

INTEGRAÇÃO DE SISTEMAS

Relatório do Trabalho Prático – Parte 2

Expansão do Sistema de Integração

Mestrado em Engenharia Informática

2024/2025

Docentes da unidade curricular:

Hugo Paredes

João Sousa

Realizado por:

Paulo Tomás da Silva Amorim; AL2024116531; al2024116531@alunos.utad.pt

António Pedro Marques Trancoso; AL2024116930; al2024116930@alunos.utad.pt

➤ Expansão do Sistema de Integração

Este relatório tem como principal objetivo apresentar de forma detalhada, a implementação e a validação funcional das diferentes componentes desenvolvidas na segunda parte do trabalho da cadeira de Mestrado em Engenharia Informática, Integração de Sistemas.

Nesta fase, o foco centrou-se na orquestração da comunicação entre sistemas heterogéneos, recorrendo a tecnologias como Web Services SOAP, RabbitMQ e RabbitMQ Streams. Adicionalmente, foram concebidas e integradas interfaces gráficas que permitem visualizar, em tempo real, o comportamento do sistema e os dados gerados, proporcionando uma interação mais intuitiva e uma melhor compreensão dos resultados obtidos. O relatório documenta ainda os testes efetuados, ilustrando o correto funcionamento de cada módulo e validando a execução da solução proposta.

➤ Serviços Financeiros via Web Services SOAP

Nesta secção, são apresentados os testes realizados à API SOAP desenvolvida, bem como à sua utilização através de um Cliente SOAP. A API expõe um conjunto de serviços orientados à análise financeira da produção, permitindo a consulta de diversos indicadores contabilísticos.

• Interface Geral dos Serviços (API_SOAP)

A Figura 1 apresenta a interface principal disponibilizada pela API SOAP, onde se encontram listadas todas as operações suportadas. Esta interface foi construída com o objetivo de facilitar o acesso a métodos de consulta financeira, tais como cálculo de prejuízos, lucros e custos de produção, para além da visualização individual de cada peça.

WebService1

As operações que se seguem são suportadas. Para obter uma definição formal, consulte a [Descrição do Serviço](#).

- [GetPecaMaiorPrejuizo](#)
- [ObterCustosTotaisPorPeriodo](#)
- [ObterDadosFinanceirosPorPeca](#)
- [ObterLucroTotalPorPeriodo](#)
- [ObterPrejuizoTotalPorPeca](#)

Este serviço Web está a utilizar <http://tempuri.org/> como espaço de nomes predefinido.

Recomendação: altere o espaço de nomes predefinido antes de o serviço XML Web Service ser tomado público.

Cada serviço XML Web Service necessita de um espaço de nomes exclusivo para que as aplicações cliente o possam distinguir de outros serviços na Web. <http://tempuri.org/> está disponível para XML Web Services que estejam em desenvolvimento; no entanto, os XML Web Services publicados deverão utilizar um espaço de nome mais permanente.

O serviço XML Web Service do utilizador deverá ser identificado por um espaço de nomes controlado pelo utilizador. Por exemplo, pode utilizar o nome de domínio da Internet da empresa como parte do espaço de nomes. Apesar de muitos espaços de nomes de XML Web Services se parecerem com URL, não necessitam de apontar para recursos reais na Web. (Os espaços de nomes de XML Web Services são URIs.)

Nos XML Web Services criados através do ASP.NET, o espaço de nomes predefinido pode ser alterado através da propriedade Namespace do atributo WebService. O atributo WebService é um atributo aplicado à classe que contém os métodos do serviço XML Web Service. Segue-se um exemplo de código que define o espaço de nomes como "http://microsoft.com/webServices/".

C#

```
[WebService(Namespace="http://microsoft.com/webServices/")]
public class MyWebService {
    // Implementation
}
```

Visual Basic

```
<WebService(Namespace="http://microsoft.com/webServices/")> Public Class MyWebService
    ' Implementation
End Class
```

C++

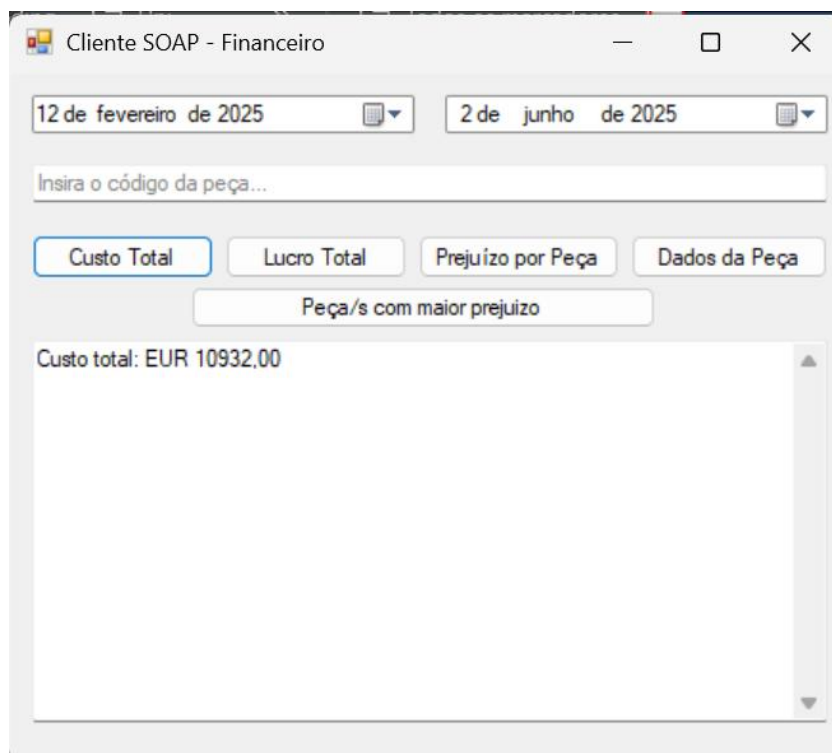
```
[WebService(Namespace="http://microsoft.com/webServices/")]
public ref class MyWebService {
    // Implementation
};
```

Para obter mais detalhes sobre espaços de nomes XML, consulte a recomendação W3C em [Namespaces in XML](#).

Figura 1 - Interface Geral dos Serviços WebService

- **Teste de Invocação ao Serviço *ObterCustosTotaisPorPeriodo***

A Figura 2 demonstra a execução do método *ObterCustosTotaisPorPeriodo*, responsável por apresentar o Custo total obtido no respetivo período selecionado, com base nos dados armazenados na base de dados Contabilidade.



Cliente SOAP - Financeiro

12 de fevereiro de 2025 2 de junho de 2025

Insira o código da peça...

Custo Total Lucro Total Prejuízo por Peça Dados da Peça

Peça/s com maior prejuízo

Custo total: EUR 10932,00

Figura 2 - Serviço *ObterCustosTotaisPorPeriodo*

- **Teste de Invocação ao Serviço *GetPecaMaiorPrejuizo***

A Figura 3 demonstra a execução do método *GetPecaMaiorPrejuizo*, responsável por retornar a(s) peça(s) que registaram o maior valor de prejuízo, com base nos dados armazenados na base de dados contabilística.

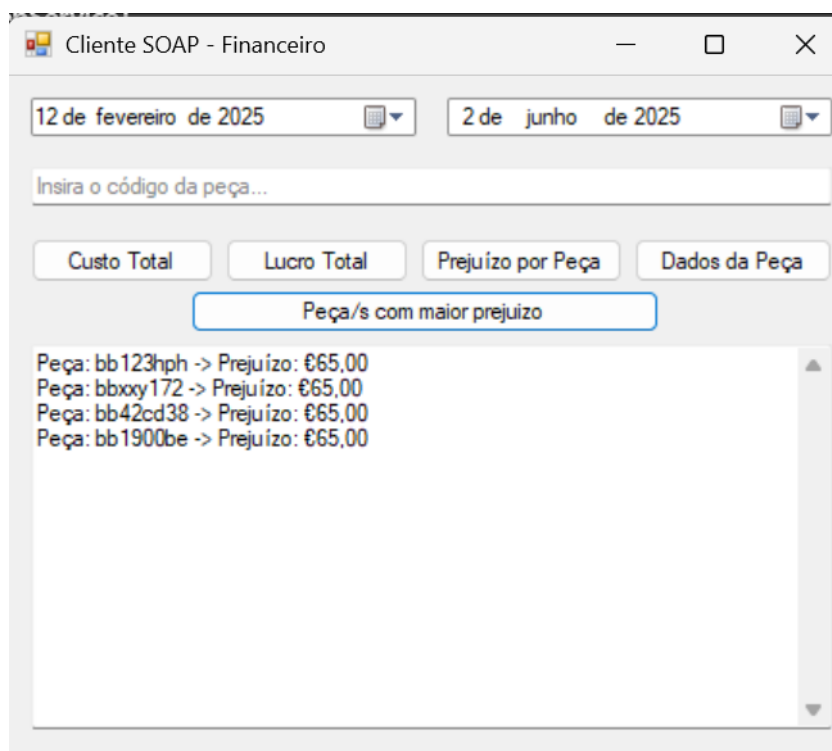


Figura 3 – Serviço GetPecaMaiorPrejuizo

- **Teste de Invocação ao Serviço ObterLucroTotalPorPeriodo**

A Figura 4 demonstra a execução do método ObterLucroTotalPorPeriodo, responsável por apresentar o lucro total obtido no respetivo período selecionado, com base nos dados armazenados na base de dados Contabilidade

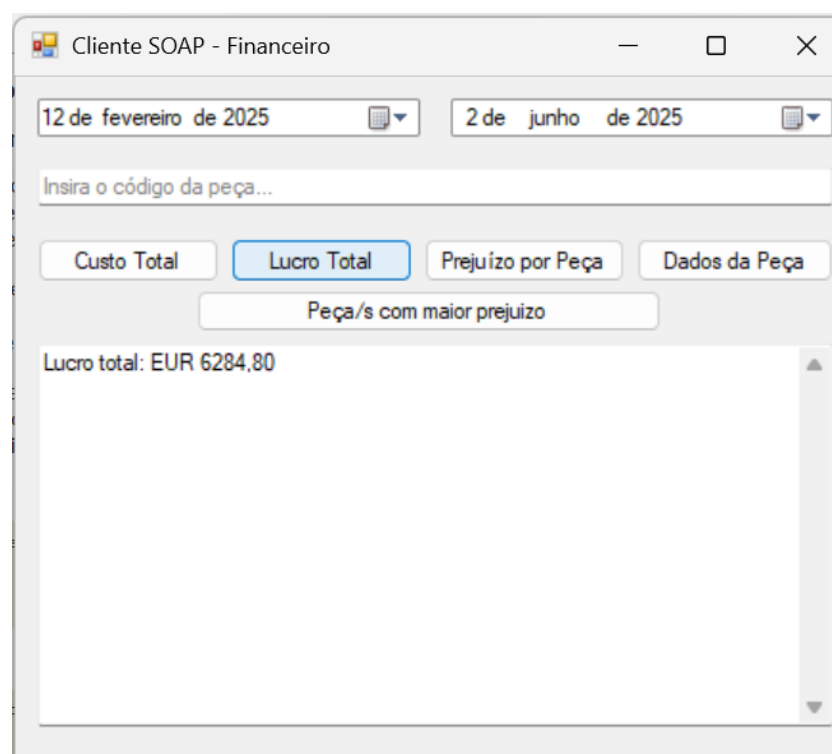


Figura 4 - Serviço ObterLucroTotalPorPeriodo

- **Teste de Invocação ao Serviço *ObterPrejuizoTotalPorPeca***

A Figura 5 demonstra a execução do método *ObterPrejuizoTotalPorPeca*, responsável por apresentar o prejuízo total obtido pelas diversas peças que causaram prejuízos, com base nos dados armazenados na base de dados Contabilidade

Cliente SOAP - Financeiro

12 de fevereiro de 2025 2 de junho de 2025

Insira o código da peça...

Custo Total Lucro Total **Prejuízo por Peça** Dados da Peça

Peça/s com maior prejuízo

Peça: aa0282b3 ->	Prejuízo: EUR 26,10
Peça: aa02bd68 ->	Prejuízo: EUR 27,00
Peça: aa0bd553 ->	Prejuízo: EUR 10,80
Peça: aa1701bb ->	Prejuízo: EUR 45,00
Peça: aa194a83 ->	Prejuízo: EUR 14,40
Peça: aa20bc46 ->	Prejuízo: EUR 9,90
Peça: aa2a3d21 ->	Prejuízo: EUR 28,80
Peça: aa3239e3 ->	Prejuízo: EUR 30,60
Peça: aa3520c6 ->	Prejuízo: EUR 28,80
Peça: aa39ab92 ->	Prejuízo: EUR 23,40
Peça: aa43824f ->	Prejuízo: EUR 20,70
Peça: aa43fb81 ->	Prejuízo: EUR 16,20
Peça: aa4ce54d ->	Prejuízo: EUR 20,70
Peça: aa51b9de ->	Prejuízo: EUR 36,90
Peça: aa55f578 ->	Prejuízo: EUR 44,10

Figura 5 - Serviço *ObterPrejuizoTotalPorPeca*

- **Teste de Invocação ao Serviço *ObterDadosFinanceirosPorPeca***

A Figura 6 demonstra a execução do método *ObterDadosFinanceiroPorPeca*, responsável por apresentar os dados da respetiva peça indicada em “Insira o código da peça...”, dados esses como o tempo de produção, custo associado, prejuízo causado e lucro obtido com essa peça.

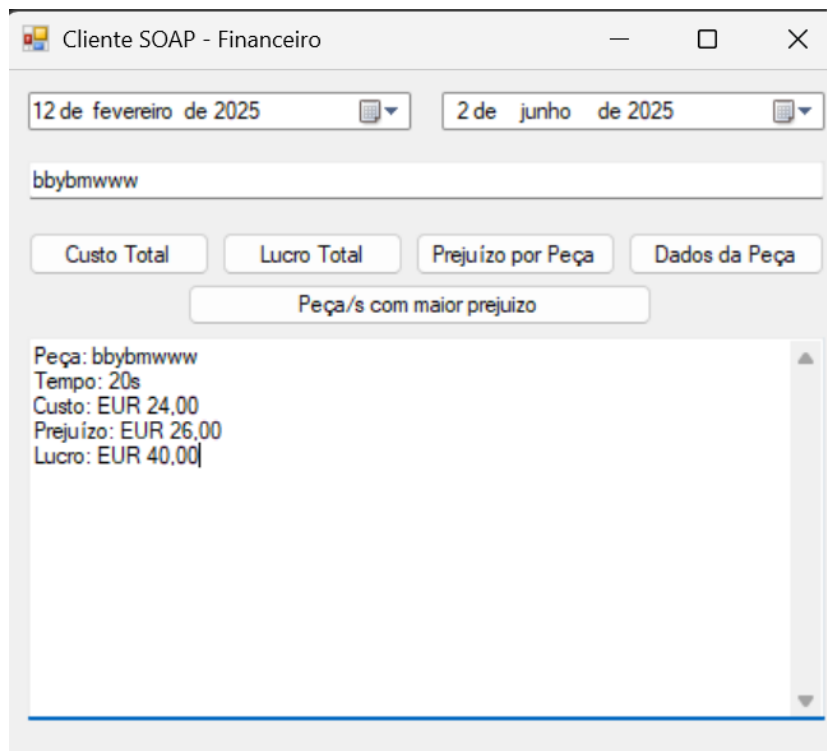


Figura 6 - Serviço ObterDadosFinanceirosPorPeca

➤ Cliente SOAP (Client_SOAP)

O Cliente SOAP foi desenvolvido com o objetivo de proporcionar uma interface gráfica intuitiva para a interação com os Web Services financeiros disponibilizados pela API SOAP. Esta aplicação permite ao utilizador consumir os serviços remotamente, sem necessidade de interação direta com o servidor ou o código subjacente.

A Figura 7 apresenta o menu principal da aplicação cliente, onde se encontram listadas todas as funcionalidades implementadas que já foram exemplificadas e testadas em cima. Através desta interface, o utilizador pode invocar então métodos como GetPecaMaiorPrejuizo, ObterCustosTotaisPorPeriodo, entre outros, com os respetivos parâmetros de entrada (Data de início e fim), e visualizar os resultados retornados de forma clara e organizada.

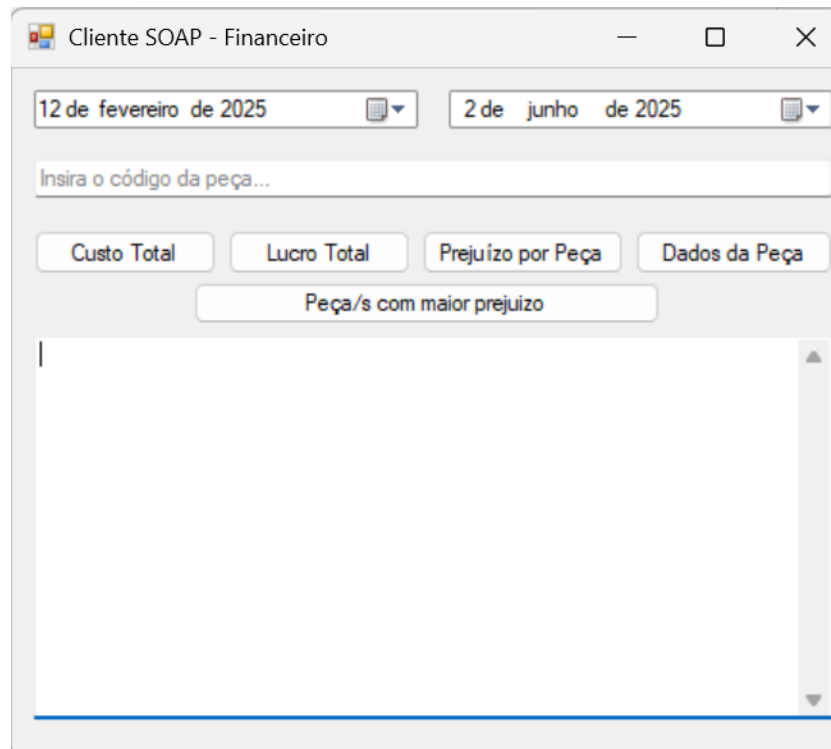


Figura 7 – Interface Gráfica de Interação do Cliente SOAP

➤ Desacoplamento Arquitetural através do RabbitMQ

Nesta secção é demonstrada a implementação de uma comunicação assíncrona entre a nova aplicação legada e a plataforma web, recorrendo ao middleware RabbitMQ como mecanismo de desacoplamento. Esta abordagem permite que os dados de produção sejam transmitidos de forma eficiente e não bloqueante, promovendo maior escalabilidade e flexibilidade no sistema.

A nova aplicação legada foi desenvolvida com o objetivo de simular o ambiente de produção (Producer), gerando automaticamente dados de peças com base em intervalos de tempo aleatórios. Estes dados são posteriormente publicados num topic exchange do RabbitMQ.

A Figura 8 apresenta a interface de consola da aplicação, onde se pode observar o envio das mensagens, bem como a sua publicação simultânea numa stream, garantindo redundância e persistência dos dados.

```
info Stream 'producao_stream' já existe.
Producer ativo.
Topic Exchange 'dados.producao.ok': {"data":"2025-06-03","hora":"00:19:44","codigo_peca":"aa1950a3","tempo_producao":
43,"resultado_teste":"01"}
Stream {"data":"2025-06-03","hora":"00:19:44","codigo_peca":"aa1950a3","tempo_producao":43,"resultado_teste":"01"}
Topic Exchange 'dados.producao.falha': {"data":"2025-06-03","hora":"00:19:49","codigo_peca":"bb242fa7","tempo_produca
o":31,"resultado_teste":"05"}
Stream {"data":"2025-06-03","hora":"00:19:49","codigo_peca":"bb242fa7","tempo_producao":31,"resultado_teste":"05"}
Topic Exchange 'dados.producao.falha': {"data":"2025-06-03","hora":"00:19:54","codigo_peca":"bb0dee79","tempo_produca
o":29,"resultado_teste":"06"}
Stream {"data":"2025-06-03","hora":"00:19:54","codigo_peca":"bb0dee79","tempo_producao":29,"resultado_teste":"06"}
Topic Exchange 'dados.producao.falha': {"data":"2025-06-03","hora":"00:19:59","codigo_peca":"bb60a728","tempo_produca
o":38,"resultado_teste":"02"}
```

Figura 8 – Teste de Publicação de Mensagens no Topic exchange do RabbitMQ

O Componente recetor (Consumer) foi desenvolvido para consumir as mensagens disponibilizadas pelo RabbitMQ. Após a receção, os dados são validados e inseridos na base de dados através da API REST da plataforma web (ProducaoAPI), API essa já desenvolvida na primeira parte da componente prática da unidade curricular. A interface do recetor está desenhada para apresentar exclusivamente as peças que falharam nos testes de qualidade, permitindo uma monitorização eficaz dos defeitos em tempo real.

A Figura 9 ilustra a consola do recetor em funcionamento, evidenciando a receção contínua de mensagens e a confirmação de inserção bem-sucedida dos registos na base de dados

```
[*] À espera de mensagens da produção...
Pressiona Enter para sair.
+++ Produto enviado para a API com sucesso.
+++ Produto enviado para a API com sucesso.
+++ Produto enviado para a API com sucesso.
!!! PEÇA COM FALHA:
? Código: aaec7ca6
? Tempo: 28 segundos
? Resultado: 03
? Produzido em: 03/06/2025 00:28:56
-----
+++ Produto enviado para a API com sucesso.
```

Figura 9 – Teste de Consumo de Mensagens via RabbitMQ e inserção na Base de dados via API Rest

➤ **Análise em Tempo Real com RabbitMQ Streams**

Nesta secção é descrita a utilização do RabbitMQ Streams como mecanismo de persistência e transmissão contínua de dados de produção, possibilitando o processamento em tempo real. Esta abordagem é fundamental para permitir a análise dinâmica dos dados e a construção de dashboards, promovendo uma maior visibilidade operacional.

A mesma aplicação legada descrita na anteriormente foi configurada para enviar, além de mensagens para o topic exchange, cópias dos dados diretamente para um stream do RabbitMQ. Esta duplicação garante a persistência fiável dos dados, mesmo em cenários onde o consumidor final possa estar temporariamente indisponível. A lógica de envio utilizada é a mesma que já foi validada nos testes iniciais.

Para visualizar os dados em tempo real, foi desenvolvida uma aplicação desktop (Gestao_app) com interface gráfica (GUI) que se liga diretamente ao stream e processa as mensagens recebidas. Esta aplicação atua como dashboard de Data Analytics, apresentando métricas operacionais relevantes que são atualizadas continuamente à medida que os dados são consumidos.

A Figura 10 ilustra o dashboard em funcionamento, onde são exibidos indicadores como:

- **Total de peças produzidas;**
- **Número de peças OK;**
- **Número de peças com falhas;**
- **Tempo médio de produção das peças.**

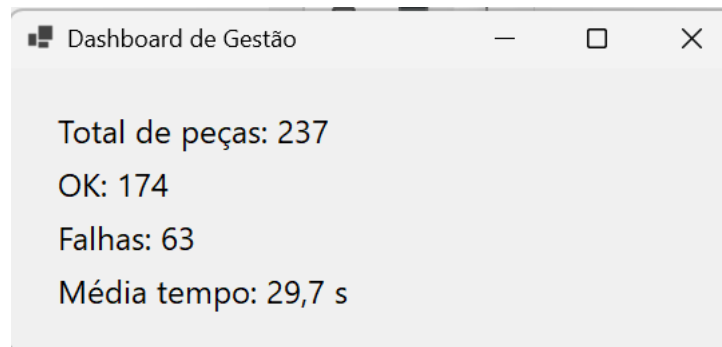


Figura 10 - Execução do Dashboard tem Tempo Real

➤ Considerações finais

Optou-se pela utilização do RabbitMQ para a realização desta segunda parte do trabalho por diversas razões, nomeadamente o facto de já estarmos familiarizados com esta tecnologia e de esta já se encontrar instalada nas nossas máquinas. Após a implementação do topic exchange, surgiu a ideia de testar outra ferramenta de mensagens assíncrona, como o Kafka, para a comunicação de streams. No entanto, dado que o desenvolvimento inicial já tinha sido feito com RabbitMQ, a mudança para o Kafka implicaria refazer grande parte do trabalho. Assim, decidiu-se manter a solução com RabbitMQ.