

Computação Orientada a Serviços

Trabalho Prático 1

MEIC, Semestre de Inverno 2013/2014

**Docente: Discente:**

Cátia Vaz Ricardo Nunes, nº 31656

Índice

[1. Introdução 1](#_Toc372304828)

[2. Modelo 2](#_Toc372304829)

[3. Serviços 3](#_Toc372304830)

[2.1 BrandAnalyticsService.Clients 4](#_Toc372304831)

[2.2 BrandAnalyticsService.Employees 5](#_Toc372304832)

[2.3 TwitterSpy 6](#_Toc372304833)

# Introdução

O trabalho tinha como objectivo a criação de uma infra-estrutura que desse suporte à empresa *BrandAnalytics* gerir o seu sistema de estudo social.

A infra-estrutura criada contém três serviços e um protótipo de uma aplicação cliente, o cliente foi desenvolvido com a tecnologia WPF, os serviços com WCF e WF.

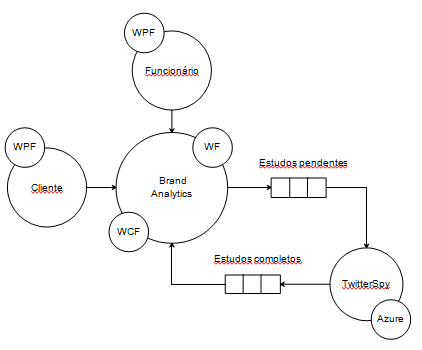


Ilustração 1 - Arquitectura desenvolvida.

A Ilustração 1 demonstra os diferentes pontos da arquitectura, apesar de não ser assunto do projecto a escalabilidade foi tida em conta. O sistema tem uma componente cliente que são os funcionários e os clientes, ambos têm uma aplicação de forma a comunicar com o sistema.

O núcleo de negocio de *BrandAnalytics* foi dividido em dois serviços, um para operações de clientes e outro para operações de funcionários, permitindo assim a separação (“física”) destes caso seja necessário.

O serviço TwitterSpy encontra-se isolado de todo o sistema e comunicável apenas através de uma *queue* permitindo assim que novas instâncias sejam adicionadas caso exista demasiada carga nesta componente do sistema.

# Modelo

# Serviços

Como referido anteriormente a implementação está divida em dois serviços, assim além da separação das lógicas de negócio permite ainda que existam modelos de dados diferentes para cada um deles. Os serviços são *BrandAnalyticsService.Clients* e *BrandAnalyticsService.Employees*.

A implementação dos serviços está divida em três componentes:

1. O *endpoint* do serviço responsável por receber os pedidos dos clientes e de chamar a lógica de negócio.
2. A *Activity* que implementa a lógica de cada funcionalidade do serviço esta normalmente contém os mesmos parâmetros que o serviço em si , para actua como uma separação entre serviço e *workflow* (podendo portanto ser reutilizada em outros contextos).

*Activities* auxiliares, por exemplo uma actividade que dado um id de cliente retorna uma instância desse cliente, útil devido ao estado inicial de grande parte das *activities* ser verificar se existem clientes ou estudos.

O “*workflow*” de um estudo segue os seguintes passos:

* O cliente pede um estudo de um tema.
* O funcionário recebe o estudo e trata de fornecer tópicos relevantes ao tema.
* O funcionário submete o pedido para o serviço *TwitterSpy.*
* O serviço *TwitterSpy* processa o estudo durante o tempo indicado.
* O serviço *TwitterSpy* notifica que o estudo está completo.
* O funcionário revê o estudo e decide completar ou recomeçar o estudo.
* O cliente recebe o relatório do estudo.

Em qualquer uma das fases o cliente pode cancelar o estudo.

## BrandAnalyticsService.Clients

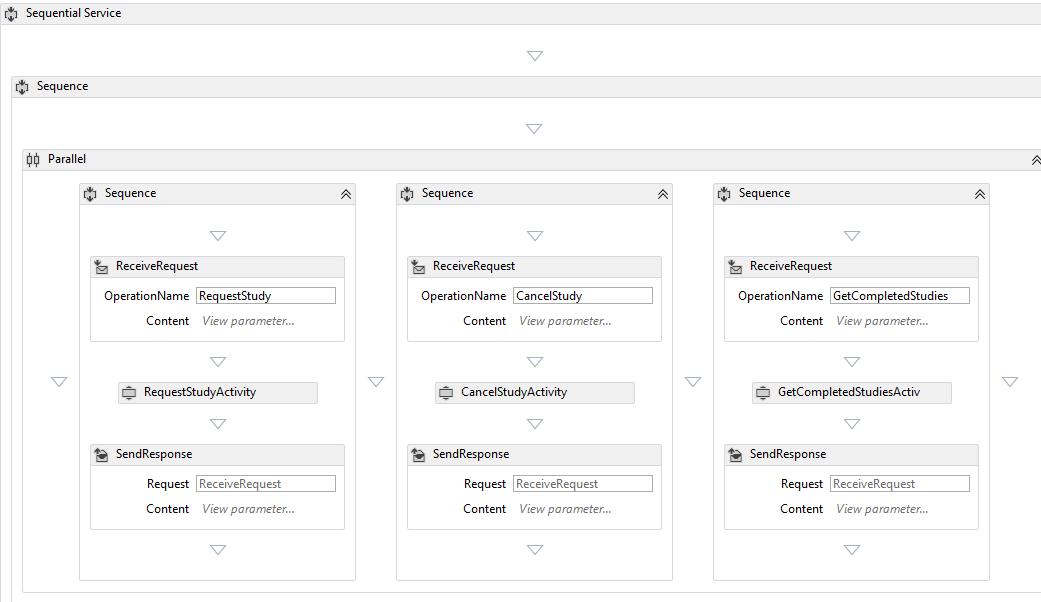


Ilustração 2 - Workflow do serviço de clientes.

Este serviço oferece as seguintes operações:

* RequestStudy – Permitindo a qualquer cliente peça um novo estudo.
* CancelStudy – Para cancelar um estudo que o utilizador tenha encomendado.
* GetCompletedStudies – Permite que o cliente visualize todos os relatórios de estudos completos.

## BrandAnalyticsService.Employees

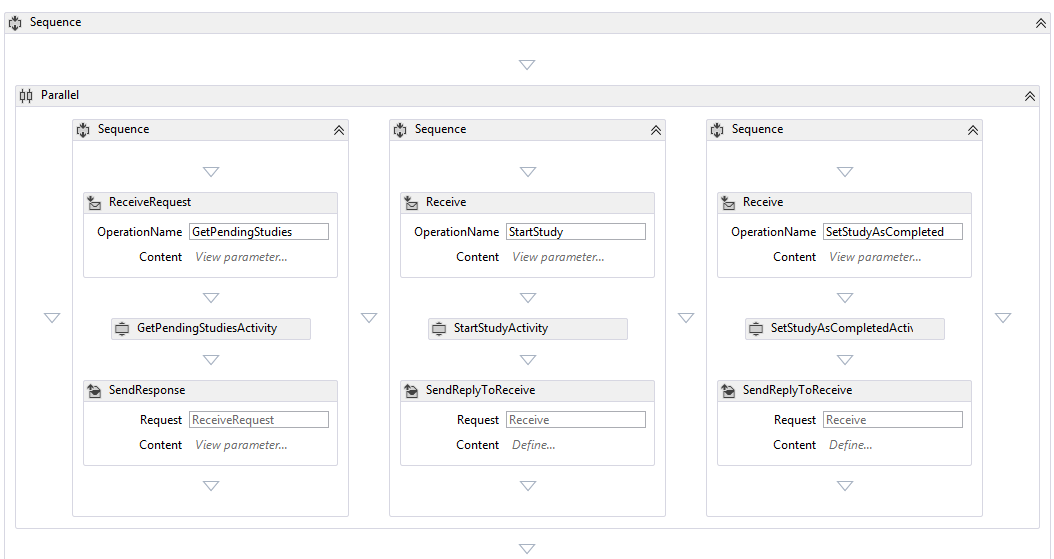


Ilustração 3 - Workflow do serviço de funcionários.

O serviço contém as seguintes funcionalidades:

* GetPendingStudies – permite ao funcionário visualizar os estudos pendentes, de forma a poder estudá-los e aprová-los.
* StartStudy – com a análise realizada permite aos funcionários começar um dado estudo.
* SetStudyAsCompleted – Após o estudo esteja completado e revisto permite a conclusão do mesmo notificando assim o cliente.

## TwitterSpy

O serviço *TwitterSpy* é uma entidade completamente independente do sistema, este foi implementado à custa de um *Worker Role* e recebe os pedidos de novos estudos através de uma Queue.

Para o acesso à *stream* *api* do *Twitter* foi utilizado a *SDK* *Tweetin*, de forma a facilitar o desenvolvimento do serviço.

A interacção com a *stream api* não é o normal pedido -> resposta, neste caso existe a possibilidade de criar várias variantes de *streams* de forma a receber em “tempo real” os *tweets* de todo o mundo. Para este serviço foi utilizada uma *FilteredStream*, para que o *Twitter* filtrasse os *tweets* que envia para o serviço. Uma das limitações desta api e que apenas permite uma *stream* por aplicação (do *twitter*), significando que o serviço terá de gerir vários estudos sob uma única *stream*.

Devido às limitações o serviço *TwitterSpy* utiliza duas *Threads*, uma que faz *pooling* sobre a *queue* de forma a receber novos pedidos e outra para a *stream* do *twitter*.

A SDK utilizada não permite adicionar novos tópicos de filtragem após a stream estar em funcionamento, portanto foi criado um sistema que reinicia-se a stream cada vez que estudo começa ou acaba, de forma a retirar ou adicionar os tópicos.

O funcionamento do *TwitterSpy* é o seguinte:

* O subscritor da *queue* recebe uma nova mensagem e valida-a.
* Após validação são extraídos os tópicos que são necessários subscrever.
* É agendado uma terminação do estudo, ou seja, utilizando um timer.
* Após o estudo estar completo é realizado um relatório do mesmo e entregue ao serviço *BrandAnalytics* para que este seja incorporado no sistema.

Um dos requisitos do serviço *TwitterSpy* é que fosse reportado as *hashtags* mais utilizadas, para tal cada vez que um novo *tweet* é processado pelo serviço este guarda num histograma (único a cada estudo) as *hashtags* utilizadas, bem como o autor.