



# Administração e Exploração de Base de Dados

## Monitor de Base de Dados

### Grupo 6:

Filipe Cunha (A83099)  
Luís Braga (A82088)

João Nunes (A82300)  
Luís Martins (A82298)

Braga, Portugal  
19 de Dezembro de 2019

# Índice

<b>1</b>	<b>Introdução</b>	<b>4</b>
<b>2</b>	<b>Modelo Concetual</b>	<b>5</b>
<b>3</b>	<b>Modelo Lógico</b>	<b>7</b>
<b>4</b>	<b>Arquitetura</b>	<b>8</b>
<b>5</b>	<b>Base de dados</b>	<b>9</b>
5.1	User . . . . .	9
5.1.1	Tablespaces e datafiles . . . . .	9
5.2	Modelo físico . . . . .	10
5.2.1	Triggers e sequences . . . . .	10
<b>6</b>	<b>Servidor update</b>	<b>12</b>
6.1	Connection . . . . .	12
6.2	Extração da informação . . . . .	13
6.2.1	Update da tabela base de dados . . . . .	13
6.2.2	Update da tabela tablespaces . . . . .	14
6.2.3	Update da tabela data files . . . . .	15
6.2.4	Update da tabela users . . . . .	16
6.2.5	Update da tabela roles . . . . .	16
6.2.6	Update da tabela privileges . . . . .	16
6.2.7	Update da tabela CPU . . . . .	16
6.2.8	Update da tabela Memória . . . . .	16
6.2.9	Update da tabela Sessions . . . . .	17
6.2.10	Update da tabela user roles . . . . .	17
6.2.11	Update da tabela user roles . . . . .	17
<b>7</b>	<b>Servidor da Página Web</b>	<b>19</b>
7.1	REST . . . . .	19
7.2	Interface . . . . .	21
7.2.1	Página inicial . . . . .	21
7.2.2	Informações sobre a base de dados . . . . .	21
7.2.3	Listagem de utilizadores . . . . .	22
7.2.4	Sessões activas . . . . .	25
7.2.5	Tablespaces . . . . .	26
7.2.6	Uso de CPU . . . . .	28
7.2.7	Uso de memória . . . . .	28



# Índice de figuras

2.1	Modelo concetual do sistema. . . . .	5
3.1	Modelo lógico do sistema. . . . .	7
4.1	Esboço da arquitetura implementada. . . . .	8
6.1	Servidor Update. . . . .	12
7.1	. . . . .	19
7.2	Exemplo de consulta com formato <i>JSON</i> na tabela <i>Sessions</i> . . . . .	20
7.3	Página principal. . . . .	21
7.4	Informação sobre a bases de dados. . . . .	21
7.5	Informação sobre os utilizadores. . . . .	22
7.6	Informação sobre os roles. . . . .	23
7.7	Informação sobre os privilégios. . . . .	24
7.8	Informação sobre as sessões activas. . . . .	25
7.9	Informação sobre as sessões activas. . . . .	26
7.10	Informação sobre os datafiles. . . . .	26
7.11	Informação histórica sobre os datafiles. . . . .	27
7.12	Informação sobre a taxa de utilização do CPU. . . . .	28
7.13	Informação sobre a memória. . . . .	28

# 1 Introdução

No decorrer deste trabalho visou-se a criação de um monitor simples mas eficaz no que toca à avaliação e monitorização de uma base de dados *Oracle*.

Para tal foi necessário no início projetar o projeto, criando para tal o modelo concetual no qual foram identificadas as principais entidades do sistema, e os seus atributos, bem como o relacionamento e cardinalidade entre as entidades.

De seguida, e com base no modelo concetual, foi elaborado o modelo lógico onde foram atribuídos a cada um dos atributos das tabelas anteriores o seu domínio de valores.

De modo a conseguir extrair os dados das views da *Pluggable DB* foi criado um utilizador com este intuito com o objetivo de depois guardar estes dados numa *tablespace* de armazenamento.

Com os dados pretendidos presentes na *tablespace* de armazenamento, procedeu-se à utilização de uma API em *REST* com a finalidade de depois utilizar os ficheiros *JSON* devolvidos para apresentar os dados numa interface gráfica.

## 2 Modelo Concetual

Numa primeira instância foi elaborado o seguinte modelo concetual, que visa a identificar as principais entidades do sistema bem como os seus atributos, as relações e as cardinalidades associadas a estas.

Este foi o ponto de partida para o desenvolvimento da aplicação, uma vez que permite identificar os principais pontos do sistema numa forma clara e sistemática. O modelo concetual serve também como base para a criação do modelo lógico numa fase mais posterior.

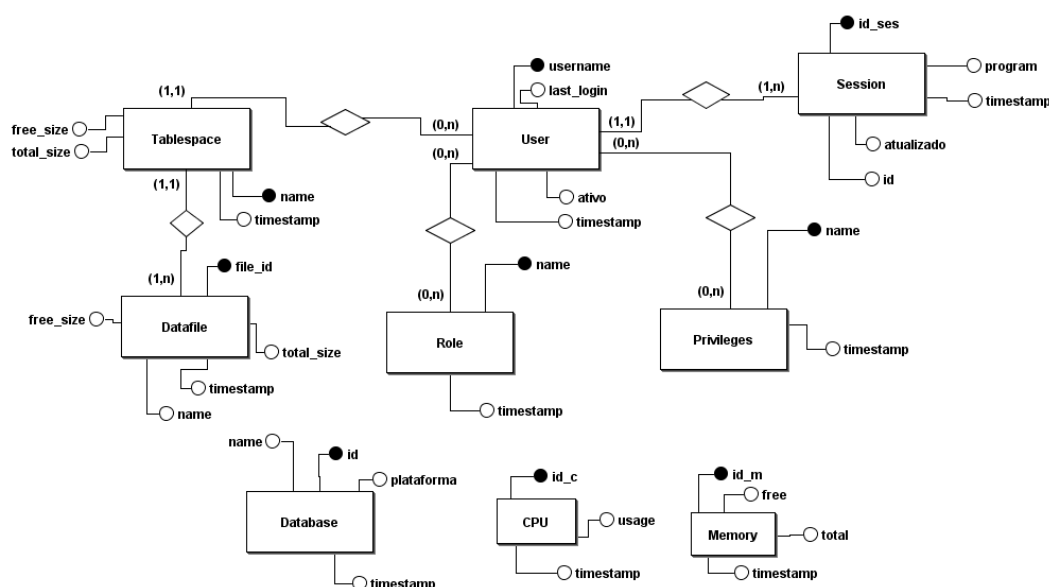


Figura 2.1: Modelo concetual do sistema.

Portanto, no modelo anterior é possível verificar que a entidade principal da aplicação será a *Database* que irá ser identificada por um identificador único a *id* da base de dados, a plataforma *host* da BD, bem como os *timestamps* da utilização da base de dados para efeitos estatísticos. Os utilizadores também irão possuir um identificador único sendo este o seu *username*, uma *flag* que indica se encontra-se ativo ou desativo, a data do seu último *login*, bem como a data de atualização do user. Relacionados com os users possui-se as entidades relativas aos *Roles* e *Privileges* onde se apresentam os nomes dos roles e privilégios associados a cada user sendo estas as chaves primárias das respetivas entidades, e um *timestamp* referente às alterações. Cada user também deverá possuir sessões associadas a ele próprio que indicam o programa que utilizou para se conectar à BD bem como uma *flag* que indica se a sessão está atualizada ou não e também uma *timestamp* com a data da atualização.

Ligado a cada user tem-se o tablespace *default* que será identificado através do seu nome e irá possuir o espaço total alocado para esse tablespace bem como o espaço livre e mais uma vez a data de atualização. Associado aos tablespaces possui-se os datafiles, o identificador único dos datafiles será um ID, o nome do datafile e mais uma vez o espaço total ocupado e o espaço livre do datafile e a data de atualização.

Para além destas entidades, existe também as entidades relacionadas com os recursos computacionais sendo estas o CPU e a memória. Relativamente à memória esta é identificada por um identificador, a memória global disponível bem como a memória total livre e por fim e tal como nos outros casos a data de atualização. O CPU tal como a memória irá possuir um identificador, a percentagem de utilização e a data de atualização.

### 3 Modelo Lógico

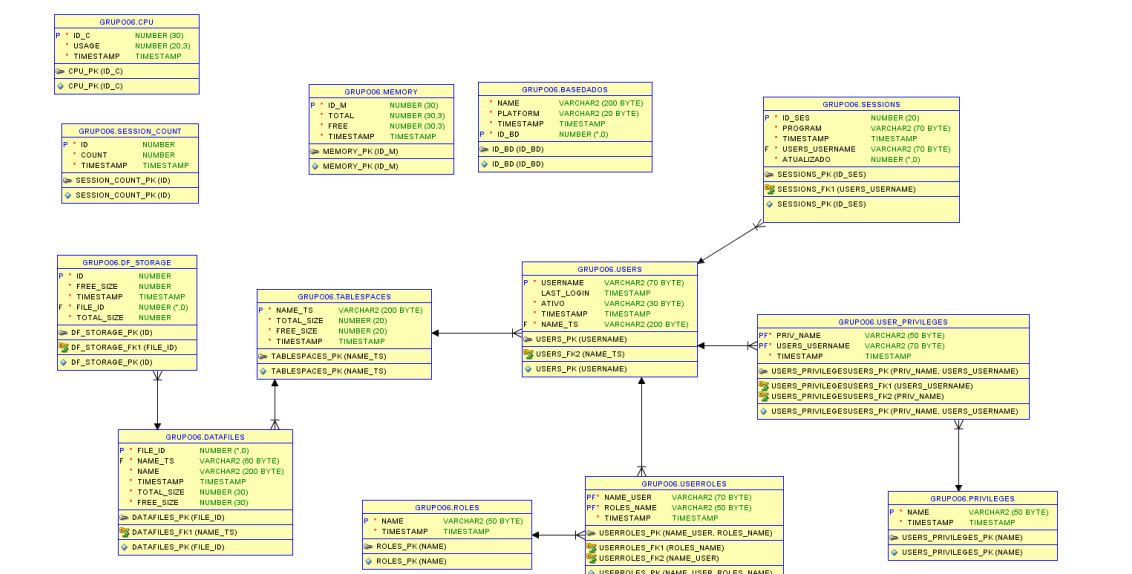


Figura 3.1: Modelo lógico do sistema.

O modelo lógico supracitado é uma consequência do modelo conceitual concebido anteriormente, e como tal foi apenas necessário aplicar o processo de normalização e a escolha dos tipos de dados para cada atributo.

A partir deste modelo é possível efetuar *foward engineering* de modo a gerar o código responsável pela implementação do modelo físico.



## 4 Arquitetura

Relativamente à arquitectura desenvolvida pelo grupo, o grupo resolveu estruturar o projeto da seguinte maneira.

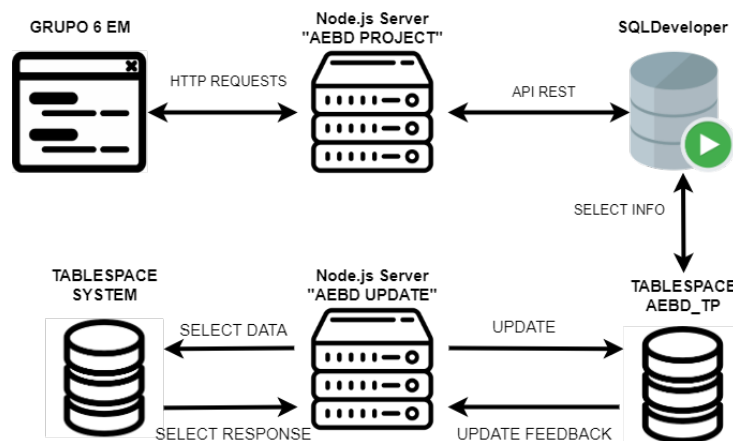


Figura 4.1: Esboço da arquitetura implementada.

A interface irá comunicar com o servidor *AEBD\_PROJECT* através de HTTP REQUESTS, onde o utilizador a cada interação que tem com a interface irá corresponder a um pedido pelos dados necessários. O servidor de modo a corresponder aos pedidos do utilizador, e utilizando a API REST comunicará com o *SQLDeveloper* de modo a conseguir obter os dados necessários. O *SQLDeveloper* de seguida através utilizará consultas SQL de modo a aceder aos dados presentes na tablespace *AEBD\_TP* que contém todos os dados relativos à implementação.

Os dados no tablespace *AEBD\_TP* têm de ser atualizados uma vez que não convém disponibilizar informação desatualizada aos utilizadores, como tal existe um outro servidor responsável por esta atualização dos dados sendo ele o *AEBD\_UPDATE*, este irá também consultar através de comandos SQL os dados do sistema, e irá atualizar o tablespace *AEBD\_TP* com o resultado destas consultas.

## 5 Base de dados

### 5.1 User

De forma a obter a informação das *views* do sistema, tipicamente o utilizador *sys* terá estes direitos, ou seja, terá o direito de recolher informação não só acerca das *pluggable DB* mas também acerca da *root DB* no que toca, por exemplo, a mostrar todos os utilizadores criados.

Foi também criado então mais um utilizador responsável por gerir o modelo físico da base de dados ou seja o tablespace *AEBD\_TP*. Como tal foi associado a este utilizador o *default* tablespace do *AEBD\_TP* e o temporary tablespace de *AEBD\_TEMP*. Foi também atribuído a este o *role* de *DBA* bem como as permissões necessárias para a criação de sessões (para conectar à BD), permissões para criar tabelas (para implementar o modelo físico) bem como as permissões *default* de recursos e de conexão.

```
01 | CREATE
02 | USER grupo06
03 | IDENTIFIED BY oracle
04 | DEFAULT TABLESPACE AEBD_TP
05 | TEMPORARY TABLESPACE TP_TEMP
06 | QUOTA 250M on AEBD_TP;
07 |
08 | GRANT CONNECT TO grupo06;
09 | GRANT RESOURCE TO grupo06;
10 | GRANT CREATE SESSION TO grupo06 ;
11 | GRANT CREATE TABLE TO grupo06 ;
```

#### 5.1.1 Tablespaces e datafiles

Foram criadas uma tablespace permanente e outra temporária denominadas de *AEBD\_TP* e *AEBD\_TEMP* tendo sido associadas ao user *grupo06* criado anteriormente. A primeira tablespace irá possuir o datafile "*AEBD\_TP\_01.dbf*" e a segunda o "*TP\_TEMPORARY\_01.dbf*". O objetivo da criação destes tablespaces é permitir o armazenamento dos dados que o grupo considerou necessários para permitir a monitorização de uma base de dados *Oracle*.

```
01 | --TABLESPACE AEBD_TP
02 | CREATE TABLESPACE AEBD_TP
03 |     DATAFILE
04 |         '\u01\app\oracle\oradata\orcl12\orcl\AEBD_TP_01.DBF' SIZE 500M;
05 |
06 | --TABLESPACE TP_TEMP
07 | CREATE TEMPORARY TABLESPACE TP_TEMP
08 |     TEMPFILE
```

```
09 |      '\u01\app\oracle\oradata\orcl12\orcl\TP_TEMPORARY_01.DBF' SIZE 250M
    |      ;
```

## 5.2 Modelo físico

Relativamente ao código responsável pela criação do modelo físico, este surgiu através de um processo de *foward engineering* partindo do modelo lógico apresentado anteriormente.

Como tal, foi possível estruturar a criação das tabelas através de um ficheiro gerado automaticamente com uma sintaxe *DLL* (linguagem de definição de dados), sendo apresentado de seguida um exemplo do código necessário para a criação de uma tabela, neste caso a tabela *base de dados*.

```
01 | CREATE TABLE grupo06.basedados (
02 |     name          VARCHAR2(200 BYTE) NOT NULL,
03 |     platform      VARCHAR2(20 BYTE) NOT NULL,
04 |     timestamp     TIMESTAMP NOT NULL,
05 |     id_bd         NUMBER(*,0) NOT NULL
06 | )
07 | PCTFREE 10 PCTUSED 40 TABLESPACE users LOGGING
08 |     STORAGE ( INITIAL 65536 NEXT 1048576 PCTINCREASE 0 MINEXTENTS 1
    |     MAXEXTENTS 2147483645 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
    | )
09 | NO INMEMORY;
10 |
11 | CREATE UNIQUE INDEX hr.id_bd ON
12 |     grupo06.basedados ( id_bd ASC )
13 |     TABLESPACE users PCTFREE 10
14 |     STORAGE ( INITIAL 65536 NEXT 1048576 PCTINCREASE 0 MINEXTENTS 1
    |     MAXEXTENTS 2147483645 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT
    | )
15 |     LOGGING;
16 |
17 | ALTER TABLE grupo06.basedados
18 |     ADD CONSTRAINT id_bd PRIMARY KEY ( id_bd )
19 |     USING INDEX hr.id_bd;
```

### 5.2.1 Triggers e sequences

No toca à apresentação dos elementos mais dinâmicos do monitor, ou seja, a memória e o CPU bem como o número de sessões, foi necessário garantir que o *ID* destas tabelas tivesse de ter sido preenchido automaticamente.

Desta maneira, as três primeira sequências têm o intuito de guardar o id da sequência ao id do elemento que está a ser inserido naquele dado momento.

A última sequência possui o intuito de antes de cada inserção na tabela relativa às sessões para elaborar um *refresh* e colocar cada uma das sessões lá presentes antes da inserção como desatualizadas.

```
01 | -- id auto incremental do CPU
02 | create sequence CPU_Sq start with 1;
03 | CREATE OR REPLACE TRIGGER CPU_Tr
04 |     BEFORE INSERT ON CPU
```

```

05 |         FOR EACH ROW
06 | BEGIN
07 |     :new.ID_C := CPU_Sq.nextval;
08 | END;
09 |
10 | -- id auto incremental da memoria
11 | create sequence Memory_Sq start with 1;
12 | CREATE OR REPLACE TRIGGER Memory_Tr
13 |     BEFORE INSERT ON MEMORY
14 |     FOR EACH ROW
15 | BEGIN
16 |     :new.ID_M := Memory_Sq.nextval;
17 | END;
18 |
19 | -- id auto incremental do numero de sessoes
20 | create sequence SESSION_C_Sq start with 1;
21 | CREATE OR REPLACE TRIGGER SESSION_C_Tr
22 |     BEFORE INSERT ON SESSION_COUNT
23 |     FOR EACH ROW
24 | BEGIN
25 |     :new.ID := SESSION_C_Sq.nextval;
26 | END;
27 |
28 | -- id auto incremental da tabela de historico dos datafiles
29 | create sequence DF_Storage_Sq start with 1;
30 | CREATE OR REPLACE TRIGGER DF_STORAGE_Tr
31 |     BEFORE INSERT ON DF_STORAGE
32 |     FOR EACH ROW
33 | BEGIN
34 |     :new.ID := DF_Storage_Sq.nextval;
35 | END;

```

Desta maneira, é também possível associar a cada ID um *timestamp* da inserção no *tablespace* garantindo assim também a presença de um histórico com o intuito de depois apresentar ao utilizador do monitor.

## 6 Servidor update

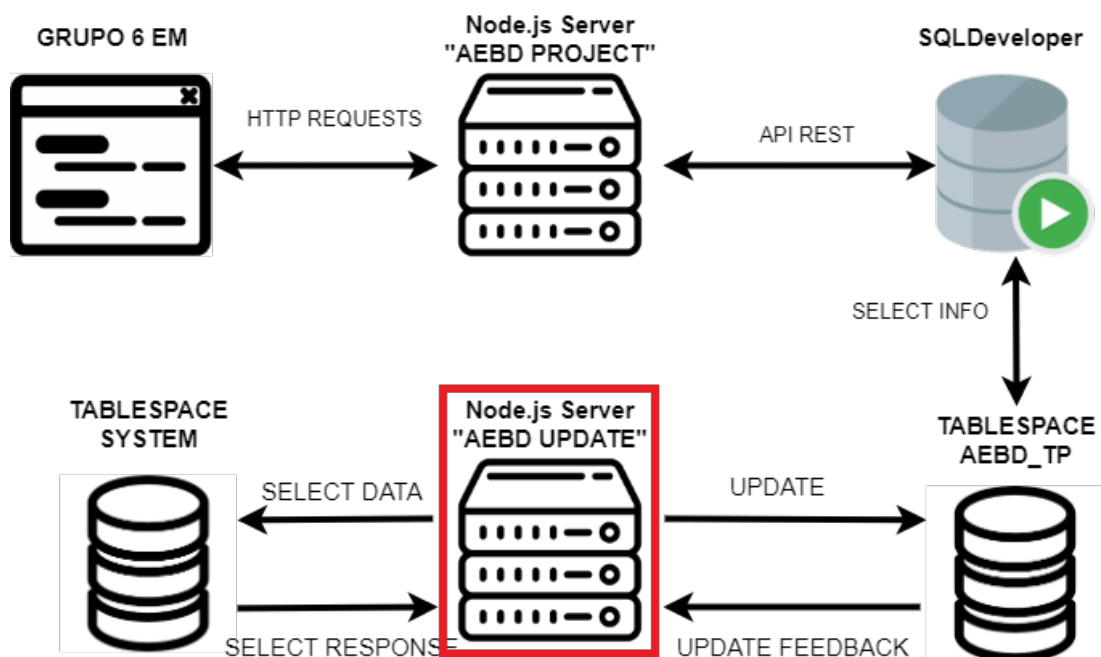


Figura 6.1: Servidor Update.

Tendo definido todo o suporte relativamente à parte da base de dados, é então possível apresentar o servidor update, cujo objetivo é recolher a informação relativa à base de dados e de seguida atualizar o *tablespace* relativo à base de dados onde se encontra agregada a informação recolhida por este.

### 6.1 Connection

Como tal foi preciso primeiro criar três tipos de conexões diferentes recorrendo para tal ao *node-oracledb* como a ferramenta para permitir a conexão ao sistema de base de dados. Este permite com que seja possível executar interrogações sobre a base de dados bem como obter o resultado dessas mesmas queries.

As duas primeiras conexão são relativas à conexão à *Container Database* e a *Pluggable Database* pelo user *system*. Abre-se também mais uma conexão à PDB desta vez pelo utilizador criado pelo grupo, ou seja, o user *grupo06*.

```
const oracledb = require('oracledb');
```

```

// Coneccao ao ORCL12c
const dbaCon12c= {
  user      : "system",
  password  : "oracle",
  connectionString : "//localhost/orcl12c"
};

const dbaCon= {
  user      : "system",
  password  : "oracle",
  connectionString : "//localhost/orcl"
};

const groupCon= {
  user      : "grupo06",
  password  : "oracle",
  connectionString : "//localhost/orcl"
};

```

## 6.2 Extração da informação

Tendo definido o método de conexão, é agora possível executar as interrogações necessárias de modo a poder atualizar o tablespace *AEBD\_TP*. Como tal, para cada uma das tabelas do tablespace *AEBD\_TP* irá existir uma function com esse intuito.

### 6.2.1 Update da tabela base de dados

Relativamente à tabela base de dados, é necessário extrair informação relativa ao nome e a plataforma de utilização das base de dados existentes. Como tal, desenvolveu-se uma interrogação SQL simples com este intuito

01 | "SELECT NAME, PLATFORM\_NAME, DBID FROM V\$DATABASE"

Esta query extrai a informação da view *V\$DATABASE* disponível na pluggable DB, com os campos pretendidos. Esta query é de seguida integrada na sintaxe *JavaScript* dentro de uma *function* onde para cada linha resultante da interrogação anterior irá atualizar os dados já existentes na tabela *BASEDEDADOS* no *AEBD\_TP* ou caso a informação extraída não se encontre nessa tabela irá ter que a adicionar.

```

function updateTableBD() {
  dbaConnection.execute("SELECT NAME, PLATFORM_NAME, DBID FROM V$DATABASE")
    .then(dados => {
      // update da informacao na tabela Base de dados
      dados.rows.forEach((dado) =>{
        var update = "UPDATE BASEDEDADOS SET NAME = :0, PLATFORM = :1,
          timestamp = CURRENT_TIMESTAMP WHERE id_bd = :2 "

```

```

groupConnection.execute(update , dado , { autoCommit : true })
.then(d => {
    // Nova informacao e adicionada (insert)
    if(d.rowsAffected == 0){
        var insert = "INSERT INTO BASEDADOS (NAME, PLATFORM,
            timestamp , id_bd) VALUES (:0 , :1 , CURRENT_TIMESTAMP, :2)
        "
        groupConnection.execute(insert , dado , { autoCommit: true
        })
        .catch(erro => console.log( 'ERRO NO INSERT DA TABLE DA BASE
            DE DADOS: ' + erro ))
    }
})
.catch(erro => console.log( "ERRO NO UPDATE DA TABLE DA BD : " +
    erro))
})
})
.catch(erro => console.log( "ERRO NO SELECT DA DATABASE: " + erro ))
}

```

## 6.2.2 Update da tabela tablespaces

Relativamente à tabela tablespaces esta necessitará de guardar dados relativos ao seu tamanho total bem como o tamanho livre para além do nome do tablespace.

Como tal, esta interrogação demonstra-se como sendo mais complexa, uma vez que irá ter de incorporar *Selects* dentro de outro *Selects* de modo a extrair a informação necessária.

```

01 | (SELECT tablespace_name, '+' SUM (bytes) / (1024 * 1024)
02 |     "FREE (MB)" ' + 'FROM dba_free_space '+' '
03 |     GROUP BY tablespace_name)

```

Esta primeira "sub-query" tem como objetivo retornar o espaço total livre de cada *tablespace* sendo os resultados agrupados pelo nome da tablespace ao qual o espaço livre é referente. Como tal, acede-se à view *dba\_free\_space* referente à pluggable DB.

```

01 | (SELECT tablespace_name, SUM(bytes) / (1024 * 1024) "SIZE (MB)", COUNT(*) '
02 |     + "File Count", SUM(maxbytes) / (1024 *1024) "MAX_EXT" '
03 |     + 'FROM dba_data_files '
04 |     + 'GROUP BY tablespace_name)

```

Esta segunda interrogação possui o intuito de retornar o espaço total associado a cada *tablespace*, como tal terá que aceder à view *dba\_data\_files* também referente à pluggable DB, e no final agrupar o espaço total consoante o nome do *tablespace*.

```

01 | '(SELECT tablespace_name '
02 |     + 'FROM dba_tablespaces) ts '
03 |     + 'WHERE fr.tablespace_name = df.tablespace_name (+) '
04 |     + 'AND fr.tablespace_name = ts.tablespace_name (+) '

```

O objetivo desta última interrogação é agregar os dados retirados anteriormente pelo nome da *tablespace* de modo produzir os dados corretos referentes a cada *tablespace*.

A juntar a estas três *sub-queries* existe a query principal cujo objetivo é selecionar os campos necessários para a tabela *tablespace*, sendo que de seguida retira os dados através da junção das três queries anteriores numa maneira sequencial.

```

01 | SELECT ts.tablespace_name, TRUNC("SIZE(MB)", 2) "Size",
02 | TRUNC(fr."FREE(MB)", 2) "Free" 'FROM '
03 | (...)

```

De seguida a função é construída de uma maneira análoga à função apresentada anteriormente. A função irá iterar sobre os dados resultantes da query anterior e irá tentar fazer *UPDATE* dos dados. Caso tal não seja possível ou seja, *ROWS AFFECTED* == 0, então irá inserir a informação na tabela uma vez que esta não se encontra lá.

### 6.2.3 Update da tabela data files

Na tabela *DATAFILES* é necessário guardar a informação relativa ao nome dos datafiles, o seu tamanho total ocupado em disco bem como o tamanho livre. Como tal, também representa uma interrogação complicada pelo que é necessário consultar views diferentes do pluggable DB.

```

01 | (SELECT file_id, " +
02 |     " Sum(Decode(bytes,NULL,0,bytes)) used_bytes " +
03 |     " FROM dba_extents " +
04 |     " GROUP by file_id) "

```

Nesta primeira *sub-query*, é necessário retirar o tamanho ocupado pelo datafile, como tal acede-se à view *dba\_extents*, agrupando este tamanho ocupado através do identificador do datafile.

```

01 | (SELECT Max(bytes) free_bytes, file_id " +
02 |     " FROM dba_free_space " +
03 |     " GROUP BY file_id) f " +
04 |     " WHERE e.file_id (+) = df.file_id " +
05 |     " AND df.file_id = f.file_id (+) "

```

Nesta segunda interrogação, acede-se à view *dba\_free\_space* novamente de modo a retirar a informação relativa ao tamanho total ainda disponível dos datafiles, sendo, obviamente, necessário fazer a referencialidade das chaves de modo a garantir a integridade dos valores retirados com o datafile.

```

01 | SELECT df.FILE_ID, " +
02 |     "Substr(df.tablespace_name,1,40) NAME_TB, " +
03 |     "Substr(df.file_name,1,20) NAME_DF, " +
04 |     "Round(df.bytes/1024/1024,0) FILE_SIZE, " +
05 |     "decode(f.free_bytes,NULL,0,Round(f.free_bytes/1024/1024,0)) FREE_SIZE
06 |     " FROM DBA_DATA_FILES DF

```

O select principal utilizado na função de atualização da informação na tabela *data files* consiste na utilização da view *DBA\_DATA\_FILES* de modo a retirar os ids dos datafiles, e possui também os mecanismos para formatar os dados obtidos através da execução aninhada das outras queries apresentadas anteriormente.

Novamente o mecanismo presente nesta função é igual aos anteriores, é feito o update da tabela datafiles através do resultado da interrogação apresentada anteriormente. Caso esta informação não se encontre na tabela os dados são inseridos.



### 6.2.4 Update da tabela users

Na tabela *users*, a informação a guardar é relativa à data do seu último login, se encontra-se ativo ou desativo e o seu username. Como tal esta informação é facilmente obtida recorrendo à view *dba\_users* presentes na pluggable DB.

```
01 | Select username, last_login, account_status, DEFAULT_TABLESPACE From  
    dba_users
```

O mecanismo de inserção ou update da informação da tabela ocorre exatamente da mesma maneira que os anteriores.

### 6.2.5 Update da tabela roles

Relativamente à tabela *roles* estes são relativos ao utilizador, portanto é apenas necessário guardar o nome do *role*. É necessário portanto aceder à view *dba\_roles* oferecido pela pluggable DB de modo a retirar este campo.

```
01 | SELECT ROLE FROM dba_roles
```

Relativamente à função propriamente dita, o seu mecanismo é análogo aos anteriores.

### 6.2.6 Update da tabela privileges

A tabela *privileges* é algo semelhante à tabela *roles* na medida em que guarda também apenas o privilégios. Portanto é apenas extraído o nome do privilégio proveniente na view *dba\_sys\_privs*, sendo feito um *DISTINCT* de modo a não aparecer privilégios repetidos.

```
01 | select distinct PRIVILEGE from dba_sys_privs
```

Na função a inserção ou update é novamente feita de um mecanismo análogo aos apresentados anteriormente.

### 6.2.7 Update da tabela CPU

Relativamente aos recursos computacionais gastos pela base de dados oracle, estes são consultados tendo por base a view *SYS.V\_\$SYSMETRIC* disponibilizada pela pluggable DB, sendo então possível retirar daí a utilização do CPU.

```
01 | select value from SYS.V_$SYSMETRIC where METRIC_NAME IN ('Database CPU Time  
    Ratio') and INTSIZE_CSEC =(select max(INTSIZE_CSEC) from SYS.  
    V_$SYSMETRIC)
```

Mais uma vez, e neste caso, é apenas possível fazer a inserção da informação. O id é gerado através do trigger também apresentado no capítulo anterior.

### 6.2.8 Update da tabela Memória

De modo a contabilizar a memória disponível e a total que é guarda na tabela *memory* foi necessário aceder a duas views diferentes, a primeira *SYS.V\_\$SGASTAT* é acedida de modo a retornar a memória disponível tendo sido feita a conversão deste campo para *bytes*, a segunda

view consultada é a *v\$sga* e esta é consultada de modo a retornar a memória total, tendo sido feita também a conversão para *bytes*. É de salvaguardar que ambas estas views são disponibilizadas pela pluggable DB também.

```
01 | SELECT round(SUM(bytes/1024/1024)) FROM \"SYS\".\"V_$SGASTAT\" Where Name  
    | Like '%free memory%' union SELECT sum(value)/1024/1024 FROM v$sga
```

A função atua de maneira semelhante à anterior, e como no caso do CPU é apenas possível ocorrer a inserção da informação, ou seja, não é feito updates. O trigger responsável por gerar o id através de um sequence é acionado sempre que ocorre uma inserção nesta tabela.

### 6.2.9 Update da tabela Sessions

Na tabela sessions é necessário guardar a informação referente a uma sessão na base de dados oracle, ou seja, é guardado o programa que está a aceder à base de dados. A partir da view *v\$session* é também possível retirar o id da session bem como o utilizador ao qual a session se refere.

```
01 | select distinct SID, PROGRAM, USERNAME " +  
02 |     "from v$session " +  
03 |     "WHERE USERNAME IS NOT NULL and PROGRAM != 'APEX Listener'"
```

Foi também necessário fazer uma verificação adicional no que toca às sessions uma vez que existia bastante sessions referentes a processos a correr em background, ou seja, não possuía nenhum utilizador em específico e através do uso do *REST* também se verificou um número elevado de sessions cujo o programa é *APEX Listener* e como tal o grupo optou por filtrar estas ocorrências. A function em si também apenas permite a inserção utilizando como tal uma query *INSERT INTO*, dos dados retirados através da query anterior.

### 6.2.10 Update da tabela user roles

É necessário atualizar ou inserir informação também referente às tabelas auxiliares de relacionamento, onde nestas são guardadas as chaves referentes ao user e ao seu role. Como tal, acede-se à view *dba\_sys\_privs* de modo a que seja possível selecionar os privilégios e os respetivos utilizadores que possuem esses dados privilégios, selecionando desta maneira as duas chaves da relação.

```
01 | select distinct SID, PROGRAM, USERNAME " +  
02 |     "from v$session " +  
03 |     "WHERE USERNAME IS NOT NULL and PROGRAM != 'APEX Listener'"
```

Na function em si, de seguida esse conjunto de dados resultantes da execução da query anterior é iterado, onde para cada linha é feito o update na tabela anteriormente referida. Caso o update não seja possível (não existe) então é necessário inserir essa informação.

### 6.2.11 Update da tabela user roles

A tabela user roles é também uma consequência do relacionamento da tabela users com a tabela roles, como tal também será necessário adicionar ou atualizar informação nesta tabela. Esta tabela guarda portanto as chaves referentes à tabela users e da tabela roles, onde é necessário

aceder à view *dba\_role\_privs* de modo a conseguir obter os roles e os utilizadores a quem os roles foram atribuídos.

```
01 | select GRANTEE, GRANTED_ROLE from dba_role_privs
```

Mais uma vez na function, é feito a atualização dos dados consoante as duas chaves retiradas anteriormente, fruto da query anterior, onde caso não seja possível atualizar uma vez que a informação não se encontra lá, é feita tal como no caso anterior uma inserção dos novos dados.

É de salvaguardar também que em cada ocorrência de uma inserção ou atualização dos dados em todos os casos apresentados anteriormente, e tal como está especificado no modelo lógico, é guardado um *timestamp* referente a estas operações sobre o *tablespace*.

A função responsável por chamar todas as funções apresentadas anteriormente é a *updateBD* sendo que esta irá chamar por ordem cada uma das funções anteriores, tendo de salvar a ordem pela qual as chaves existem nas tabelas, uma vez que, por exemplo, não pode ser chamada primeiro a função *updateUsersRoles* sem ter primeiro chamado a função *updateRoles* uma vez que existe a referencialidade de chaves entre estas duas tabelas.

```
function updateBD() {
    updateTableBD()
    updateCPU()
    updateMemory()
    updateTableSpaces()
        .then(d => {
            updateDataFiles()
            updateUsers()
                .then(de => {
                    updateSessions()
                    updateRoles()
                        .then(d => {
                            updateUsersRoles()
                        })
                    updatePrivileges()
                        .then(d => {
                            //concluiu update com sucesso!
                            updateUsersPrivileges()
                        })
                })
            })
        .catch(error => console.log(error))
    })
    .catch(erro => console.log(erro))
}
```

## 7 Servidor da Página Web

Com a finalidade de se conseguir visualizar os dados para melhor realizar a gestão das bases de dados criou-se uma interface web que traduz esta funcionalidade.

Para tal também se implementou um servidor que trata de servir páginas e requisitar os dados através da API REST definida no SQLDeveloper. Esse servidor é o que se encontra rodeado na seguinte figura.

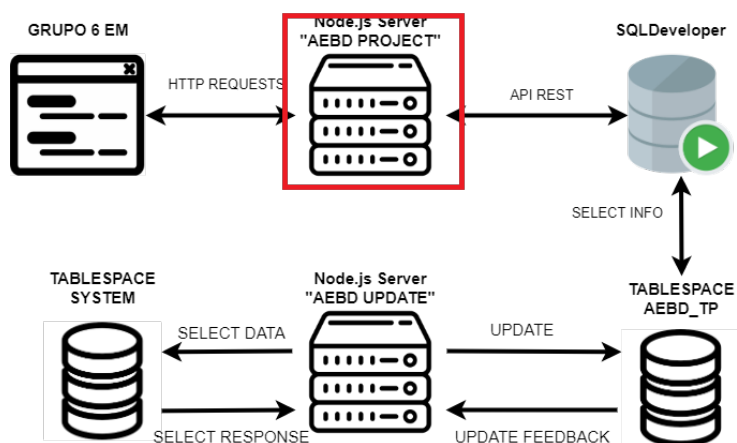


Figura 7.1: .

### 7.1 REST

Relativamente ao desenvolvimento da API *REST* este serviço foi disponibilizado pelo *ORACLE*. O *Oracle Rest Data Services (ORDS)* disponibiliza maneiras fáceis e modernas de desenvolver interfaces *REST* permitindo comunicação *HTTP* através de métodos *GET* permitindo que ocorra transações com a base de dados retornando o resultado da transação no formato *JSON*.

Ao nível da implementação nas *interfaces* relativas a cada uma das tabelas do tablespace *AEBD\_TP* mostra-se de seguida de uma maneira geral o procedimento para fazer os *HTTP GETS* utilizando a API *REST* de cada uma das respectivas tabelas. De seguida, os dados são enviadas para o respetivo ficheiro *template pug* de modo a poder apresentar graficamente a informação guardada na base de dados.

Os ficheiros *JSON* resultantes de cada consulta à base de dados utilizando a API *REST* irá possuir o seguinte formato:

```

{
  "items": [
    {
      "id": 24558,
      "id_ses": 13,
      "program": "oracle@localhost.localdomain (OFSD)",
      "timestamp": "2019-12-19T14:45:57.076Z",
      "users_username": "SYS",
      "atualizado": 1,
      "links": [
        {
          "rel": "self",
          "href": "http://localhost:8080/ords/grupo06/sessions/24558"
        }
      ]
    },
    {
      "id": 24557,
      "id_ses": 46,
      "program": "SQL Developer",
      "timestamp": "2019-12-19T14:45:57.073Z",
      "users_username": "SYS",
      "atualizado": 1,
      "links": [
        {
          "rel": "self",
          "href": "http://localhost:8080/ords/grupo06/sessions/24557"
        }
      ]
    },
    {
      "id": 24556,
      "id_ses": 82,
      "program": "SQL Developer",
      "timestamp": "2019-12-19T14:45:57.07Z",
      "users_username": "GRUPO06",
      "atualizado": 1,
      "links": [
        {
          "rel": "self",
          "href": "http://localhost:8080/ords/grupo06/sessions/24556"
        }
      ]
    }
  ]
}

```

Figura 7.2: Exemplo de consulta com formato *JSON* na tabela Sessions.

## 7.2 Interface

### 7.2.1 Página inicial



Figura 7.3: Página principal.

### 7.2.2 Informações sobre a base de dados


DataBase	
	
<b>Id</b>	776972821
<b>Name</b>	ORCL12C
<b>Platform</b>	Linux x86 64-bit
<b>TimeStamp</b>	2019-12-19T14:45:56.49Z

Figura 7.4: Informação sobre a bases de dados.

## 7.2.3 Listagem de utilizadores




















<b>MENU</b> HomePage Database Users Sessions TableSpaces CPU Memory	Users					
						
	Name	Last Login	Account Status	Default TableSpace	Privileges	Roles
	X\$NULL	Undefined	EXPIRED & LOCKED	SYSTEM		
	SYSTEM	2019-12-20T16:13:22Z	OPEN	SYSTEM		
	OUTLN	Undefined	EXPIRED & LOCKED	SYSTEM		
	LBACSYS	Undefined	EXPIRED & LOCKED	SYSTEM		
	SYS	Undefined	OPEN	SYSTEM		
	APPQOSSYS	Undefined	EXPIRED & LOCKED	SYSAUX		
	DBSNMP	Undefined	EXPIRED & LOCKED	SYSAUX		
	GGSYS	Undefined	EXPIRED & LOCKED	SYSAUX		
	DBSFUSER	Undefined	EXPIRED & LOCKED	SYSAUX		

Figura 7.5: Informação sobre os utilizadores.


SYS Roles	
	
Role	
ADM_PARALLEL_EXECUTE_TASK	
APEX_ADMINISTRATOR_READ_ROLE	
APEX_ADMINISTRATOR_ROLE	
APEX_GRANTS_FOR_NEW_USERS_ROLE	
APPLICATION_TRACE_VIEWER	
AQ_ADMINISTRATOR_ROLE	
AQ_USER_ROLE	
AUDIT_ADMIN	
AUDIT_VIEWER	
AUTHENTICATEDUSER	
CAPTURE_ADMIN	
CDB_DBA	
CONNECT	
CSW_USR_ROLE	
CTXAPP	
DATAPATCH_ROLE	
DATAPUMP_EXP_FULL_DATABASE	

Figura 7.6: Informação sobre os roles.




SYS Privileges	
	
Privilege	
MANAGE TABLESPACE	
BACKUP ANY TABLE	
SELECT ANY SEQUENCE	
DROP ANY TRIGGER	
CREATE PUBLIC DATABASE LINK	
DROP ANY PROCEDURE	
CREATE ANY SYNONYM	
ALTER ANY MATERIALIZED VIEW	
DROP ANY SYNONYM	
UNDER ANY TYPE	
DROP PROFILE	
CREATE DIMENSION	
EXECUTE ANY OPERATOR	
DROP ANY RULE	
ADMINISTER RESOURCE MANAGER	
ADMINISTER DATABASE TRIGGER	
READ ANY FILE GROUP	

Figura 7.7: Informação sobre os privilégios.

## 7.2.4 Sessões activas

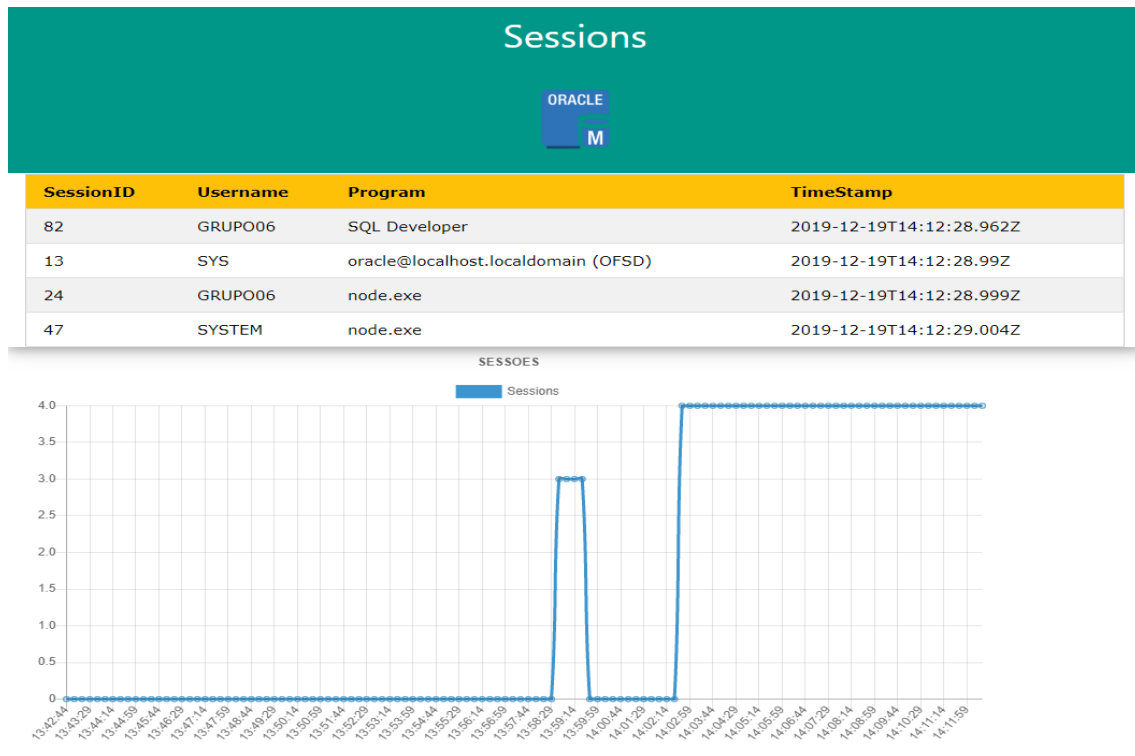


Figura 7.8: Informação sobre as sessões activas.

## 7.2.5 Tablespaces

TableSpaces				
ORACLE M				
Name	Total Space (MB)	Free Space (MB)	TimeStamp	
SYSAUX	1220	66	2019-12-19T14:10:33.116Z	
USERS	84	6	2019-12-19T14:10:33.12Z	
APEX_1941389856444596	8	1	2019-12-19T14:10:33.119Z	
SYSTEM	350	2	2019-12-19T14:10:33.125Z	
AEBD_TABLES	600	407	2019-12-19T14:10:33.122Z	
TP_TEMPJ1	250	249	2019-12-19T14:10:33.123Z	
AEBD_TPJ	500	499	2019-12-19T14:10:33.129Z	
UNDOTBS1	615	505	2019-12-19T14:10:33.117Z	

Figura 7.9: Informação sobre as sessões activas.

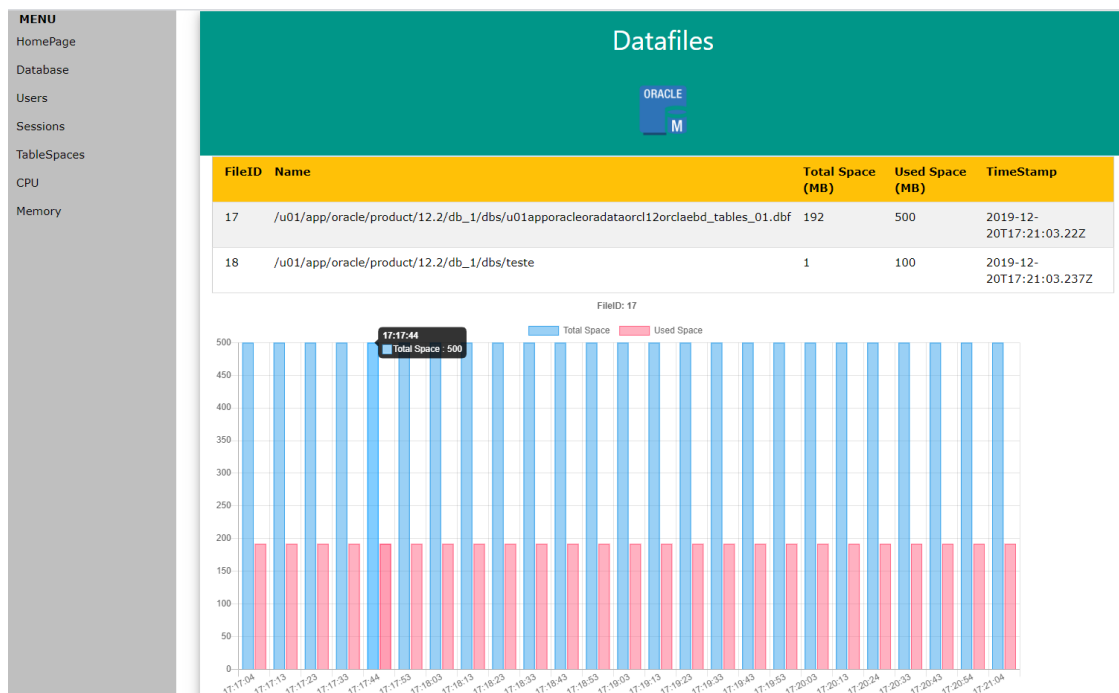


Figura 7.10: Informação sobre os datafiles.

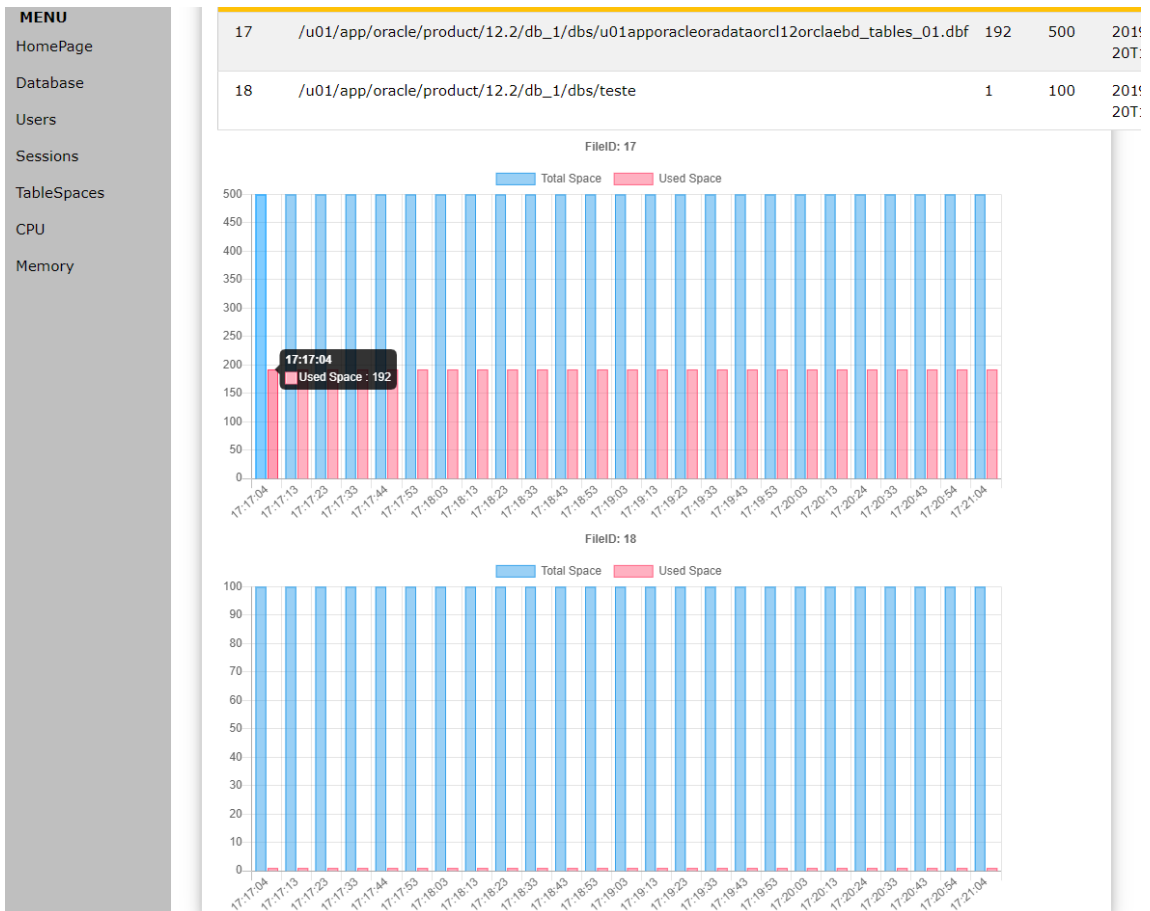


Figura 7.11: Informação histórica sobre os datafiles.

## 7.2.6 Uso de CPU

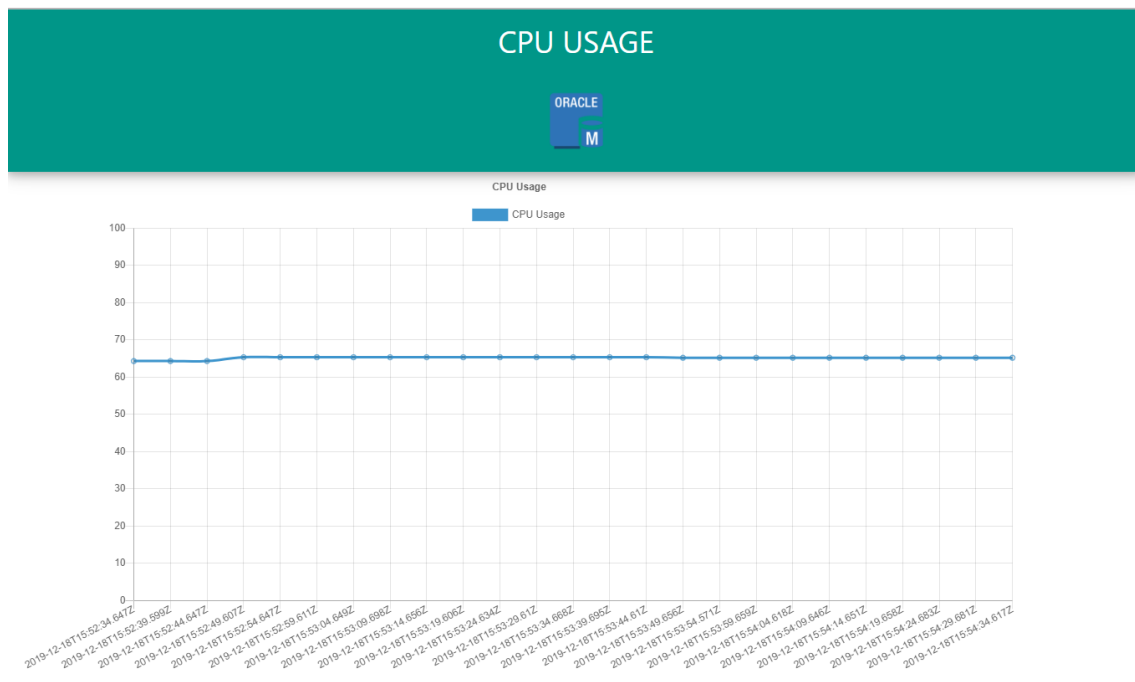
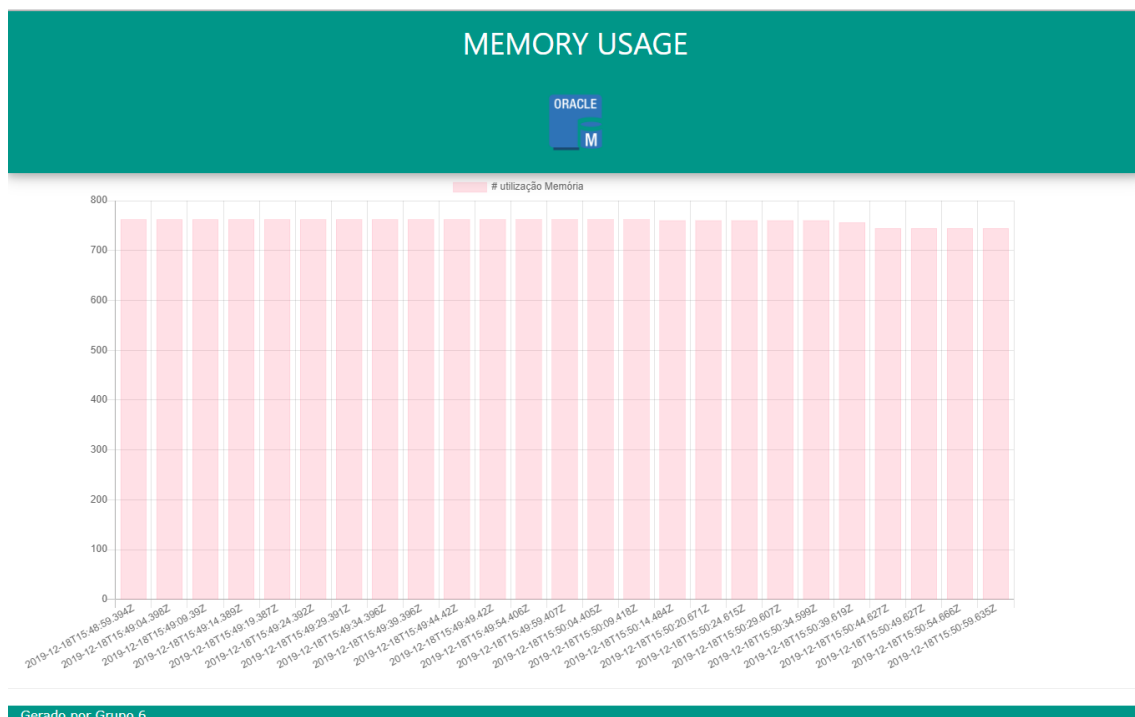


Figura 7.12: Informação sobre a taxa de utilização do CPU.

## 7.2.7 Uso de memória



Gerado por Grupo 6

Figura 7.13: Informação sobre a memória.

## 8 Conclusão

Em suma, a criação de um monitor de base de dados *Oracle* permitiu avaliar a *performance* da base de dados, e verificar dados sobre este de uma forma mais simples e intuitiva.

Como tal, durante o desenvolvimento deste trabalho surgiram alguns *stumbling blocks*, sendo que o primeiro e talvez o mais complicado foi a própria ligação ao *Oracle* através do node, uma vez que pouca documentação acerca deste processo. Outra complicação que surgiu durante o desenvolvimento do projeto foi a necessidade de mudar em vários estágios diferentes o modelo concetual e o modelo lógico, uma vez através do desenvolvimento incremental do monitor o grupo foi identificando alguns aspetos que estavam incorretos ou faltavam no modelo original.

Em suma, o projeto permitiu com que o grupo de trabalho consolida-se os conhecimentos relativos à base de dados *Oracle* bem como as estruturas por detrás deste.

# Bibliografia

- [1] Chart.js: Open source HTML5 Charts for your website,  
<https://www.chartjs.org/>
- [2] Burleson Consulting,  
<http://www.dba-oracle.com/>
- [3] StackOverflow,  
<https://pt.stackoverflow.com/>