Relatório Técnico do Projecto

CleanSheets

Grupo 2

Elemento
João Dias 1100604

1 Introdução

1.1 Apresentação

O problema inicial é relativo à exportação e importação para Base de Dados.

1.2 Requisitos

- Ser possível ao utilizador indicar qual a área da folha que pretende gravar;
- Ser possível ao utilizador escolher o nome da nova tabela e indicar os dados necessários para a conexão;
- Escolher em que SGBD pretende gravar (MySQL, Postgres e SQLserver).

1.3 Objectivos

O objectivo do problema está centrado no facto de ser possível a exportação 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, também foi difícil encontrar o software necessário para ir em encontro do que é pedido no enunciado.

1.5 Estrutura do Relatório Técnico

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

2 Enquadramento

2.1 Descrição dos Requisitos

Seleccionar a área a gravar

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.

2.2 Enquadramento do Projecto

O projecto tem como requisito principal a opção de gravar e exportar conteúdo para vários tipos de base de dados, uma das fronteiras que achei necessário não prestar tanta atenção e cuidado, foi a parte visual do programa, visto que o requisito principal do projecto é que o mesmo faço o que é pedido e que siga os padrões correctamente. De maneira, a que futuramente seja fácil introduzir mais tipos de SGBD.

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.

3 Análise

3.1 Requisitos Funcionais

Casos de uso

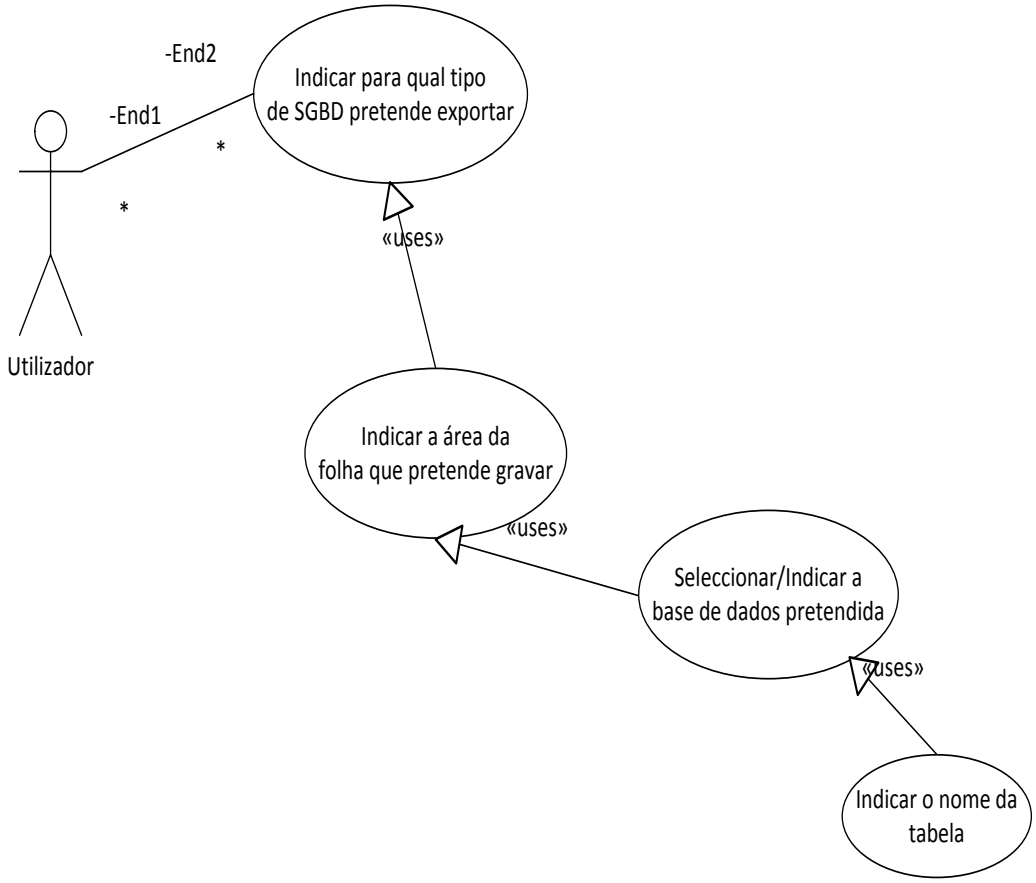


Diagrama de sequências

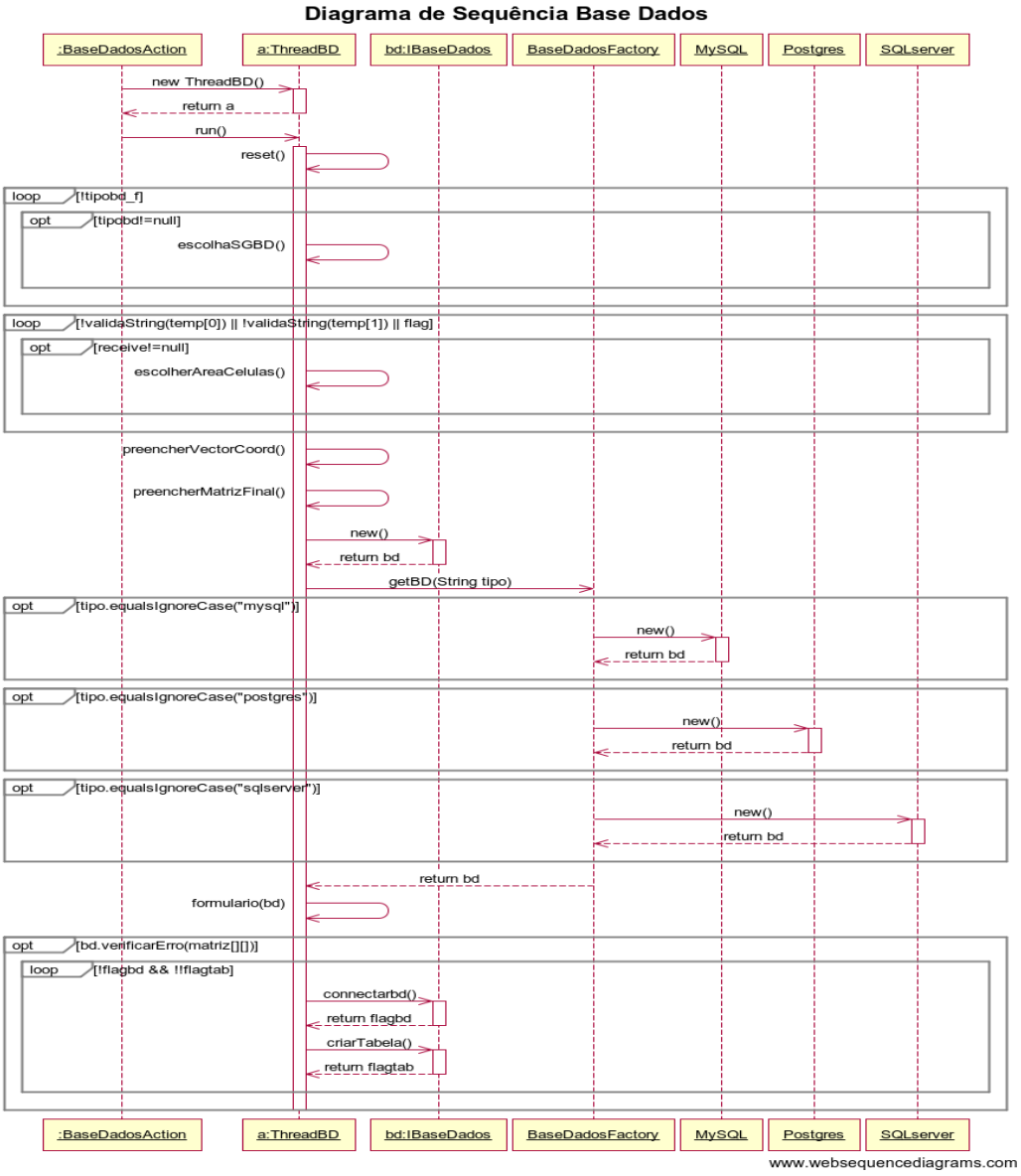
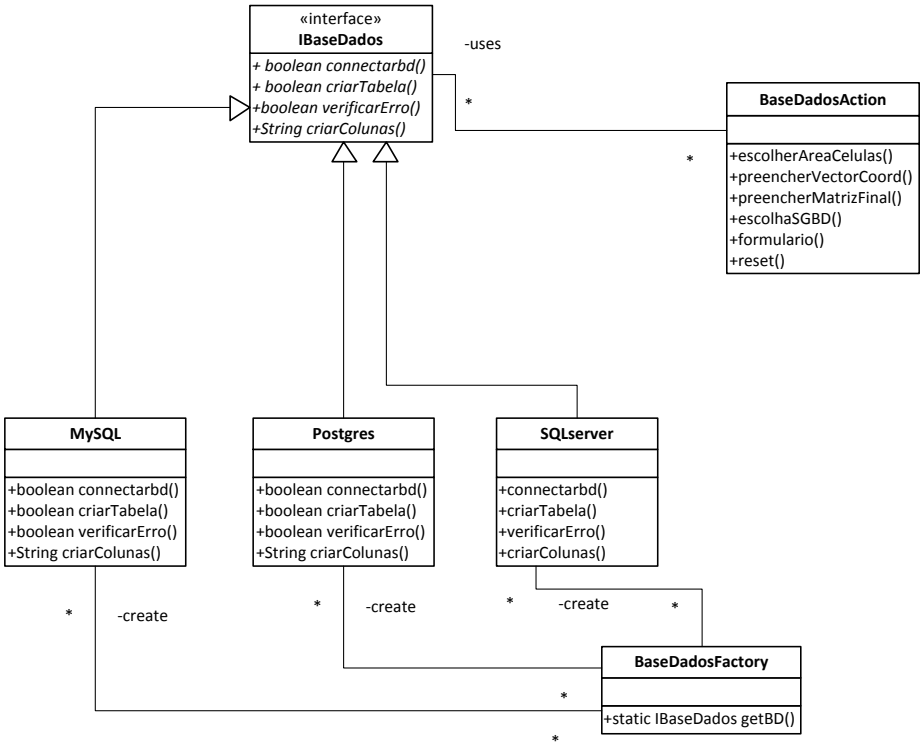


Diagrama de Classes



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

Só o caso de exportação está a funcionar, visto que a importação virá para a iteração 2.

3.4 Planeamento

- Efectuar casos de uso, diagrama de sequências e diagrama de classes;
- JDBC driver e a sua implementação com os seguintes driveres : MySQL, Postgres e SQLserver;
- Adaptação a MySQL;
- Adaptação a Postgres;
- Adaptação a SQLserver;
- Implementação do código responsável pela indicação da área da folha que deve exportar;
- Implementação do código responsável pela indicação dos dados necessários para a gravação.

4 Concepção

4.1 Especificação

Os mesmos da análise.

4.2 Aspectos técnicos da solução

Utilizei o padrão Factory, para a classe BaseDadosAction (Cliente) não ficar responsável por saber criar novos objectos do tipo MySQL, Postgres e SQLserver. Assim só basta ao cliente, como classe, que o utilizador introduza a informação sobre em que tipo de SGBD pretende gravar. A classe BaseDadosFactory é responsável pela criação e pelo retorno do objecto para a classe principal. Deste modo, no futuro, será mais fácil a adaptação de outros tipos de SGBD.

4.3 Testes

4.3.1 Testes unitários

```
/**
     * A1 é uma coordenada correcta
     */
    @Test
    public void testValidaString_A1() {
        System.out.println("validaString");
        String a = "A1";
        BaseDadosAction instance = new BaseDadosAction(null);
        boolean expResult = true;
        boolean result = instance.validaString(a);
        assertEquals(expResult, result);

    }
```

```
 /**
 * A127 é uma coordenada correcta
 */
@Test
public void testValidaString_A127() {
    System.out.println("validaString");
    String a = "A127";
    BaseDadosAction instance = new BaseDadosAction(null);
    boolean expResult = true;
    boolean result = instance.validaString(a);
    assertEquals(expResult, result);

}

/**
 * AB é uma coordenada errada
 */
@Test
public void testValidaString_AB() {
    System.out.println("validaString");
    String a = "AB";
    BaseDadosAction instance = new BaseDadosAction(null);
    boolean expResult = false;
    boolean result = instance.validaString(a);
    assertEquals(expResult, result);

}
```

```
    /**
     * A1-A2 é uma coordenada errada
     */
    @Test
    public void testValidaString_A1_A2() {
        System.out.println("validaString");
        String a = "A1-A2";
        BaseDadosAction instance = new BaseDadosAction(null);
        boolean expResult =false;
        boolean result = instance.validaString(a);
        assertEquals(expResult, result);

    }

/**
     * Se a matriz é algo assim A B
     *                          A B C
     * deverá dar erro, visto que a coluna 3 não tem qualquer valor, gerando assim uma mensagem de
erro
      */
    @Test
    public void testVerificarErro() throws SQLException, ClassNotFoundException {
        System.out.println("verificarErro");
        matriz[0][0]="A";
        matriz[0][1]="B";
        matriz[0][2]="";
        matriz[1][0]="A";
        matriz[1][1]="B";
        matriz[1][2]="C";
        IBaseDados instance = BaseDadosFactory.getBD("mysql");
        boolean expResult = false;
        boolean result = instance.verificarErro(matriz);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.

    }
```

```
/**
     * Se a matriz é algo assim A B C
     *                          A B C
     * deverá deixar continuar, visto que as colunas estão todas preenchidas com valores.
      */
    @Test
    public void testVerificarErro_correcto() 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");
        boolean expResult = true;
        boolean result = instance.verificarErro(matriz);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.

    }
```

4.3.2 Casos de teste

Semana 1
(28 de Maio a 3 de Junho)

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

Objectivo	Pretende-se implementar uma nova linguagem de expressões que inicie pelo caractere "#" e que se baseie na língua portuguesa – nova gramática.
Pré-requisitos	
Dados de teste	*Fórmula inserida na barra principal, uma string do tipo "#fórmula(argumento1;argumento2)*
Passos	*1. inserir formula* *2. verificar se o valor é o correcto*
Notas e Questões	

Resultados

#Execução	Dados	Resultados	Passou?	Observações
#1	*#Media(4;6;8)*	6	Sim	
#2	#Soma(4;8)	12	Sim	
#3	#Se(A1>A2,Maior,Menor)	Maior	Sim	

Comentário [Alexandre1]: Exemplo de descrição de um caso de teste. A substituir pelos casos de teste do projecto.

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, no entanto, penso que no futuro, caso haja tempo, poderá melhorar-se um bocado mais a parte gráfica, de maneira a não ser tão repetitiva a forma como o utilizador tem que introduzir e seleccionar informação.

7 Bibliografia

- JDBC Connection Example,JDBC Connection Example,Connecting to a MySQL Database in Java

- Quadro 1 - Exemplo de conexão JDBC com o banco de dados for Persistência em banco de dados relacionais usando linguagens orientada a objetos

- Connect To SQL Server 2008 From NetBeans IDE On A Mac

- Lesson: JDBC Basics (The Java™ Tutorials > JDBC(TM) Database Access)