Computação na Nuvem (LEIRT e LEIC) – Versão 2019/2020

Síntese de atividades de laboratório

Laboratório nº: 5

Data: domingo, 10 de maio de 2020

Turma: 61D

Grupo: **06**

Número e nome dos alunos presentes:

Número	*Nome*
43861	Francisco Chicharro
43874	João Florentino
Click or tap here to enter text.	Click or tap here to enter text.

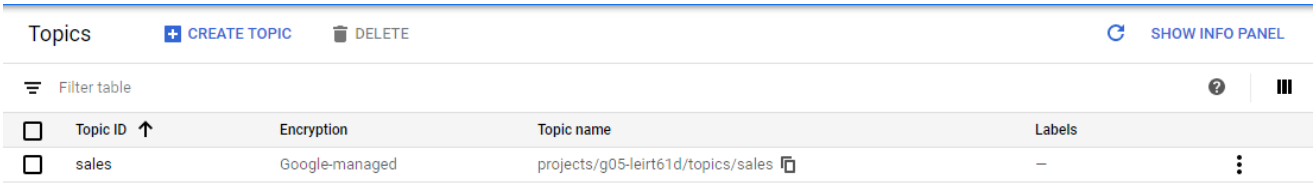
1. Objetivo da atividade (descrição por palavras simples do que entendeu como objetivo da atividade);
   - ➔ Saber usar a API Java no acesso ao serviço Google Pub/Sub.
   - ➔ Compreender as diferenças entre tópico e subscrição e as garantias de entrega de mensagens.
   - ➔ Saber integrar o Pub/Sub com outros serviços.
2. Indicação das tecnologias e as ferramentas (*tools*) utilizadas;
   - ➔ Big Data – Pub/Sub da Google Cloud Platform
   - ➔ Intellij IDEA
3. Descrição da arquitetura das partes (componentes) envolvidas, com eventuais diagramas:

Alínea A)

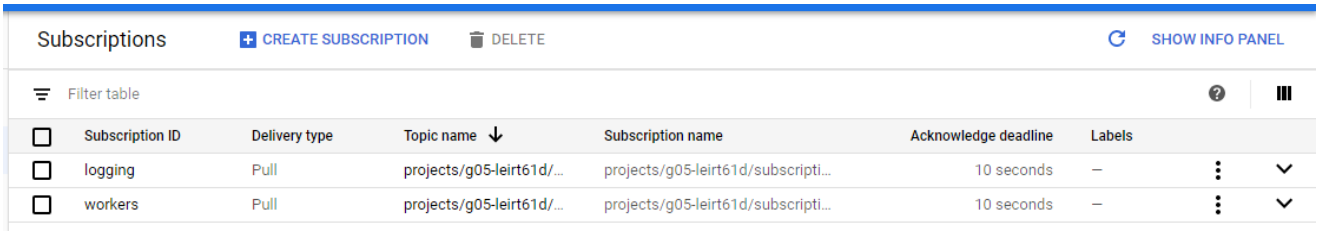
Considerando um cenário de um supermercardo de bens alimentares com múltiplas caixas de pagamento de produtos. Por cada venda paga nas caixas é produzida a seguinte informação (Em anexo, como exemplo, existe um ficheiro CSV com dados de venda de produtos):

Caixa	Item	Quant.	Preço Unit.	Total
Cx1	Arroz Agulha	2	0,99	1,98
Cx1	Água sem Gás	6	0,39	2,34
Cx50	Laranjas	2	0,9	1,8
...	...	...	...	...
Cx121	Lâmpada Led	3	2,99	8,97

Utilizando a consola da Google Cloud Platform, realizou-se inicialmente a criação de um tópico de nome sales:

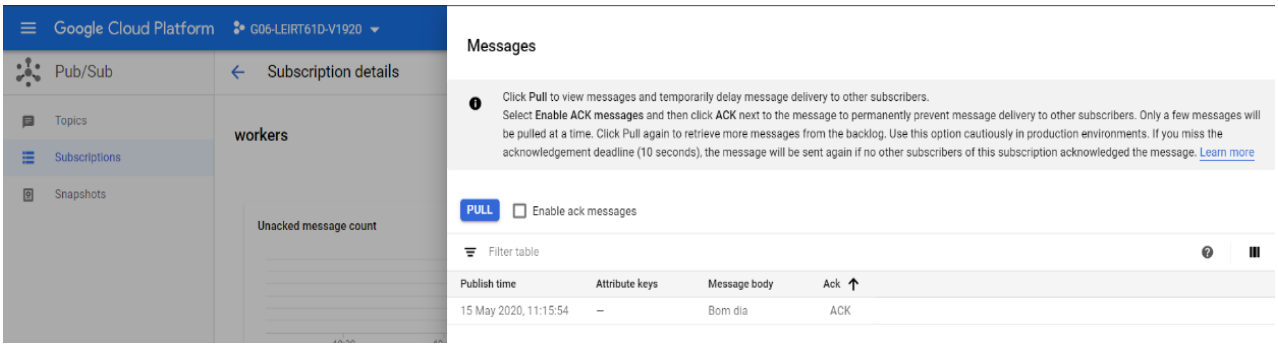


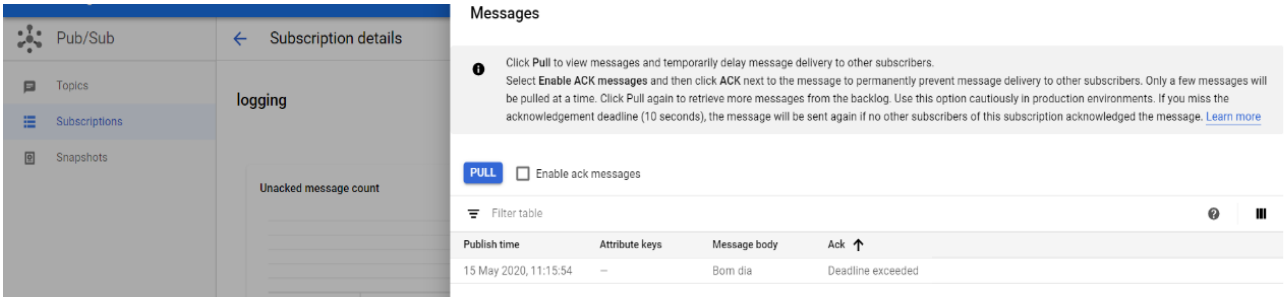
Associado a este tópico, colocou-se duas subscrições “logging” e “workers”



No tópico sales, publicou-se uma mensagem com o texto “Bom dia”, verificando que ambas as subscrições receberam a mensagem. As subscrições obtêm a mensagem através de um pedido pull, sendo que as subscrições vão receber a mesma mensagem.

Para que esta situação aconteça, é necessário que as duas subscrições já estejam previamente criadas, caso contrário, a mensagem do tópico sales não iria aparecer na subscrição se esta tivesse sido criada posteriormente ao envio da mensagem





Alínea B)

Pretende-se criar um sistema que integre o serviço Pub/Sub assim como o serviço de Firestore, composto por três aplicações: Caixa, loggerApp e workerApp.

Inicialmente, foi necessário criar uma conta de serviço com permissões á Firestora e ao serviço PubSub, onde, como já feito anteriormente, as credenciais são iniciadas através de uma instância de Google Credentials. Para o serviço de Firestore, foi também necessário criar uma coleção para guardar os documentos que seriam criados.

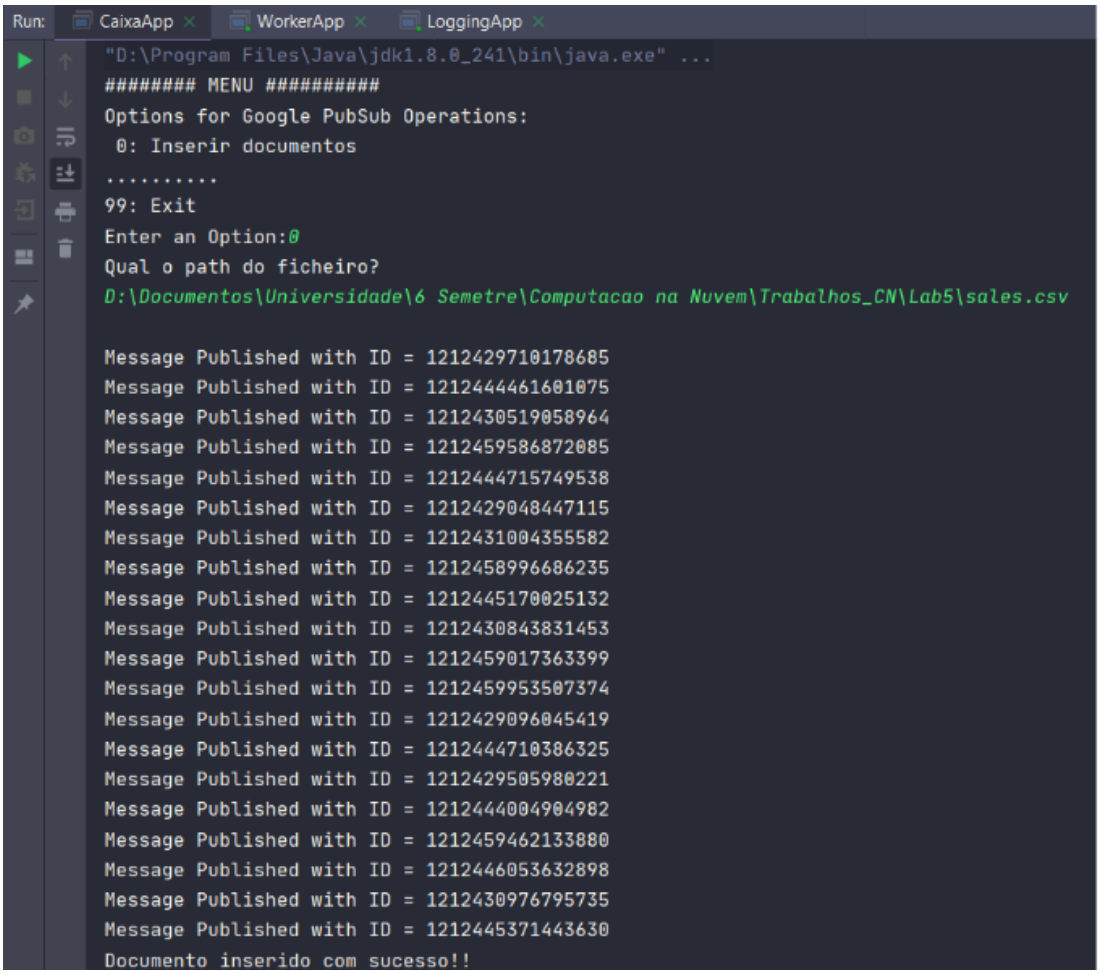
Para permitir o acesso ao serviço Pub/Sub, criou-se também uma varíavel com o identificador do projeto criado na GCP.

Aplicação Caixa:

Nesta aplicação, apenas se criou um menu onde o utilizador iria escolher as opções que desejava executar, neste caso, a única operação de Inserir Documentos, sendo necessário passar o pathname do ficheiro CSV fornecido pelos docentes.

O grupo criou uma classe CnUtils, e é esta classe que terá as operações que vão dar suporte ao menu da aplicação caixa: É então nesta classe que se irá realizar uma publicação do tópico sales, criando um objeto TopicName e e Publisher. O ficheiro CSV é tratado aqui, sendo a cada linha lida, é criada uma mensagem com os atributos pedidos no enunciado que será publicada. A mensagem é identificada por um identificador único que será gerado a cada iteração para a criação da mensagem:

Executando então a aplicação Caixa, obtemos a publicação de sucessivas mensagens criadas em CnUtils que são publicadas no tópico, identificadas por um ID único:



Aplicação logger:

Esta aplicação pretende simular um subscritor que possui uma subscrição no tópico Sales, sendo esta subscrição do tipo Logging.

Assim, nesta aplicação tal como na aplicação Caixa, é apenas chamada a função que permite obter as mensagens da subscrição pela função getMessagesLogger(), sendo que se recorreu novamente ao CnUtils para criar esta função.

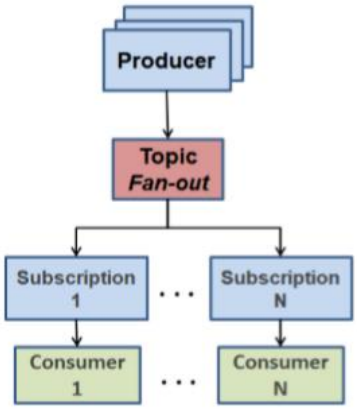
A função getMessagesLogger() , através da criação de um objeto ProjectSubscriptionName e Subscriber e recebendo o MessageReceiveLoggerHandler, tem o objetivo de esperar por uma mensagem publicada, e assim que essa chegue imprime-a no standard output mostrando a mensagem ao subscritor do tipo Logging.

O messageReceiverLoggerHandler tem a função receiveMessage onde se obtem o conteúdo da mensagem que se quer mostrar ao subscritor. É enviado um acknowledge de confirmação de receção de mensagem.

Executando assim a Aplicação logger, o subscritor tem acesso ás mensagens publicadas:

```
"D:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
######## MENU ##########
Options for Google PubSub Operations:
 0: Get Messages
.........
99: Exit
Enter an Option:0
Qual o nome da subscription?
logging
To stop press enter
Message (Id: 1212429710178685Data: Cx1,Arroz Agulha,2,0.99,1.98)
Message (Id: 1212444461601075Data: Cx1,�gua sem G�s,6,0.39,2.34)
Message (Id: 1212430519058964Data: Cx2,Leite Meio-gordo,6,0.59,3.54)
Message (Id: 1212459586872085Data: Cx3,Limpa Vidros,2,1.94,3.88)
Message (Id: 1212444715749538Data: Cx3,L�mpada Led,3,2.99,8.97)
Message (Id: 1212429048447115Data: Cx4,Caf� Soluvel,1,3.24,3.24)
Message (Id: 1212431004355582Data: Cx5,Bolachas Soja,4,1.19,4.76)
Message (Id: 1212458996686235Data: Cx100,Couve ,1,1.76,1.76)
Message (Id: 1212445170025132Data: Cx100,Alface,2,1.25,2.5)
Message (Id: 1212430843831453Data: Cx100,Ma�as,1,2.9,2.9)
Message (Id: 1212459017363399Data: Cx100,Laranjas,3,1.1,3.3)
Message (Id: 1212459953507374Data: Cx200,Papo Seco,12,0.1,1.2)
Message (Id: 1212429096045419Data: Cx50,Agua Monchique,2,1.25,2.5)
Message (Id: 1212444710386325Data: Cx53,Batatas,2,2.4,4.8)
Message (Id: 1212429505980221Data: Cx53,Couve ,1,1.6,1.6)
Message (Id: 1212444004904982Data: Cx53,Laranjas,2,0.9,1.8)
Message (Id: 1212459462133880Data: Cx55,Caf� ,1,3.8,3.8)
Message (Id: 1212446053632898Data: Cx10,Laptop,1,443,443)
Message (Id: 1212430976795735Data: Cx11,Impressora,1,125,125)
Message (Id: 1212445371443630Data: Cx12,Cabo HDMI,1,6.9,6.9)
```

Esta aplicação segue um padrão fan-out pattern onde todos os consumidores recebem todas as mensagens que são enviadas:



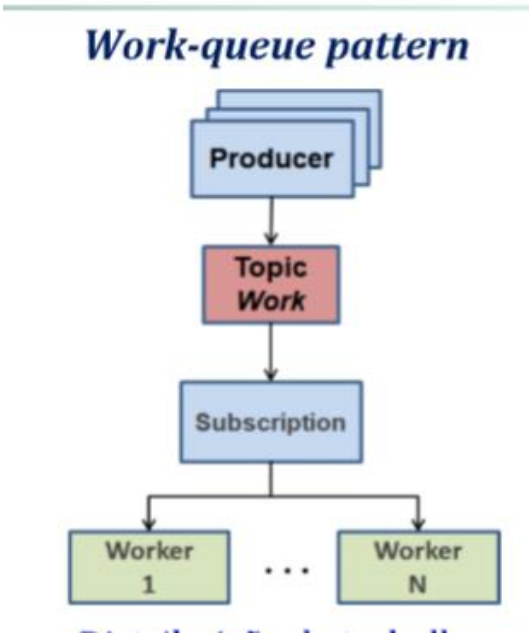
Aplicação Worker:

Esta aplicação segue o mesmo objetivo do LoggerApp, no entanto foram criados vários subscritores workers em que cada um recebe mensagens distintas e acrescentando também a condição de iniciar o acesso á coleção criada anteriormente na Firestore, onde serão armazenadas as mensagens publicada pelo tópico sales e obtidas pelos subscritores workers.

Assim, a implementação desta aplicação segue o mesmo seguimento da aplicação Logger onde se iniciando novamente objetos ProjectSubscriptionName e Subscriber, recebe-se um novo MessageReceiveHandlerWorker que dada a situação, irá tratar da necessidade de criar múltiplos subscritores, assim como a colocação das mensagens recebida na coleção “subscription-worker” do firestore, criando assim um documento nesta base de dados.

É então a função getMessagesWorker() criada na classe CnUtils que irá obter as mensagens que estão nos workers, apresentando-as no Firestore e também no standard output.

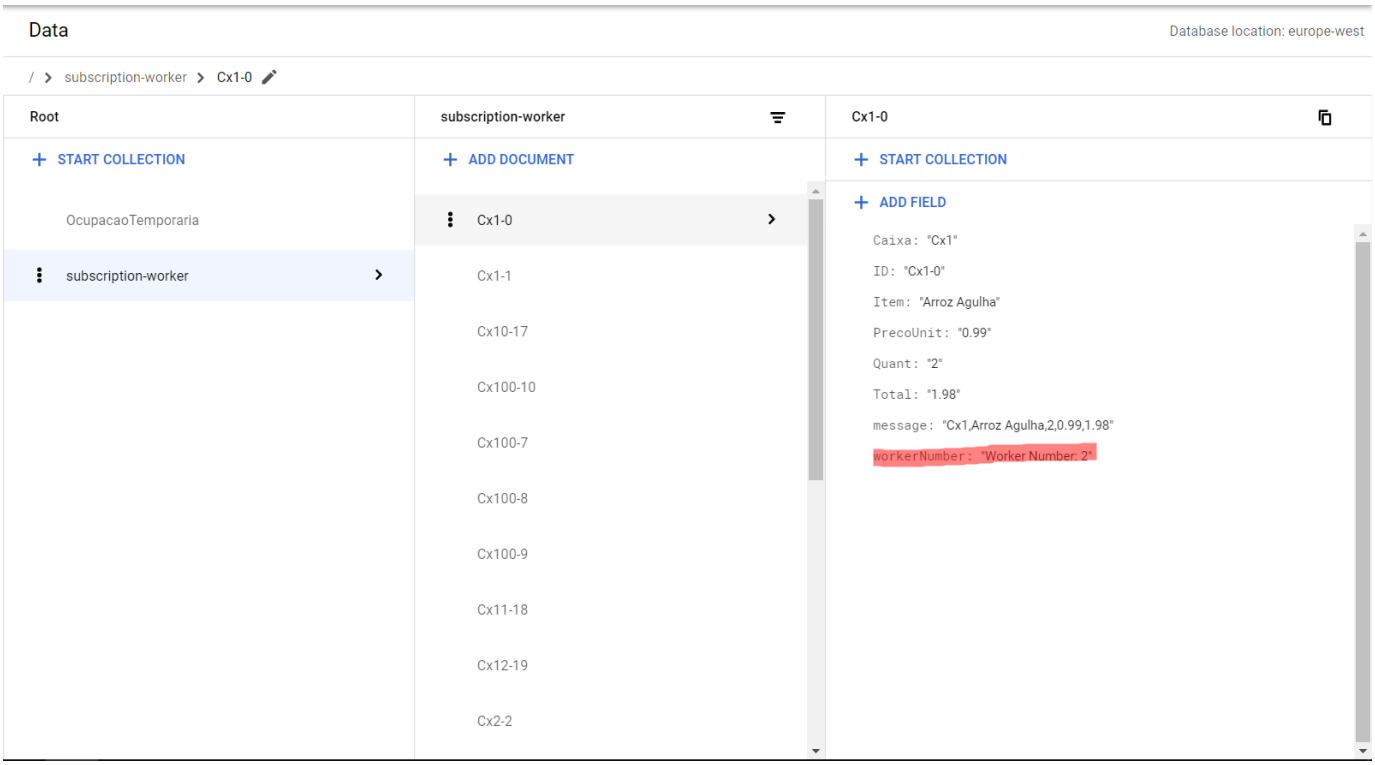
De notar que o padrão desta aplicação é do tipo Work-queue pattern, onde para cada subscrição estão associados múltiplos subscritores:

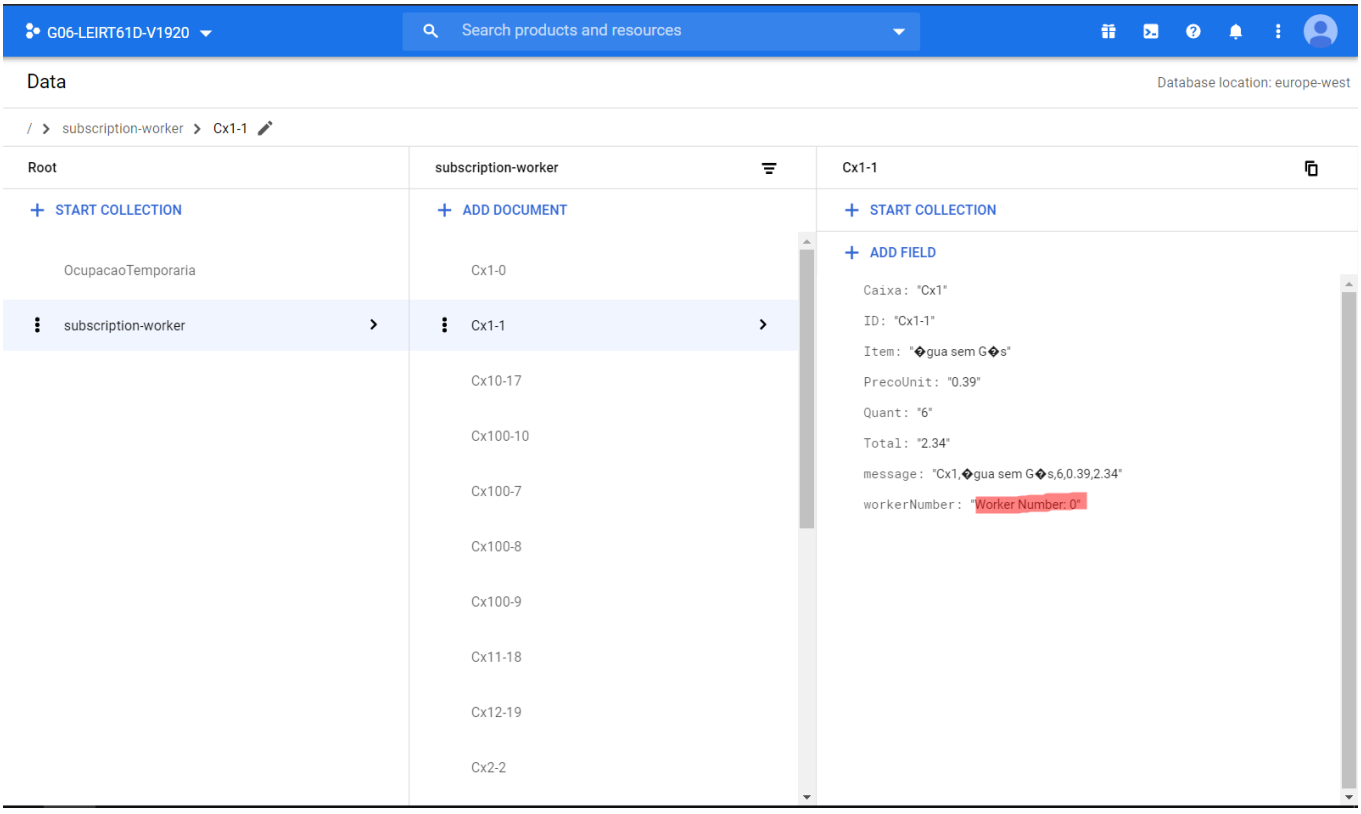


Testando a aplicação Workers, obtém-se as múltiplas mensagens dos múltiplos workers :

```
"D:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
######## MENU ##########
Options for Google PubSub Operations:
 0: Get Messages
.........
99: Exit
Enter an Option:0
Qual o nome da subscription?
workers
To stop press enter
Message (Id: 1212429710178685Data: Cx1,Arroz Agulha,2,0.99,1.98)
Message (Id: 1212444461601075Data: Cx1,�gua sem G�s,6,0.39,2.34)
Message (Id: 1212430519058964Data: Cx2,Leite Meio-gordo,6,0.59,3.54)
Message (Id: 1212459586872085Data: Cx3,Limpa Vidros,2,1.94,3.88)
Message (Id: 1212444715749538Data: Cx3,L�mpada Led,3,2.99,8.97)
Message (Id: 1212429048447115Data: Cx4,Caf€ Soluvel,1,3.24,3.24)
Message (Id: 1212431004355582Data: Cx5,Bolachas Soja,4,1.19,4.76)
Message (Id: 1212458996686235Data: Cx100,Couve ,1,1.76,1.76)
Message (Id: 1212445170025132Data: Cx100,Alface,2,1.25,2.5)
Message (Id: 1212430843831453Data: Cx100,Ma€as,1,2.9,2.9)
Message (Id: 1212459017363399Data: Cx100,Laranjas,3,1.1,3.3)
Message (Id: 1212459953507374Data: Cx200,Papo Seco,12,0.1,1.2)
Message (Id: 1212429096045419Data: Cx50,Agua Monchique,2,1.25,2.5)
Message (Id: 1212444710386325Data: Cx53,Batatas,2,2.4,4.8)
Message (Id: 1212429505980221Data: Cx53,Couve ,1,1.6,1.6)
Message (Id: 1212444004904982Data: Cx53,Laranjas,2,0.9,1.8)
Message (Id: 1212459462133880Data: Cx55,Caf€ ,1,3.8,3.8)
Message (Id: 1212446053632898Data: Cx10,Laptop,1,443,443)
Message (Id: 1212430976795735Data: Cx11,Impressora,1,125,125)
Message (Id: 1212445371443630Data: Cx12,Cabo HDMI,1,6.9,6.9)
```

Onde, no Firestore, se verifica que estão diferentes workers a apresentar diferentes mensagens da subscrição:





4. Resumo dos problemas encontrados e as soluções aplicadas:

   ➔ O teste dos vários workers não estava a executar corretamente pois haviam informações que ainda não tinham recebido acknowledge de entrega. Como solução para este problema, foram apagadas todas as mensagens por receber, na google cloud platform, e de seguida procedeu-se ao re-envio das mesmas.

5. Indicação se a solução final é executável e demonstrável

   ➔ A solução final é executável e demonstrável

6. Conclusões e lições aprendidas

   Com este trabalho aprendeu-se a implementar novas funcionalidades da GCP, nomeadamente, pub/sub e foi possível aprofundar os conhecimentos adquiridos no laboratório anterior sobre Firestore uma vez que foi pedido para implementar essa funcionalidade novamente para guardar as mensagens relativamente a subscrições dos workers.

7. Auto-avaliação qualitativa por parte dos alunos

   Bom