Relatório Técnico do Projecto

CleanSheets

Grupo 2

Elemento
João Dias 1100604

1 Introdução

1.1 Apresentação

O problema inicial é relativo à importação e actualização/inserção para Base de Dados.

1.2 Requisitos

- Ser possível ao utilizador indicar qual a área da folha que pretende actualizar/inserir numa tabela já existente;
- Ser possível ao utilizador importar informação de qualquer tabela já criada.
- Escolher de que SGBD pretende importar (MySQL, Postgres e SQLserver).

1.3 Objectivos

O objectivo do problema está centrado no facto de ser possível a actualização e importação a partir de vários SGBD, tendo como base, MySQL, Postgres e SQLserver.

1.4 Dificuldades

A principal dificuldade foi encontrar informação para conseguir adaptar o código JDBC para os vários SGBD, visto que as queries para, por exemplo, obter as chaves PKs ou o número de colunas são diferentes para cada SGBD. Outra dificuldade encontrada foi o facto de ter que se utilizar sequências em todos os sgbd, de maneira, a ser-se possível efectuar a importação de forma correcta e eficaz.

1.5 Estrutura do Relatório Técnico

- Introdução;
- Enquadramento;
- Análise;
- Concepção;
- Implementação:
- Conclusão;
- Bibliografia.

2 Enquadramento

2.1 Descrição dos Requisitos

Seleccionar a área a actualizar

O utilizador deverá indicar a área da folha que pretende gravar em formato string, exemplo : "A1-D4", o programa tratará de ir buscar os valores correctamente.

Escolher o nome da tabela e indicar os dados de conexão

Tal como requisito anterior, o utilizador terá que introduzir toda a informação necessária para que a conexão seja efectuada com êxito.

Escolher o SGBD que pretende gravar

O utilizador terá que indicar em que SGBD pretende gravar a sua informação, sendo que pode gravar nos 3 SGBD disponíveis sem qualquer problema.

Escolher o SGBD que pretende importar

O utilizador terá que indicar de que SGBD pretende importar a sua informação, e indicar os dados de conexão, mais o nome da tabela que pretende importar para a sua folha de cálculo.

2.2 Enquadramento do Projecto

Sendo que o projecto tem como requisito principal a opção de gravar e exportar conteúdo para vários tipos de base de dados, penso que esse propósito foi concluído com sucesso. Visto que neste momento, já faz isso, tendo ainda a opção de actualizar/inserir novos dados numa tabela já existente.

2.3 Funcionalidades

A principal funcionalidade é a exportação e importação a partir de uma Base de Dados. Outro requisito importante, é sem dúvida, a escolha do tipo de SGBD que pretende gravar/importar.

3 Análise

3.1 Requisitos Funcionais

Casos de uso

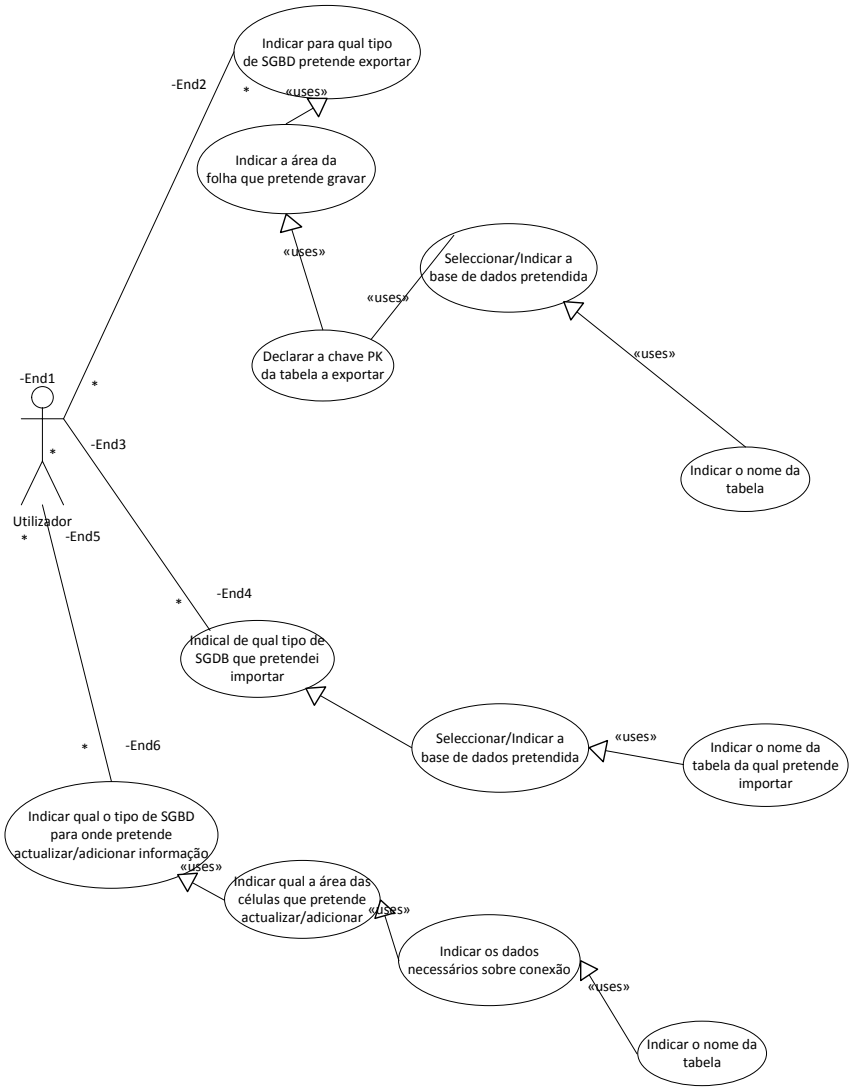


Diagrama de sequências

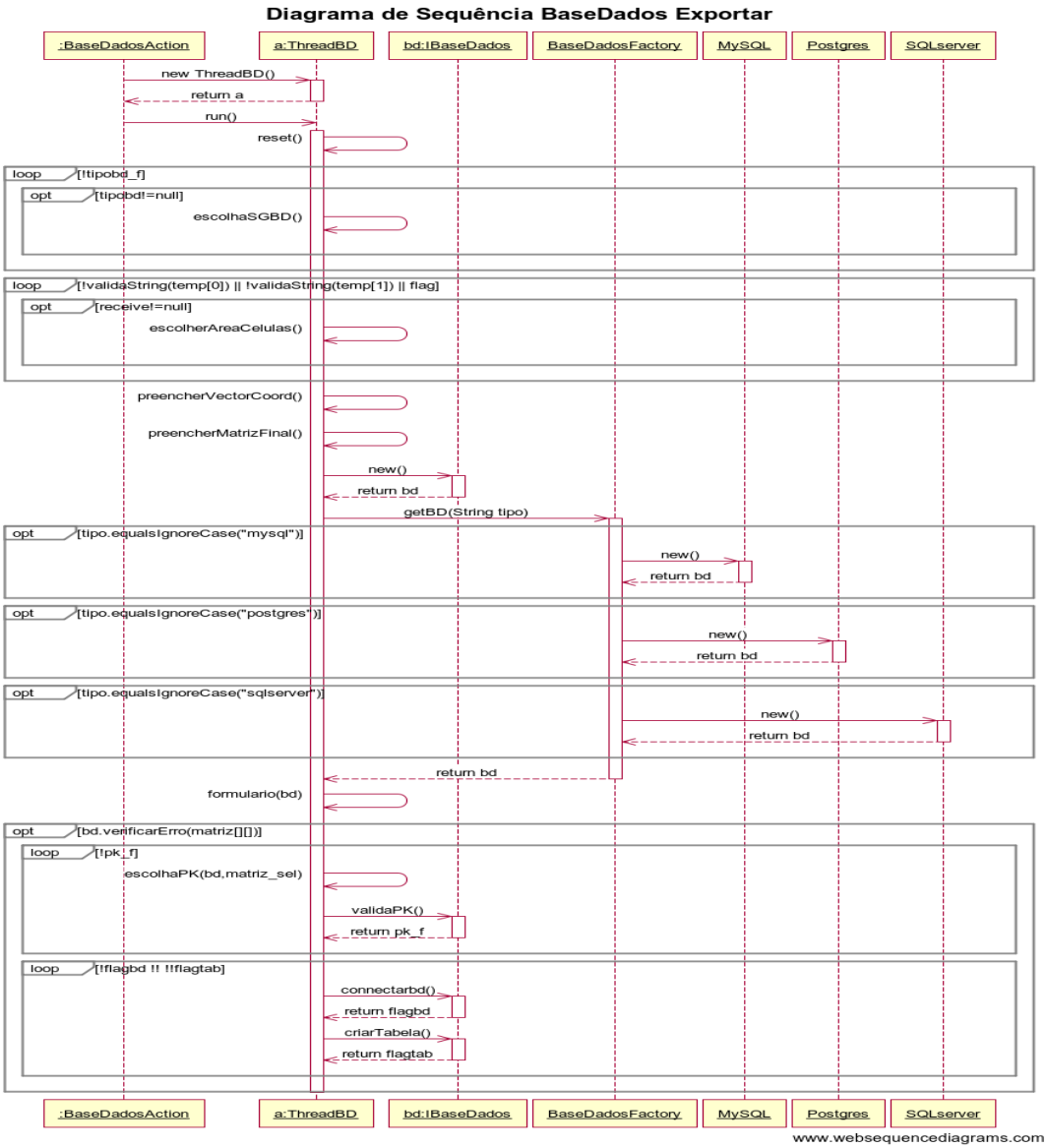
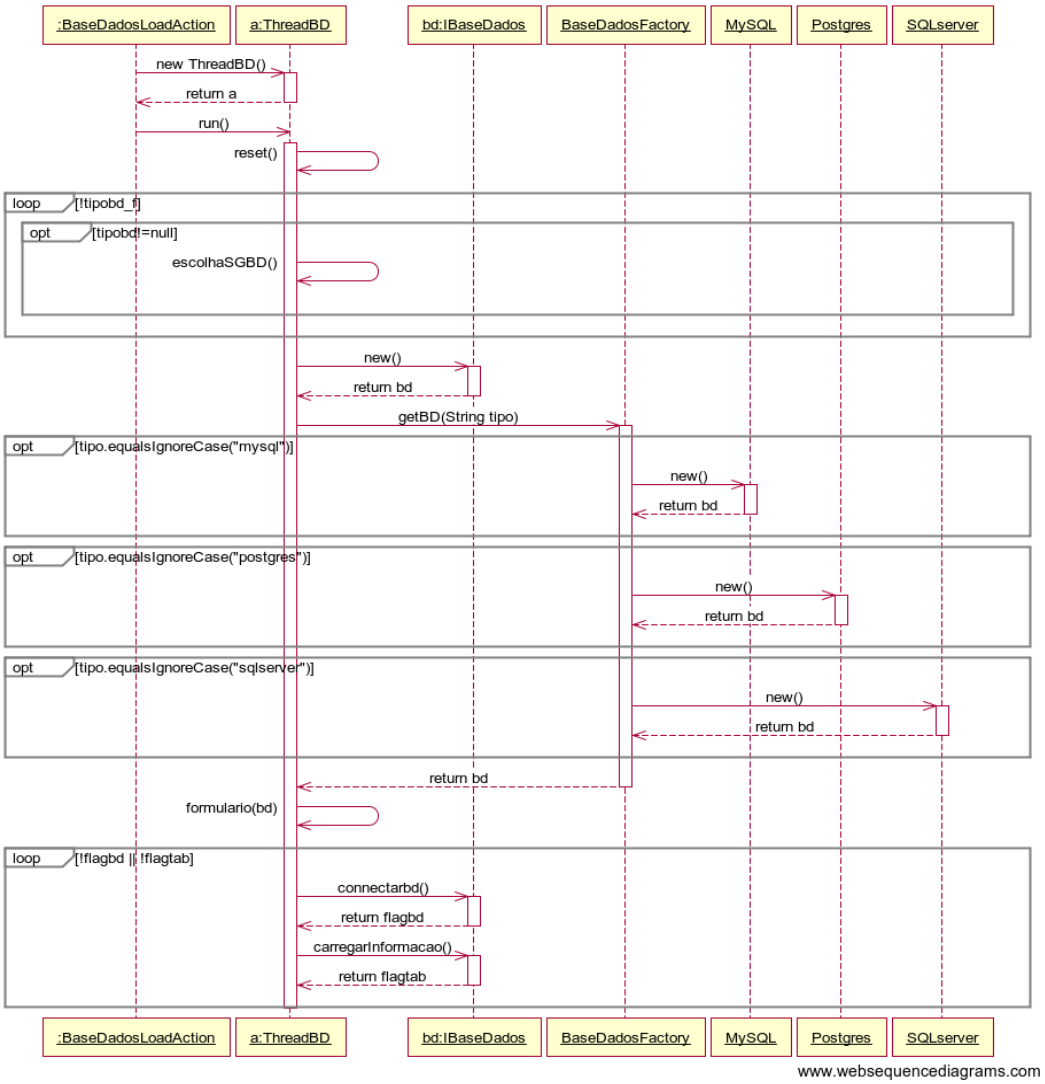
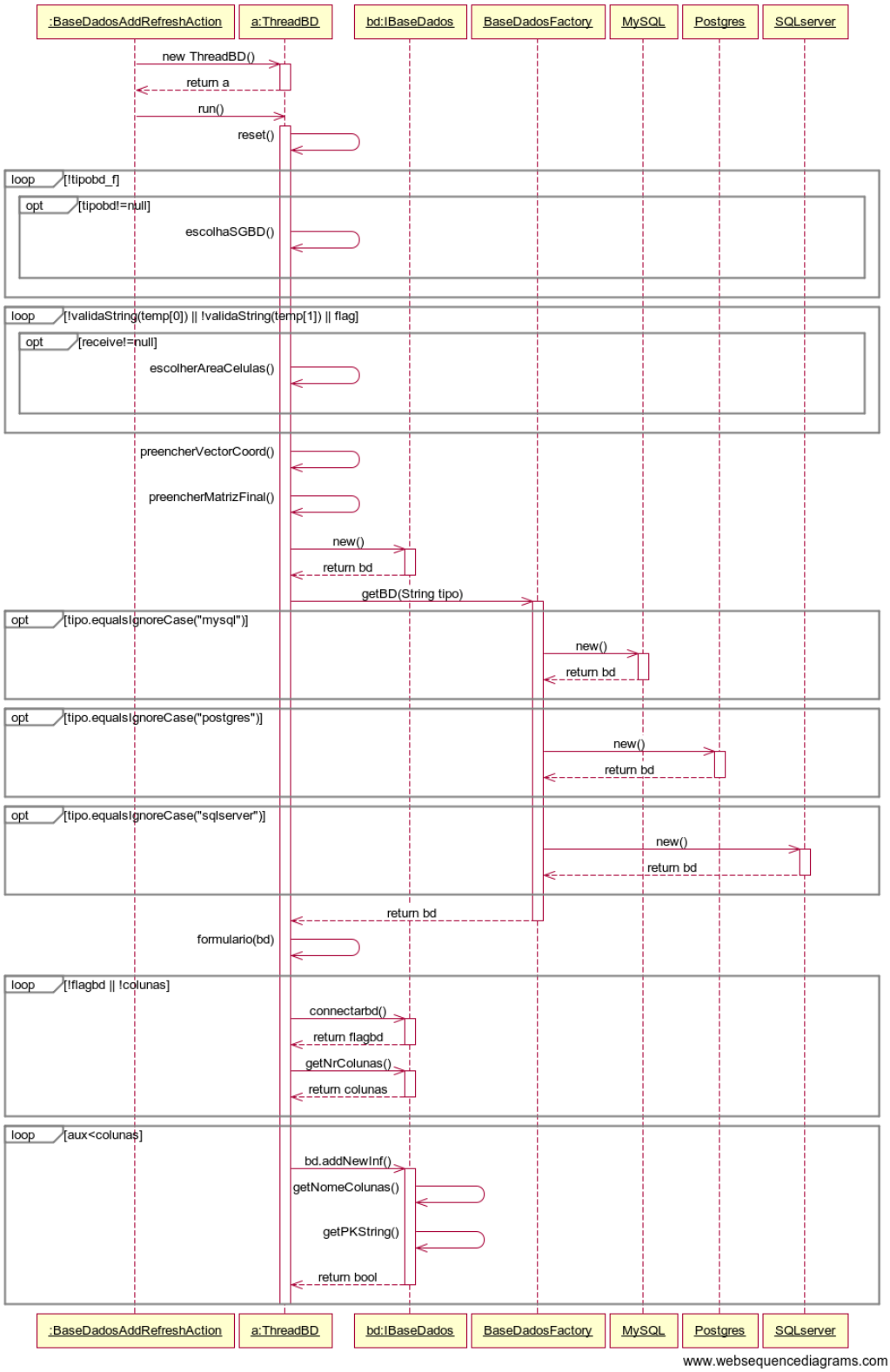


Diagrama de Sequência BaseDados Importar



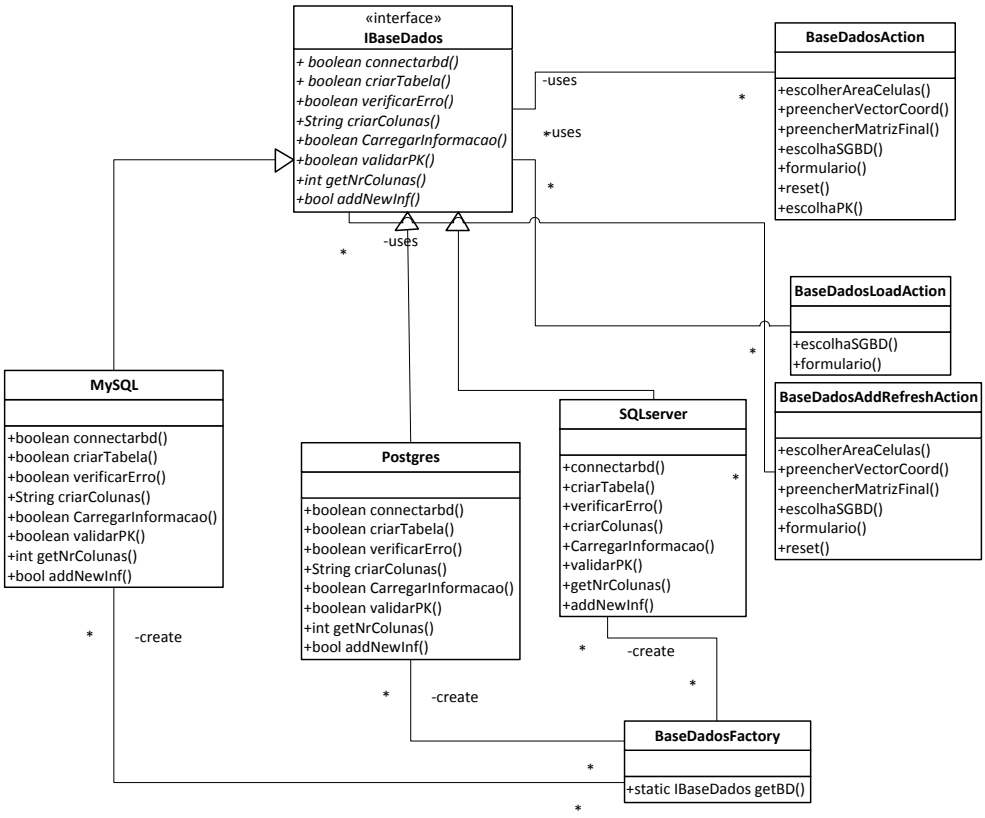
www.websequencediagrams.com

Diagrama de Sequência BaseDados Actualizar



www.websequencediagrams.com

Diagrama de Classes



3.2 Requisitos Não Funcionais

A utilização de três librarias para ser possível a conexão e exportação de dados.

- mysql-connector-java-5.1.20-bin
- sqljdbc4
- postgresql-9.1-902.jdbc4

3.3 Contextualização

A importação e exportação estão a funcionar correctamente.

3.4 Planeamento

- Importação de dados a partir de uma tabela;
- Declaração da chave PK da tabela;
- Actualização de campos de uma tabela;
- Inserção de novos dados numa tabela;
- Actualização de dados numa tabela.

4 Concepção

4.1 Especificação

Os mesmos da análise.

4.2 Aspectos técnicos da solução

Decidi separar cada caso de uso em classe, neste caso temos :

- BaseDadosAction - Exportar
- BaseDadosLoadAction – Importar
- BaseDadosAddRefreshAction - Actualizar

4.3 Testes

Os testes desta iteração são quase os mesmos da iteração anterior, tendo mais um método e uma alteração noutro. Visto que o código da iteração anterior foi reutilizado, e além do mais, para fazer-se os testes unitários aos novos métodos, para que estes fossem efectuados com sucesso, o programa teria que estar sempre conectado à base de dados. Sendo que esta premissa é impossível de realizar-se, decidi deixar os mesmos métodos. No entanto, deixo métodos que seriam casos prováveis de teste, getPkString() e getNrColunas().

4.3.1 Testes unitários

```
/**
 * Criar Coluna se tem A B C, o resultado esperado para receber a string é
 * A varchar(120), B varchar(120), C varchar(120)
  */
@Test
public void CriarColuna() throws SQLException, ClassNotFoundException {
    System.out.println("verificarErro");
    matriz[0][0]="A";
    matriz[0][1]="B";
    matriz[0][2]="C";
    matriz[1][0]="A";
    matriz[1][1]="B";
    matriz[1][2]="C";
    IBaseDados instance = BaseDadosFactory.getBD("mysql");;
    String expResult ="A varchar(120), B varchar(120), C varchar(120), )";

    String result = instance.criarColunas(matriz);
```

```
    assertEquals(expResult, result);

    // TODO review the generated test code and remove the default call to fail.

}

   /**
 * Verificar se a coluna 1 pode ser chave primária da tabela
  */
@Test
public void validarPk() throws SQLException, ClassNotFoundException {
    System.out.println("verificarErro");
    matriz[0][0]="Nome";
    matriz[0][1]="Nif";
    matriz[0][2]="Morada";
    String [] vec = new String [1];
    vec[0]="2";
    IBaseDados instance = BaseDadosFactory.getBD("mysql");
    boolean result = instance.validarPK(vec,matriz);
    boolean expResult = true;
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.

}

    /**
 * Verificar se a coluna 1 pode ser  2x chave primária da tabela
  */
@Test
public void validarPk_false() throws SQLException, ClassNotFoundException {
    System.out.println("verificarErro");
    matriz[0][0]="Nome";
    matriz[0][1]="Nif";
    matriz[0][2]="Morada";
    String [] vec = new String [2];
    vec[0]="2";
    vec[0]="2";
```

```
    IBaseDados instance = BaseDadosFactory.getBD("mysql");
    boolean result = instance.validarPK(vec,matriz);
    boolean expResult = false;
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.

}
```

4.3.2 Casos de teste

Semana 2
(4 de Junho a 10 de Junho)

Nome do caso de teste: *Core*
Casos de uso relacionados:

Comentário [Alexandre1]:

Objectivo	Pretende-se implementar uma nova linguagem de expressões que inicie pelo caractere "#" e que se baseie na língua portuguesa – nova gramática. Nesta iteração é pedido que a aplicação reconheça os comandos "#cell:=formula" e #{formula,formula,…}
Pré-requisitos	
Dados de teste	*.Comando inserido na barra principal, do tipo "#cell=formula"* *.Inserir sequência de comandos*
Passos	*1. Inserir fórmula no formato* *2. Verificar se o valor é o correcto*
Notas e Questões	

Resultados

#Execução	Dados	Resultados	Passou?	Observações
#1	*#A2:=Media(4;6;8)*	A2=6	Sim	
#2	#a45:=Soma(4;8)	A45=12	Sim	
#3	#{a1:=4;a2:=6;a3:=soma(a1;a2)}	A3=10	Sim	

5 Implementação

Não existem quaisquer dependências de ficheiros de configuração, visto que assim o utilizador é mais livre de escolher quais as bases de dados que pretende e quantas vezes pretende gravar. O código foi dividido em várias classes, e como pode ser visto no diagrama de classes, existe uma interface que define o comportamento de cada classe que implementa a mesma.

6 Conclusão

Penso que o que foi pedido para esta iteração foi concluído com sucesso, penso que também fui um bocado para além dos requisitos, visto que implementei com sucesso as sequências em cada tabela, de forma a que a importação fosse efectuada, tendo em conta a forma como foi exportada.

7 Bibliografia

- http://stackoverflow.com/questions/10904145/query-select-data-by-first-inserted
- http://www.coderanch.com/t/307291/JDBC/java/MySQL-auto-increment