figura 8 – UserService 10](#_Toc186800945)

[Figura 9 – UserModel 11](#_Toc186800946)

[Figura 10 - UserController (GetAll) 12](#_Toc186800947)

[Figura 11 - UserController (Post) 13](#_Toc186800948)

[Figura 12 - UserController (Put) 14](#_Toc186800949)

[Figura 13 - UserController (Delete) 15](#_Toc186800950)

[Figura 14 - JWT Login Implementation 16](#_Toc186800951)

[Figura 15 - Authorize Request 16](#_Toc186800952)

[Figura 16 - Login Test 17](#_Toc186800953)

[Figura 17 - Authorized User teste 18](#_Toc186800954)

[Figura 18 - Controller Stripe 19](#_Toc186800955)

[Figura 19 - Controller Stripe (1) 19](#_Toc186800956)

[Figura 20 - SOAP API Estrutura 21](#_Toc186800957)

[Figura 21 - SOAP PaymentModel 22](#_Toc186800958)

[Figura 22 - SOAP PaymentResponseModel 22](#_Toc186800959)

[Figura 23 - SOAP PaymentResponseModel 23](#_Toc186800960)

[Figura 24 - SOAP PaymentStatusResponse 23](#_Toc186800961)

[Figura 25 - PaymentServiceWS 24](#_Toc186800962)

[Figura 26 - PaymentServiceWS(1) 24](#_Toc186800963)

[Figura 27 - SOAP MakePayment 25](#_Toc186800964)

[Figura 28 - SOAP RefundPayment 26](#_Toc186800965)

[Figura 29 - Insomnia Variáveis 27](#_Toc186800966)

[Figura 30 - Insomnia Estrutura 28](#_Toc186800967)

[Figura 31 - Insomnia Test 29](#_Toc186800968)

[Figura 32 - Azure Painel 30](#_Toc186800969)

[Figura 33 - Azure SQL Database 30](#_Toc186800970)

# Introdução

Este projeto consiste no desenvolvimento de uma aplicação de loja online, com o objetivo de permitir a gestão de produtos, utilizadores, encomendas e pagamentos. Para isso, foi criada uma *API* utilizando duas abordagens: *RESTful* para as funcionalidades principais e *SOAP* para o processamento de pagamentos, garantindo flexibilidade e integração com sistemas externos.

A autenticação dos utilizadores foi realizada com *tokens* *JWT*, assegurando a segurança na comunicação. A aplicação também inclui testes automatizados para garantir o funcionamento correto de suas funcionalidades e foi hospedada na plataforma *Azure* para garantir escalabilidade e disponibilidade.

Além disso, foi integrada a *API* do *Stripe* para o processamento de pagamentos, e a documentação da *API* foi realizada utilizando o *Open* *API* *Swagger*, facilitando a compreensão e utilização das funcionalidades da aplicação.

# Base de Dados

A base de dados para este projeto foi feita em *SQL*, diretamente no *Microsoft SQL Server Management Studio* como está evidente na **Figura 1**. Apesar da base de dados ter sido também feita com recurso ao ***Microsoft Azure***, infelizmente e acidentalmente, os créditos da mesma expiraram e não foi possível contar com o *deploy* quer seja da *api* ou da base de dados na *cloud*.

A computer screen shot of a computer program

Description automatically generated

Figura 1 - Base de Dados

# API

Para este projeto, foram desenvolvidas *APIs* em *RESTful e SOAP*. Sendo que o foco principal foi na vertente *RESTful*, já que apenas a parte dos Pagamentos foram feitos em *SOAP*.

Na Figura 2, podemos observar a estrutura das pastas e ficheiros da solução da *api*. Dentro da solução podemos encontrar a RESTful API e a SOAP API devidamente separadas em projetos diferentes na mesma solução

A screenshot of a computer program

Description automatically generated

Figura 2 - Estrutura da API

De seguida, damos início à exploração destas *API*.

## RESTful API

Foi desenvolvida uma *API RESTful* que cobre todas as operações necessárias para a gestão da loja online, incluindo utilizadores, produtos, encomendas, carrinhos e pagamentos.

* Estrutura *MVC* (*Model*-*View*-*Controller*).
* Uso do *Microsoft*.*Data*.*SqlClient* para interações com a base de dados.
* Conformidade com os princípios *RESTful*.
* Utilização de Tokens JWT

Esta *RESTful* *API* conta com *Controllers*, *Models*, *Services* e uma pasta *Jwt* que contém as *settings* dos *Tokens* *JWT*.

### *Appsettings.json*

No ficheiro *appsettings.json*, através da Figura 3, podemos encontrar várias *conncetion strings* que foram utilizadas ao longo do projeto. Sendo as mais importantes, a “*Home PC Server*”e “*Azure SQL Server*”*.*  Como referido anteriormente, infelizmente, o servidor da plataforma *Azure* não se encontra disponível pois os créditos da mesma expiraram.

A black screen with colorful lines

Description automatically generated

Figura 3 - appsetings.json

Para além das ligações ao servidor, temos também os parâmetros necessários para a integração com o *Stripe* e com os *Tokens JWT*.

### Program.cs

No ficheiro base, *program.cs*, de onde existem várias configurações necessárias para o funcionamento da *API*, podemos evidenciar as configurações da integração externa com o *Stripe*, *Swagger OpenAPI Documentation* e *Tokens JWT. Os mesmos estão presentes nas Figuras 4, 5 e 6, respetivamente.*

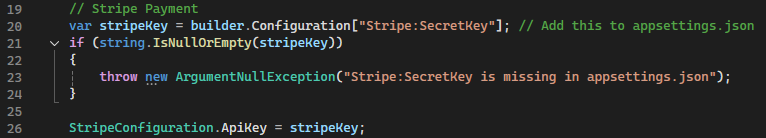


Figura 4 - Stripe Program.cs

A screen shot of a computer code

Description automatically generated

Figura 5 - Swagger Documentation Program.cs

A black screen with text

Description automatically generated

Figura 6 - JWT Token Program.cs

### Services

A pasta *Services*, contém dois ficheiros. Sendo os mesmos, *JwtService.cs* e *UserService.cs*. Ambos os ficheiros servem para auxiliar os *Tokens JWT*.

Como proteções de dados como a adição de *hashes* ou *salts* ou quaisqueres outros procedimentos não foram implementados neste projeto, as credenciais dos *users* são verificadas diretamente entre o *username* e *password* inseridas com os dados na base de dados.

#### JwtService.cs

Este serviço auxilia diretamente os *Tokens JWT*, sendo que é este mesmo serviço que é encarregue de os gerar e atribuir a um *UserID*.

A computer screen with text and images

Description automatically generated

Figura 7 – JwtServices

#### UserService

O *UserService* é responsável pela validação dos dados de autenticação introduzidos pelo *user,* quanto aos dados existentes ne *database*.

*A screen shot of a computer program

Description automatically generated*

Figura 8 – UserService

### Models

A camada *Models*, tal como o próprio nome indica, detém os modelos, ou tabelas, da base de dados em *SQL*. Nesta pasta encontram-se cerca de **20 modelos**, dito isto, apenas 1 modelo será representado. Já que, os modelos **são bastante semelhantes** entre si e repetem o mesmo processo.

A screenshot of a computer program

Description automatically generated

Figura 9 – UserModel

Para além do *UserModel* completo e igualado às tabelas da base de dados, temos também o *LoginModel* que é utilizado para efeitos de autenticação em conjunto com os serviços de *Tokens JWT*.

### Controllers

A camada *Controllers*, é responsável pelas interações do utilizador com a base de dados através da *web*. Tal como nos *Models*, esta camada é bastante repetidora e os ficheiros acabam por ser bastantes semelhantes uns dos outros.

É aqui, que são feitos os *“CRUD”*, através de *POSTS, GETS, PUTS e DELETES*. No *controller* na Figura 10, podemos ver um *GET Request* do *controller Users*.

A screen shot of a computer program

Description automatically generated

Figura 10 - UserController (GetAll)

Na seguinte Figura 11, podemos observar um *POST Request* para adicionar um *User*.

A computer screen shot of a program code

Description automatically generated

Figura 11 - UserController (Post)

De seguida, na Figura 12, temos um *PUT Request* para atualizar um *user* já existente.

A computer screen shot of a program code

Description automatically generated

Figura 12 - UserController (Put)

Por fim, na Figura 13, o *DELETE Request* apaga um *user* da base de dados.

A computer screen shot of a program code

Description automatically generated

Figura 13 - UserController (Delete)

### JWT Authentication

A autenticação por *JWT (JSON Web Token)*, foi implementada dentro do *LoginController* como está evidente na Figura 14.

A computer screen shot of a program code

Description automatically generated

Figura 14 - JWT Login Implementation

Para comprovar que a autenticação foi bem implementada, podemos adicionar “*[Authorize]*” por cima de um *request* para que seja restrito a *users* autorizados. Para efeitos de teste, no *controller* das encomendas, *order,* o request para criar uma encomenda necessita de um user devidamente autorizado. Podemos observá-lo na Figura 15

A computer screen shot of a program code

Description automatically generated

Figura 15 - Authorize Request

Com recurso à aplicação ***Insomnia***, que será mencionada mais detalhadamente brevemente, podemos confirmar a correta e o funcionamento desta funcionalidade. Na Figura 16, utilizamos uma combinação de *username* e *password* válidas para efeitos de login, e a resposta, se válida, será de um *token*.

A screenshot of a computer

Description automatically generated

Figura 16 - Login Test

Tal como esperado, o resultado ***200 OK*** reflete um *request* bem-sucedido, tal como a resposta do *token* que foi atribuído ao *user*.

Por fim, na Figura 17, utilizamos o *user* autorizado para criar uma *order*. Para que seja possível, adicionamos o *token* deste *user* ao campo ***Auth*** na aplicação *Insomnia*.

A screenshot of a computer

Description automatically generated

Figura 17 - Authorized User teste

### Integração Externa – *Stripe*

Como integração externa, foi decido recorrer ao serviço de pagamentos ***Stripe***. Uma vez que esta aplicação modela uma loja online, faria mais que sentido integrar um serviço de pagamentos externo. Dada a facilidade de interações com a sua *API*, recorremos então a este serviço.

Na Figura 18 e 19, está presente o *controller* encarregue dos Pagamentos via *Stripe.*

A computer screen with many colorful text

Description automatically generated

Figura 18 - Controller Stripe

A screen shot of a computer program

Description automatically generated

Figura 19 - Controller Stripe (1)

Este controller, com recurso a outras funções que fazem interações diretas com a base de dados tais como recolher dados de encomendas, guardar o ID do pagamento do *Stripe* na nossa base de dados e atribuí-lo à encomenda correspondente, atualizar *status* dos pagamentos e recolher os métodos de pagamentos disponíveis na base de dados, é responsável pela verificação de dados e de iniciar um pagamento de acordo com os parâmetros do *Stripe*.

## SOAP API

Quanto à SOAP API, foi decidido fazer apenas o *controller* dos pagamentos (“interno”, não é o do *Stripe*). Na Figura 20 podemos ver a estrutura do projeto ***SOAP-API***

A screenshot of a computer program

Description automatically generated

Figura 20 - SOAP API Estrutura

Dentro do projeto estão presentes os *Models* necessários e o *PaymentServiceWS.asmx* que trata devidamente dos *WebMethods*.

### Models

Os *Models* presentes na *SOAP API*, são, tal como na *RESTful*, modelos de dados da base de dados que representam as tabelas da mesma. Ou, neste caso, “preparam” modelos necessários para interagir com os *requests*. Nas Figuras 21, 22, 23 e 24, estarão disponíveis os *Models* utilizados.

A screen shot of a computer program

Description automatically generated

Figura 21 - SOAP PaymentModel

A screen shot of a computer program

Description automatically generated

Figura 22 - SOAP PaymentResponseModel

A screen shot of a computer program

Description automatically generated

Figura 23 - SOAP PaymentResponseModel

A screen shot of a computer program

Description automatically generated

Figura 24 - SOAP PaymentStatusResponse

### Service

Na pasta *Service*, está presente o ficheiro *PaymentServiceWS.asmx*. Serviço este que é responsável por todos os *WebMethods* e as suas interações com a base de dados. Nas próximas Figuras 25 e 26, podemos observar todos os *WebMethods*.

A screenshot of a computer program

Description automatically generated

Figura 25 - PaymentServiceWS

A screenshot of a computer program

Description automatically generated

Figura 26 - PaymentServiceWS(1)

Novamente, dado que todos os estes métodos recorrem ao mesmo processo repetidamente com algumas modificações, será apenas exibido com detalhe apenas dois *WebMethods*. Sendo o primeiro, o método para criar um pagamento, como será exibido na Figura 27.

A computer screen shot of a program code

Description automatically generated

Figura 27 - SOAP MakePayment

Este método aceita como parâmetros o *orderId*, *amount* e o *paymentMethodId para efetuar as necessárias operações com a base de dados.*

O próximo *WebMethod*, será o *RefundPayment*. Onde este efetua um novo pagamento com o valor equivalente ao da *order* só que negativo. Com o intuito de simular um reembolso. O mesmo está explícito na seguinte Figura 28.

A screen shot of a computer program

Description automatically generated

Figura 28 - SOAP RefundPayment

## Testes

Como já foi referido, os testes de aplicação foram realizados com recurso à ferramenta ***Insomnia***. Esta ferramenta facilita a gestão dos *requests*. Possibilita nos de os organizar, adicionar *bodies*, *auth tokens*, adicionar *variáveis*, recorrer a todos os *requests* existentes e entre outros.

### Insomnia

Para facilitar ainda mais o processo, recorremos à utilização de **variáveis** **globais**. Estas variáveis encurtam um *url* de acesso para uma variável com um nome à escolha. Todas as variáveis utilizadas encontram-se disponíveis na próxima Figura 29.

A screenshot of a computer

Description automatically generated

Figura 29 - Insomnia Variáveis

Quanto à estrutura, todos os *requests*, sejam eles *RESTful* ou *SOAP*, foram todos estruturados e agrupados por modelo como está evidente na Figura 30.

A screenshot of a computer

Description automatically generated

Figura 30 - Insomnia Estrutura

Por fim, foram feitos verdadeiros **Testes** com recurso à aceitação de *scripts* que estão ao dispor do utilizador. Como podemos presencial na seguinte Figura 31, foi efetuado um teste cum *script* que atua **depois** da resposta do *request*. Onde este **passa** o teste se o resultado do *request* for de ***Status Code: 200 OK***.

A screenshot of a computer

Description automatically generated

Figura 31 - Insomnia Test

## Cloud

Apesar de não estar devidamente expresso neste relatório nem no projeto, houve uma inicial base de dados na *cloud* da *Azure*.

Contudo, e infelizmente, os créditos de **Estudante** foram todos utilizados acidentalmente até ao esgotamento, como podemos confirmar na seguinte Figura 32.

A screenshot of a computer

Description automatically generated

Figura 32 - Azure Painel

Mesmo não estando ativa nem presente no projeto, fica aqui, na figura 33, o *SQL Server* no painel da *Microsoft Azure*.

A screenshot of a computer

Description automatically generated

Figura 33 - Azure SQL Database

## Conclusão

Em suma, este projeto resultou na criação de uma *API* funcional e segura para um sistema de *e-commerce*, integrando tecnologias modernas como autenticação *JWT* e serviços externos como *Stripe*. A *RESTful* *API* e a *SOAP* API, atendem a diferentes necessidades de integração.

A documentação detalhada e os testes realizados reforçam a confiabilidade do sistema, que cumpre os objetivos propostos e está preparado para futuras expansões. Este trabalho consolidou boas práticas de desenvolvimento e será uma base sólida para projetos futuros.

## Referências

<https://www.youtube.com/watch?v=V5ooPC3wBWI>

<https://www.youtube.com/watch?v=3Op008xgFjw>

<https://www.youtube.com/watch?v=ZVkLrw37j1Q>

<https://docs.stripe.com/payments/payment-intents>

<https://stackoverflow.com/>

<https://chatgpt.com/>

<https://portal.azure.com/>