

ISEL

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Trabalho Prático

Computação Orientada a Serviços

Inverno 2013/14

Alunos

Celso Fernandes nº 29580

Docente

Eng.^a Cátia Vaz

Isel, 5 de dezembro de 2013

Índice

<u>1. INTRODUÇÃO</u>	<u>4</u>
1.1. NOTAS DE UTILIZAÇÃO	4
<u>1. ARQUITETURA</u>	<u>5</u>
<u>2. BRANDANALYTICS.DATA</u>	<u>6</u>
<u>3. TWITTERSPY</u>	<u>7</u>
3.1. TWITTERSERVICES	7
3.2. TWITTERACTIVITIES	7
3.3. WORKFLOW	8
<u>4. BRANDANALYTICS</u>	<u>9</u>
4.1. DESCRIÇÃO	9
4.2. CONTRATO	9
4.3. ACTIVIDADES AUXILIARES	10
4.4. WORKFLOW	11
4.4.1. MÁQUINA DE ESTADOS	11
<u>5. BRANDANALYTICS.WEB</u>	<u>12</u>
<u>6. WINDOWS AZURE</u>	<u>13</u>

Figura 1 – Arquitetura da solução	5
Figura 2 – Esquema de dados.....	6
Figura 2 – Estrutura <i>TwitterSpy</i>	7
Figura 4 – Estrutura <i>TwitterServices</i>	7
Figura 5 – Estrutura <i>TwitterActivities</i>	8
Figura 6 – Contrato <i>TwitterSpy</i>	8
Figura 7 – Casos de uso <i>BrandAnalytics</i>	9
Figura 8 – Contrato <i>BrandAnalytics</i>	9
Figura 9 – Atividades auxiliares <i>BrandAnalytics</i>	10
Figura 10 – Máquina de estado <i>BrandAnalytics</i>	11

1. Introdução

Este trabalho tem como objetivo criar um protótipo duma aplicação que efetua estudos de mercado, com base nas mensagens do *twitter*.

O código está disponível no “GitHub”, no seguinte endereço.

<https://github.com/celsoaf/BrandAnalytics>

1.1. **Notas de utilização**

Para motivos de simplicidade do protótipo é criado um utilizador “admin”, com password “admin”. Este utilizador tem permissões de funcionário. A plataforma permite o registo de novos utilizadores que vão ter permissões de cliente.

Ao arrancar o projeto web, é criada uma base de dados “*BrandAnalyticsDB*”, na instancia “*SQLExpress*”. Para alterar a localização da base de dados basta alterar as “*connectionstrings*” nos “*web.configs*”

1. Arquitetura

A arquitetura da solução está implementada em quatro *assemblies*, como mostra a figura 1.

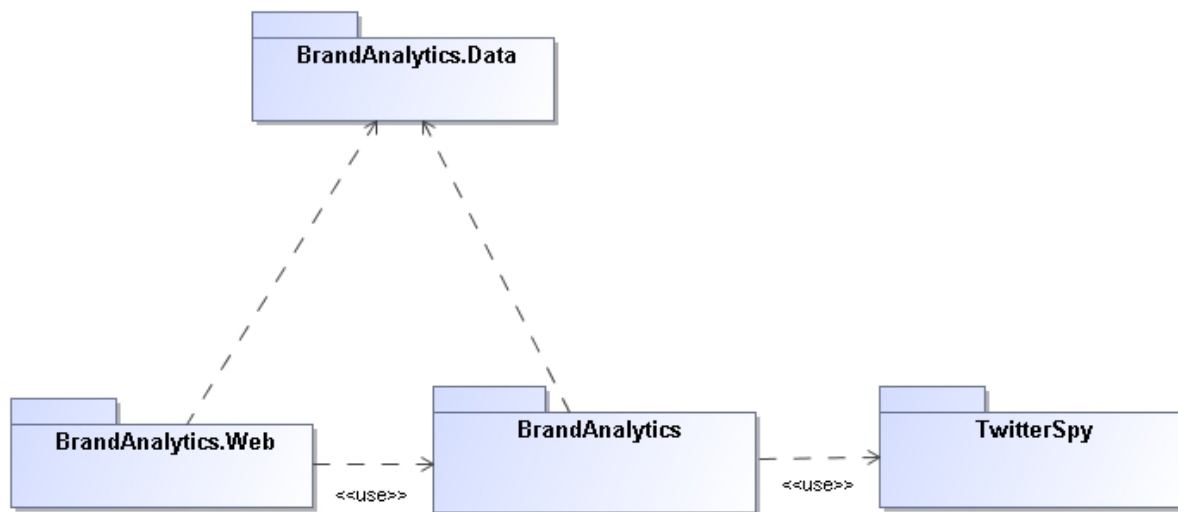


Figura 1 – Arquitetura da solução

Assembly	Descrição
BrandAnalytics.Data	Camada de acesso a dados.
BrandAnalytics.Web	Cliente web usa o serviço e o acesso a dados
BrandAnalytics	Implementa o workflow do protótipo pedido
TwitterSpy	Implementa o workflow que controla o acesso ao Twitter Streaming API.

2. BrandAnalytics.Data

A figura 2 mostra a estrutura de dados implementada, os dados são guardados na base de dados através da “*EntityFramework*”, com a técnica do “*Code-First*”.

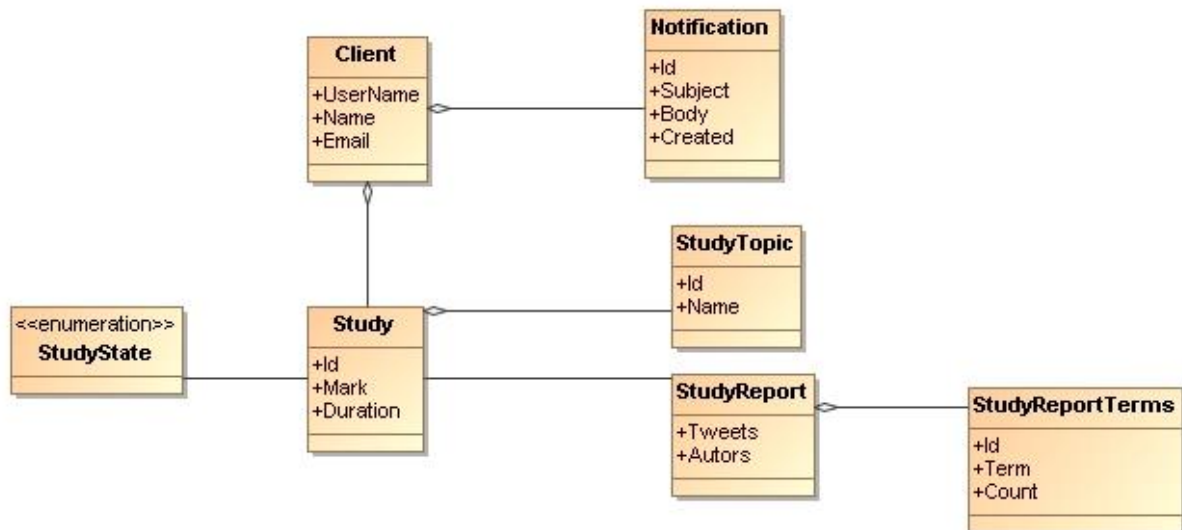


Figura 2 – Esquema de dados

Por motivos de simplicidade do protótipo o empregado também é um cliente. Uma evolução futura o empregado derivava de Cliente.

3. TwitterSpy

A implementação do *TwitterSpy* está baseada em três grupos, como mostra a figura 3.



Figura 3 – Estrutura *TwitterSpy*

3.1. *TwitterServices*

A implementação da parte “*TwitterServices*”, é composta por 3 classes, com a estrutura apresentada na figura 3. Tem como responsabilidade recolher informação dos “*Tweets*”.

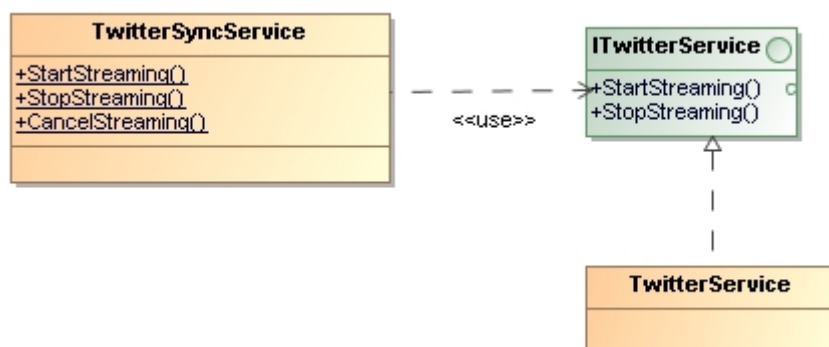


Figura 4 – Estrutura *TwitterServices*

Classe	Descrição
TwitterService	Esta classe utiliza a funcionalidade “ <i>Streaming</i> ” da biblioteca “ <i>LinqToTwitter</i> ”. Esta funcionalidade baseia-se na arquitetura “ <i>Event Based</i> ”, recebendo os “ <i>Tweets</i> ” publicados enquanto o canal está aberto.
TwitterSyncService	Esta classe controla o acesso ao “ <i>TwitterService</i> ”, visto que a biblioteca não suporta concorrência, implementa o padrão “ <i>Singleton</i> ” e gere de forma sincronizada uma lista de pedidos. Os métodos recebem um “ <i>token</i> ” para poder associar as chamadas.

3.2. *TwitterActivities*

A implementação da parte “*TwitterActivities*”, é composta pelas classes apresentadas na figura 5.

As classes “*TwitterStartActivity*”, “*TwitterStopActivity*” e “*TwitterCancelActivity*” fazem a ponte entre o “*TwitterServices*” e o “*Workflow*”, a classe “*TwitterReportActivity*”, gera um “*report*” do resultado dos vários tópicos recolhidos.

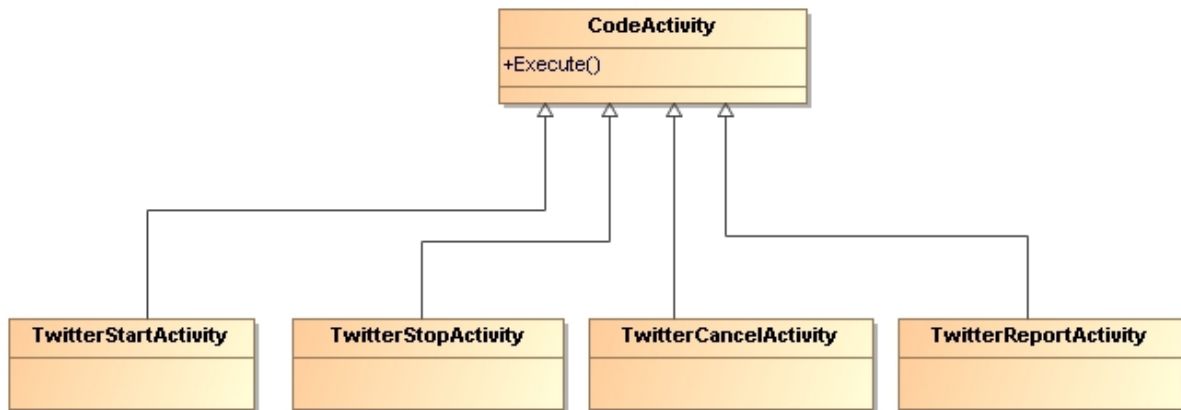


Figura 5 – Estrutura *TwitterActivities*

3.3. Workflow

Na parte do “*Workflow*” faz a implementação do contrato de “WCF” do contrato da figura 6.



Figura 6 – Contrato *TwitterSpy*

O “*Workflow*” está implementado de forma a fazer um ciclo para recolha dos “*Tweets*” para todos os tópicos e no final efetuar um relatório com os resultados. Este ciclo é feito em paralelo de forma a permitir o cancelamento a qualquer momento. O ciclo está dentro um “*CancellationScope*” de forma a permitir cancelar a recolha de dados do serviço “*TwitterService*”, caso seja invocado o método “*CancelSpyTopics*”.

4. BrandAnalytics

4.1. Descrição

A implementação da componente “*BrandAnalytics*” é baseada no esquema de casos de estudo apresentados na figura 7.

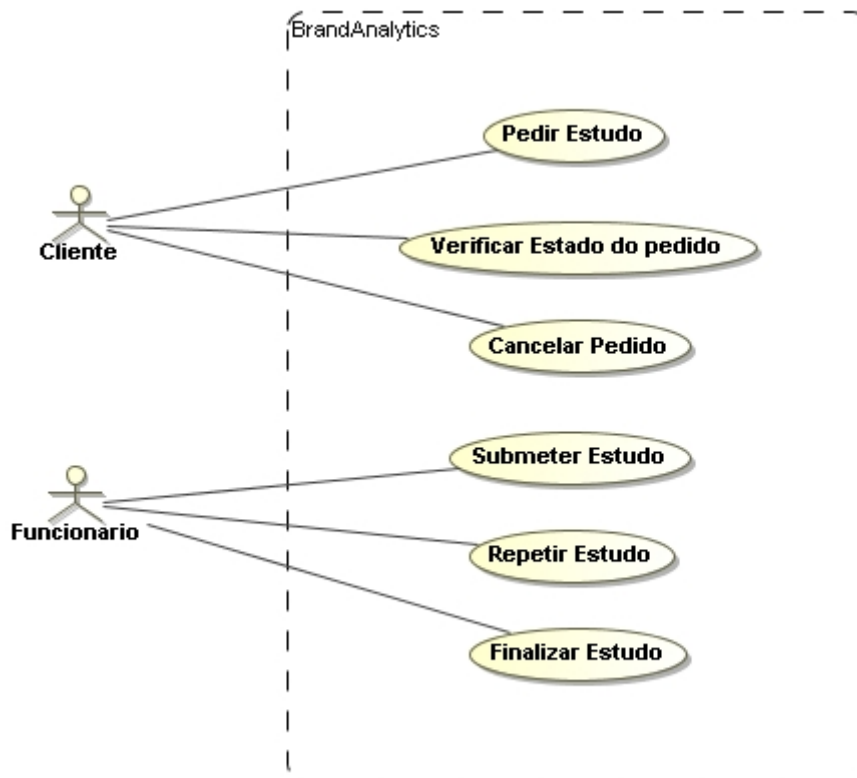


Figura 7 – Casos de uso *BrandAnalytics*

4.2. Contrato

Os casos de uso estão implementados através dum serviço *WCF* com o contrato apresentado na figura 8. O controlo de acessos é feito na componente *web site*.

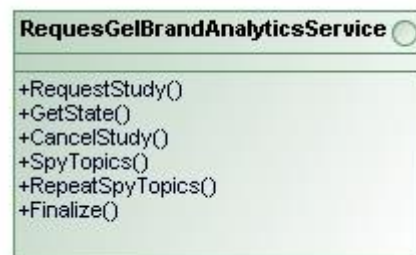


Figura 8 – Contrato *BrandAnalytics*

Na primeira implementação tentou se separar as funcionalidades em dois contratos, mas devido a um problema de geração do *proxy* do lado do site, em que só se conseguia importar um dos interfaces, resolveu-se juntar os contratos num só.

4.3. Atividades Auxiliares

Para auxiliar à implementação do *workflow* foram criadas atividades apresentadas na figura 9.

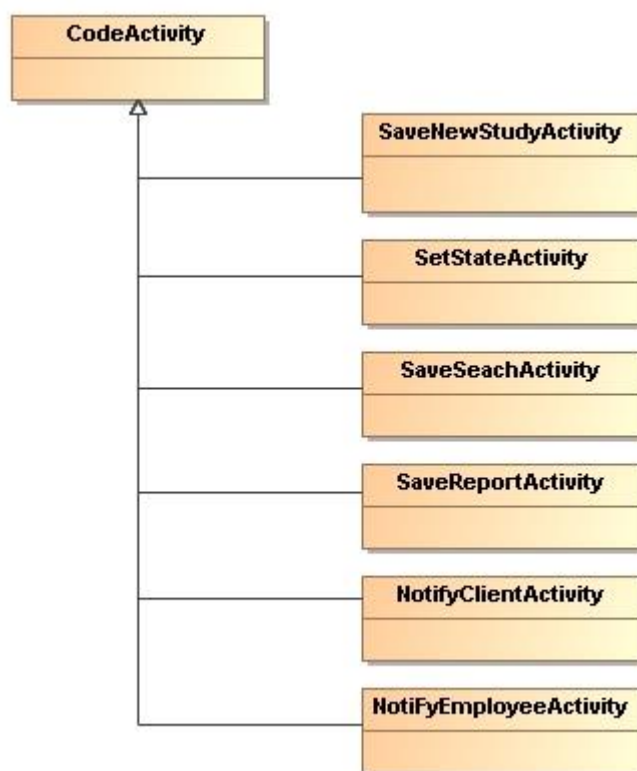


Figura 9 – Atividades auxiliares *BrandAnalytics*

Atividade	Descrição
SaveNewStudyActivity	Guarda o pedido do estudo do cliente na base de dados
SetStateActivity	Altera o estado o do estudo na base de dados
SaveSearchActivity	Guarda os dados dos tópicos que vão ser estudados no TwitterSpy
SaveReportActivity	Guarda o relatório devolvido pelo TwitterSpy na base de dados
NotifyClientActivity	Insere uma notificação para o cliente na base de dados
NotiFyEmployeeActivity	Insere uma notificação para o empregado na base de dados

4.4. Workflow

O *workflow* foi implementado com base num “*Paralell*”, que contem uma máquina de estados, um ciclo para responder aos pedidos de estado do estudo e uma sequência para cancelar o estudo.

4.4.1. Máquina de estados

A máquina de estados foi implementada com base no diagrama apresentado na figura 10.

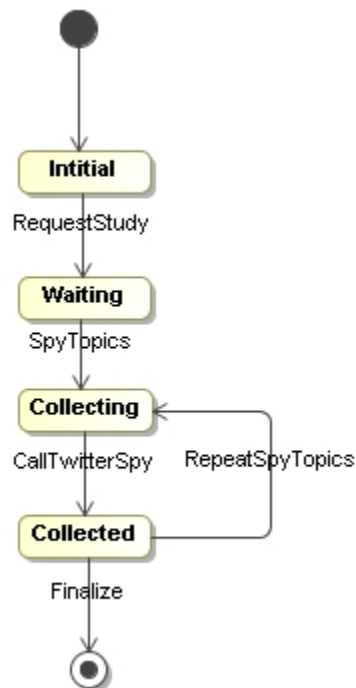


Figura 10 – Máquina de estado *BrandAnalytics*

As transições de estado são efetuadas pela chamada aos métodos do contrato WCF, exceto o “*CallTwitterSpy*” que é uma chamada ao WCF exposto pelo “*TwitterSpy*”. Este está dentro dum “*CancellationScope*” de forma a permitir que o estudo seja cancelado.

5. BrandAnalytics.Web

Na implementação do cliente foi selecionado um cliente *ASP.NET MVC 4*, visto ser um cliente web de fácil publicação para o *Azure*, assim como os conhecimentos adquiridos, tornam mais simples esta implementação.

Na implementação foram criados quatro *controllers*.

Controller	Descrição
StudyClient	Permite criar novos estudos, listar e cancelar projetos
StudyEmployee	Permite submeter estudos para o <i>TwitterSpy</i> , repetir o estudo ou finalizar o estudo
Report	Permite ver os detalhes do resultado do estudo
Notification	Permite listar todas as notificações do cliente/funcionário

Nota: Em Debug é possível o cliente eliminar estudos

6. Windows Azure

Com o objetivo de fazer publicar a solução para o *Azure* foi criado um projeto “*BrandAnalytics.Azure*”, foram adicionados três *web roles* ao projeto.

A solução funciona no simulador, no entanto não foi possível publicar, devido a conta já ter expirado. Caso ainda fosse possível aceder à conta teria de separar os roles em projetos *Azure*, de forma a poder definir os endereços dos *WCFs*.

Teria ainda de criar uma conta de *Storage* para usar o *SQL-Azure* como servidor de base de dados.