Relatório Técnico do Projecto

CleanSheets

Grupo 2

Elemento
João Dias 1100604

1 Introdução

1.1 Apresentação

O problema inicial é relativo à sincronização entre uma tabela de uma Base de Dados e um grupo de células da folha CleanSheets.

1.2 Requisitos

Ser possível ao utilizador indicar qual a área da folha que pretende sincronizar com uma tabela já existente.

1.3 Objectivos

O objectivo do problema está centrado no facto de ser possível a actualização automática entre uma tabela base dados e uma área da CleanSheets.

1.4 Dificuldades

A principal dificuldade foi conceber o algoritmo para a actualização, tendo em conta o requisito de ser em X em X segundos. Mais uma vez, existem algumas diferenças significativas entre os vários SGBD, o que torna o trabalho um pouco mais complexo. Tive que adoptar a seguinte ordem no código, por razões de performance (*devido à complexidade do algoritmo*) , além do mais não existe qualquer diferença significativa face ao que é pedido no enunciado.

```
apaga bd -> apaga células
insere bd -> insere células
update com prioridade para a bd
apaga células -> apaga bd
insere células -> insere bd
```

1.5 Estrutura do Relatório Técnico

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

2 Enquadramento

2.1 Descrição dos Requisitos

Escolher o SGBD, nome da tabela e indicar os dados de conexão para a sincronização

O utilizador terá que introduzir toda a informação necessária para que a conexão seja efectuada com êxito.

Requisito importante : - O utilizador só consegue sincronizar a partir de uma tabela já existente. Caso não exista, basta gravar a tabela na base de dados e carregar a partir dela.

Escolher a área da folha que pretende sincronizar

O utilizador terá que indicar a área da folha que pretende sincronizar com a tabela.

Requisito importante : - o utilizador terá que indicar uma área que seja compatível com a tabela, quer na sua largura, como na sua "altura". Isto é, a área a sincronizar terá que ser igual, inicialmente, à área da tabela.

2.2 Enquadramento do Projecto

Todos os requisitos estão elaborados, sendo que o projecto está concluído.

2.3 Funcionalidades

A principal funcionalidade é a sincronização a partir de uma tabela de uma Base de Dados e uma área da folha.

3 Análise

3.1 Requisitos Funcionais

Casos de uso

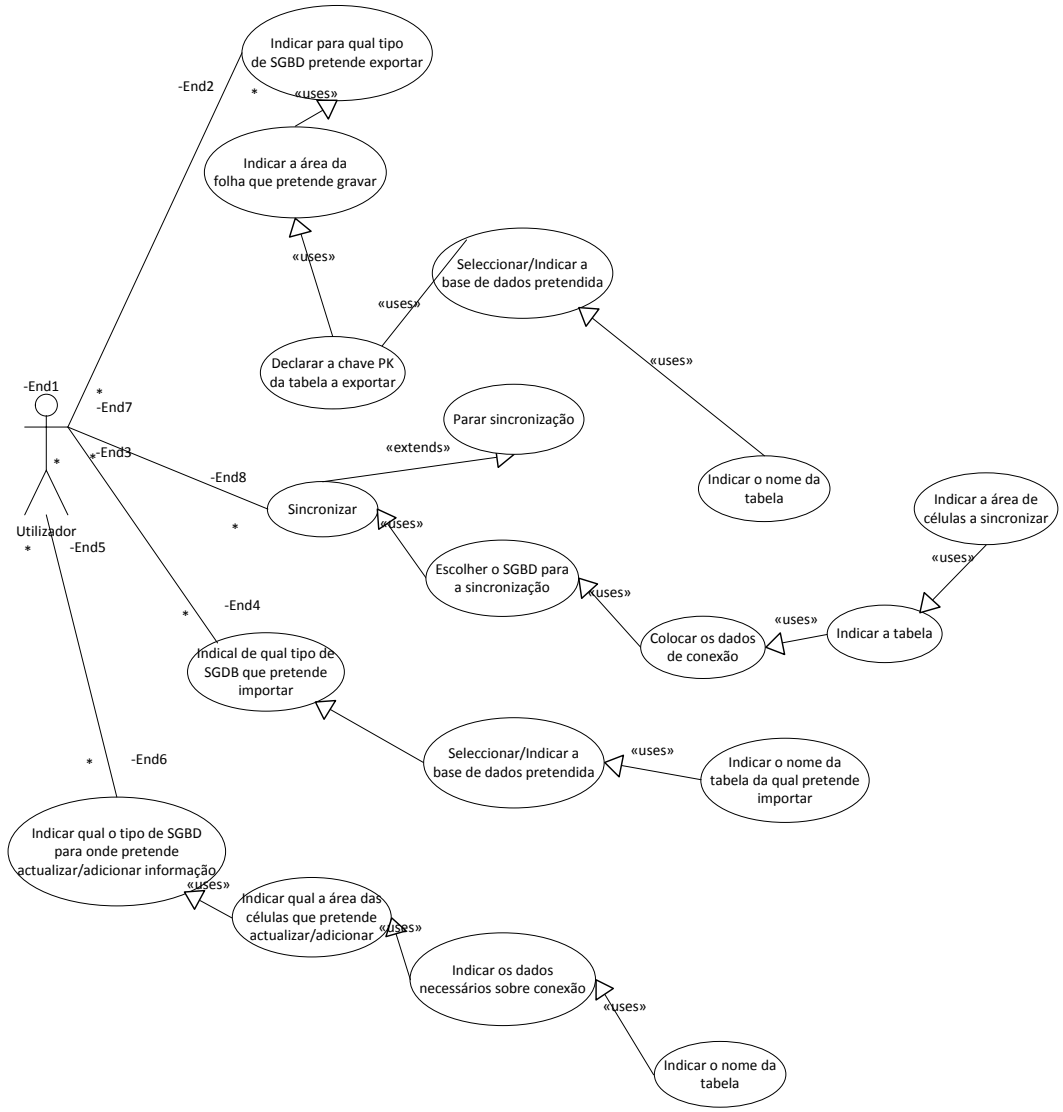
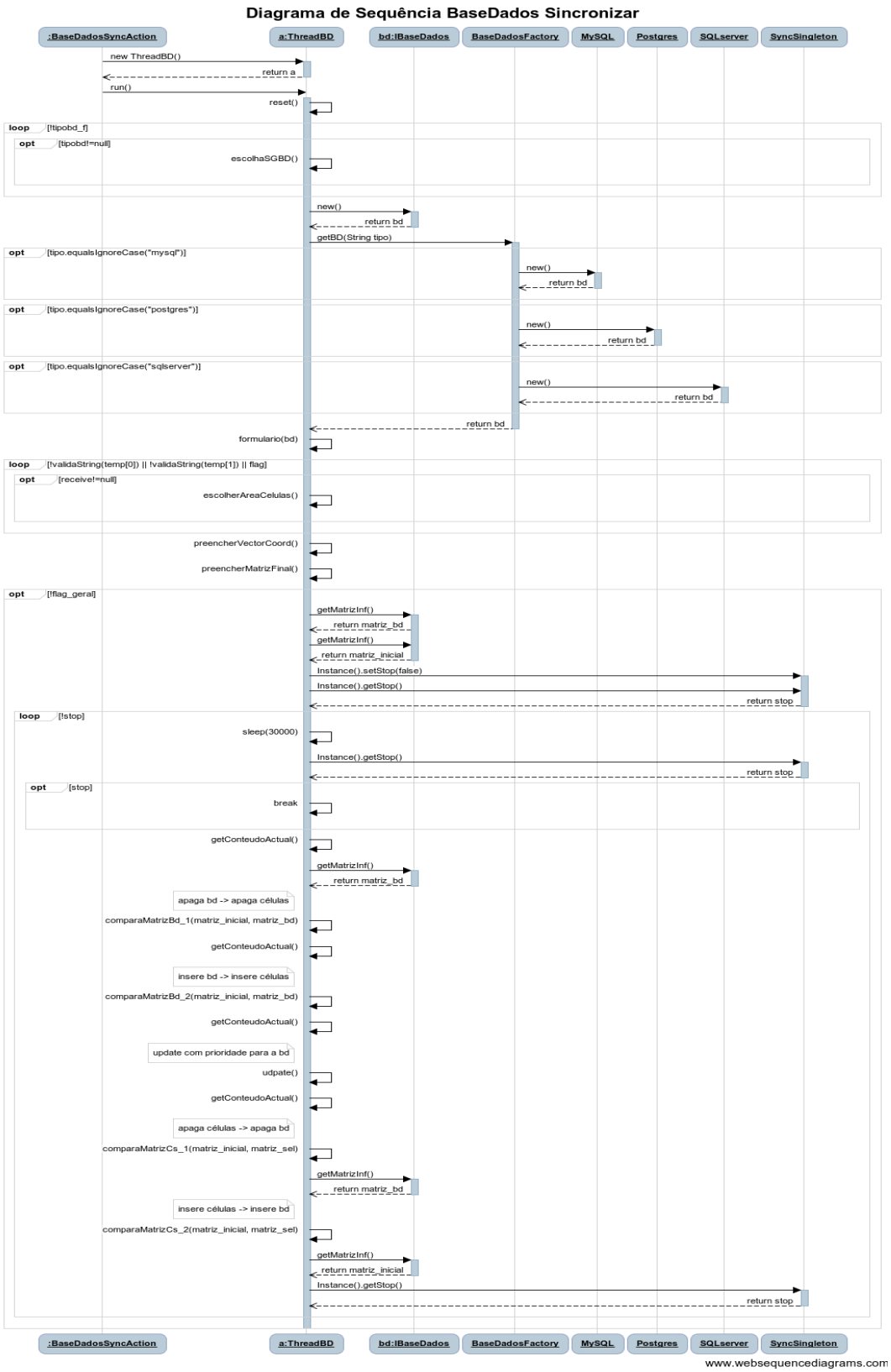


Diagrama de sequências



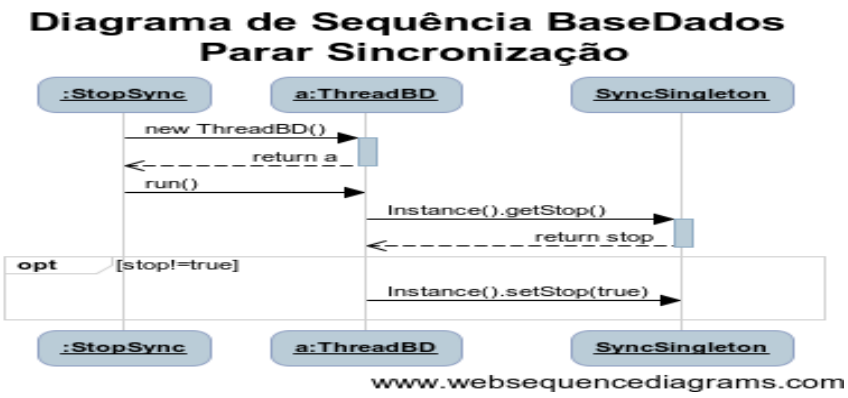
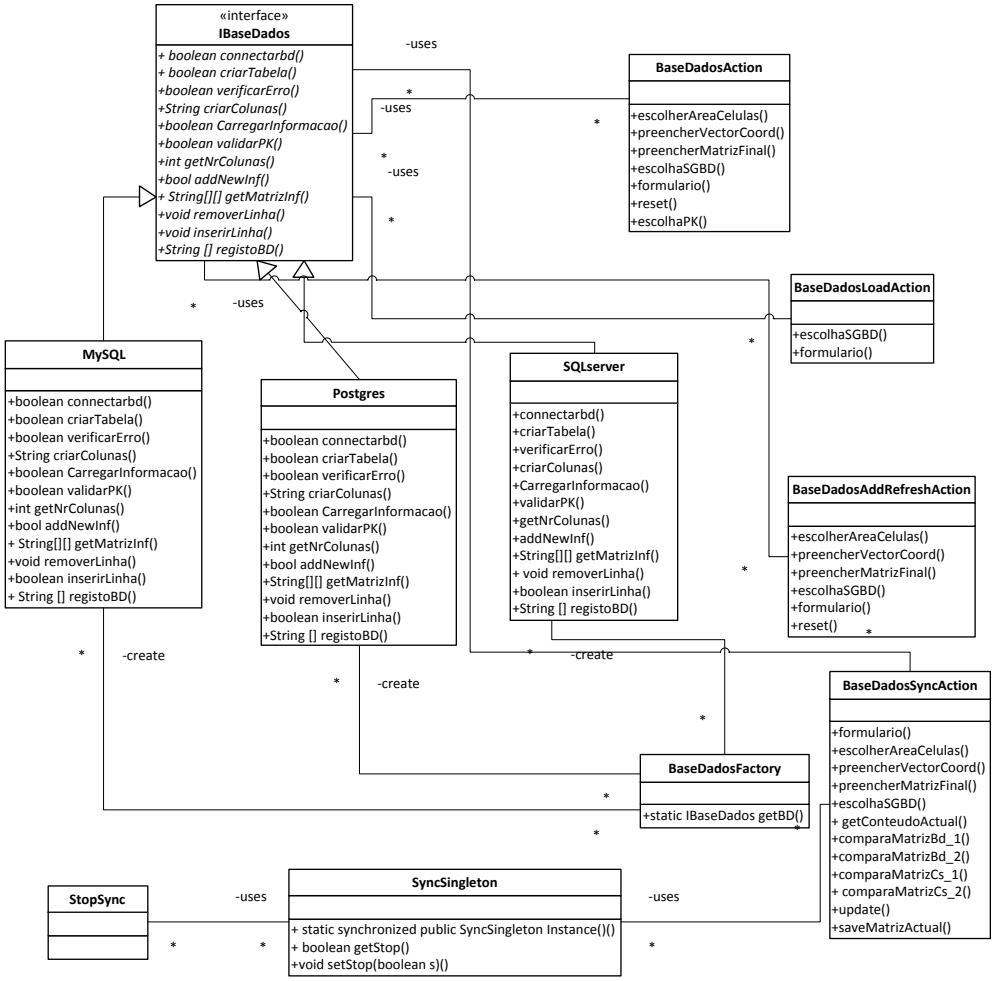


Diagrama de Classes



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

Todos os requisitos estão devidamente implementados.

3.4 Planeamento

- Criar novo item de opção para sincronização;
- Incorporar código referente ao grupo de células;
- Incorporar código referente à tabela;
- Incorporar código referente à junção da tabela com o grupo de células;
- Correcção de alguns bugs.

4 Concepção

4.1 Especificação

Os mesmos da análise.

4.2 Aspectos técnicos da solução

Nesta iteração, decidi usar o padrão **Singleton**, de maneira a ser possível ao utilizador parar a sincronização quando o desejar. Visto que a sincronização está embutida num ciclo while, caso não existisse esta opção, o código nunca iria sair do ciclo, sendo que o utilizador teria que forçar o encerramento da aplicação, para parar com a sua sincronização. Não faz parte dos requisitos, mas achei por bem colocar.

4.3 Testes

Alguns testes que poderiam ser feitos a estes métodos, caso os mesmos não dependessem de dados de conexão, que nem sempre são verdadeiros.

public String[][] getMatrizInf(String nome_tab,String nome_bd,String tipobd, String end,String porta, String user, String pass)

public boolean inserirLinha(String[] linha, String nome_tab,String nome_bd,String tipobd, String end,String porta, String user, String pass)

public String [] registoBD(String vec[],String nome_tab,String nome_bd,String tipobd, String end,String porta, String user, String pass)

4.3.1 Testes unitários

```
/**
     * Inicialmente o valor de STOP terá que ser verdadeiro.
     */
    @Test
    public void testGetStop() {
        System.out.println("getStop");
        SyncSingleton instance = SyncSingleton.Instance();
        boolean expResult = true;
        boolean result = instance.getStop();
        assertEquals(expResult, result);
    }
```

4.3.2 Casos de teste

Semana 3
(1 de Junho a 15 de Junho)

Nome do caso de teste: *Base Dados*
Casos de uso relacionados:

Comentário [Alexandre1]:

Objectivo	Pretende-se que um grupo de células possa ficar ligada de forma permanente a uma tabela de uma base de dados e periodicamente hajam actualizações (nos dois sentidos). Para tal deve-se usar uma thread que executa regularmente (por exemplo a cada 30 segundos) a actualização entre as células e a tabela (sincronização). O cleansheets, sempre que faz uma sincronização, guarda o estado das células após a sincronização numa estrutura de dados interna. Quanto correr a nova sincronização essa informação deve ser usada para determinar se : • um registo foi apagado na base de dados (deve ser removido nas células); • um registo foi inserido na base de dados (deve ser inserido nas células); • um registo foi removido nas células (deve ser removida na base de dados); • um registo foi inserido nas células (deve ser inserido na base de dados); • um registo existe nos dois "lados" mas com valores diferentes (devem as células e as colunas ficarem com os valores alterados, com prioridade para as alterações efectuadas na base de dados)
Pré-requisitos	
Dados de teste	*Nome da base de dados.* *Endereço e porta onde a bd está instalada* *Username e pass para aceder*
Passos	1. *Escolher o SGBD dos 3 disponíveis* 2. *Indicar o nome da tabela para a sincronização* 3. *Escolher a área a sincronizar*
Notas e Questões	*O utilizador deverá ter os servidores do SGBD para testar no seu computador e criar os respectivos utilizadores + pass de maneira a conseguir aceder.*

Resultados

#Execução	Dados	Resultados	Passou?	Observações
#1	Indicar o nome da tabela a sincronizar, mais a área pretendida. Adiciona uma linha na BD.	Linha adicionada nas células	Sim	
#2	Indicar o nome da tabela a sincronizar, mais a área pretendida. Remove uma linha na BD.	Linha removida nas células.	Sim	
#3	Indicar o nome da tabela a sincronizar, mais a área pretendida. Adiciona uma linha nas células	Linha adicionada na base de dados	Sim	
#4	Indicar o nome da tabela a sincronizar, mais a área pretendida. Remove uma linha nas células	Linha removida na base de dados	Sim	
#5	Indicar o nome da tabela a sincronizar, mais a área pretendida. Edita uma linha nas células com a mesma PK de uma já existente na bd	É feito um update ao valor da linha que se encontra nas células.Dando prioridade à informação da bd.	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.

Convém referir que na importação, a mesma só é feita caso exista um parâmetro id_t_c nas tabelas a exportar, visto que é feito um **selec * from X order by id_t_c**, de maneira a garantir a **equidade** dos dados exportados perante os dados importados.

6 Conclusão

Penso que o que foi pedido para estas três iterações foi concluído com sucesso, visto que está tudo a funcionar devidamente e como é pedido no enunciado. Foi um bom trabalho de se realizar, visto o que o mesmo foi feita de forma iterativa, o que torna tudo mais simples para quem programa e para quem pede o programa.

7 Bibliografia

- http://stackoverflow.com/questions/1673841/examples-of-gof-design-patterns