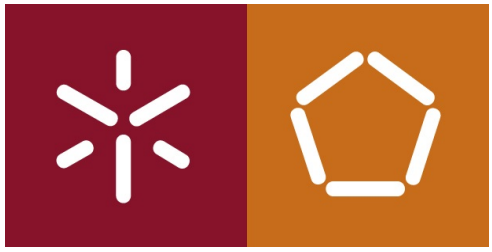

Universidade do Minho
Mestrado Integrado em Engenharia Informática
4ºano - 1º Semestre

Administração e Exploração de Bases de Dados

Trabalho Prático

Grupo 02

A83732 – Gonçalo Rodrigues Pinto
A80791 – João Diogo Mendes Teixeira Da Mota
PG42839 - José Gonçalo Macedo Costa
A84829 - José Nuno Martins da Costa

27 de Janeiro de 2021

Conteúdo

Lista de Figuras

1 Introdução

Hoje em dia, com a evolução da tecnologia e exigência de segurança de dados, os administradores de Bases de Dados têm vindo a ganhar uma enorme importância em qualquer sistema. Estes, entre outras funções, têm a responsabilidade de monitorizar constantemente o funcionamento de uma Base de Dados.

O presente trabalho enquadra-se na Unidade Curricular de Administração e Exploração de Bases de Dados, presente no perfil de Engenharia do Conhecimento e tem como objetivo criar competências aos alunos nessa mesma área. Destacam-se a identificação, descrição e definição das principais funções de um administrador de bases de dados, selecionando, para isso, as metodologias apropriadas para normalizar e modelar sistemas de dados e ainda desenhar modelos conceptuais. Tem também como meta consciencializar os alunos das propriedades essenciais dos sistemas de dados, em particular no que diz respeito à sua transportabilidade, disponibilidade permanente, integridade e segurança dos dados. Outro ponto essencial é o desenvolvimento da capacidade de avaliar e melhorar o desempenho dos sistemas de dados, utilizando para isso ferramentas de administração e exploração de Bases de Dados.

Neste trabalho, pretende-se que cada grupo construa um monitor de base de dados que apresente, de forma simples, os principais parâmetros de avaliação de performance de uma base de dados Oracle, permitindo aos alunos ter uma visão e uma aproximação do que são as tarefas de um administrador de Bases de Dados.

2 Descrição do problema

Com o objetivo final de se obter uma interface *web* capaz de apresentar alguns parâmetros interessantes, bem como a avaliação de desempenho de uma Base de Dados Oracle, foi definido previamente o percurso a percorrer para este fim. Este foi dividido em quatro fases principais:

Etapa 1: A criação de um agente em **Java** que, através das *views* de administração selecionadas, efetuou a recolha da informação necessária;

Etapa 2: A criação de uma nova *Pluggable Database* e respetivo *schema*, incluindo os seus *tablespaces* permanentes e temporários e *datafiles* associados, de forma a armazenar os dados recolhidos na etapa anterior. Nesta fase, também foi criado o modelo relacional de dados, de forma a organizar e armazenar os dados recolhidos;

Etapa 3: A implementação de uma API REST, com o intuito de se conectar à *Pluggable Database* criada na fase precedente e devolver os resultados pretendidos num formato à escolha (sendo este o formato **JSON**);

Etapa 4: A criação de uma interface *web*, utilizando **Node.js** com a *framework* **Vue.js**, de forma a serem apresentados de forma agradável e intuitiva os resultados obtidos.

3 Caracterização das Bases de Dados Oracle

De forma a construir este monitor, foi necessário efetuar um levantamento de características e estudo das mesmas, com o objetivo de perceber os principais parâmetros de avaliação de performance de uma base de dados *Oracle*. Apresenta-se, em seguida, o estudo realizado.

O SGBD Oracle tem vindo a ganhar muita popularidade nos últimos anos e, atualmente, ocupa a primeira posição no *ranking* elaborado pela DB-Engines [2]. Curiosamente, o Oracle está à frente do popular MySQL e também do SQL Server da Microsoft.

Neste monitor, foi utilizado o **Oracle Enterprise 12c** que possui uma arquitetura *multitenant*, isto é, possui três componentes centrais: exatamente uma *root*, que armazena metadados de suporte *Oracle* e utilizadores; uma *seed*, que se define como um *template* para a criação de *Pluggable Databases* (PDB); e zero ou mais PDB's (entidades criadas por utilizadores que contêm uma coleção de esquemas) (figura 1). Esta arquitetura possui várias vantagens como:

- **Redução de custos**, uma vez que a consolidação do *hardware* e da infraestrutura da base de dados, através de um único processo de *background*, e partilhando recursos, é possível reduzir custos de *hardware* e manutenção;
- **Facilidade de gestão e monitorização** porque o administrador da infraestrutura consegue administrar todo o ambiente, como aplicar *patches* de segurança, *upgrades* ou executar *backups*;
- São facilitadas estratégias de *backup* e *disaster recovery*;
- Garante **separação segura de competências**, dado que os utilizadores podem ser comuns, com acesso global a qualquer *container* que tenham acesso, ou locais, com acesso apenas à PDB com a qual foram criados;
- O administrador dessa PDB não tem acesso de administração às restantes PDB's;
- Permite efetuar **avaliação de desempenho**, no sentido em que é mais simples a avaliação de desempenho quando se monitoriza apenas um processo de *background*.

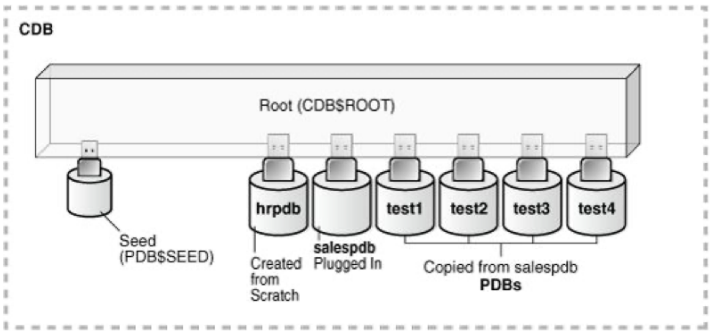

Figura 1: Exemplo da arquitetura do Oracle Enterprise 12c

Uma Base de Dados Oracle consiste em uma ou mais unidades lógicas de armazenamento denominadas *tablespaces* que, de forma unificada, permitem o armazenamento de toda a informação de uma Base de Dados. Estas unidades lógicas podem ser permanentes, *Permanent Tablespaces*, que contêm dados de objetos persistentes do *schema* (por exemplo, dados consolidados nas tabelas), ou podem ser temporárias, *Temporary Tablespaces*, contendo apenas informação de transição que persiste durante a sessão. Neste tipo de *tablespace*, não são armazenados objetos permanentes.

Cada *tablespace* consiste em um ou mais ficheiros físicos, denominados *datafiles*. Estes ficheiros são estruturas físicas presentes no sistema operativo do servidor, no qual a base de dados Oracle está a correr. Os objetos presentes nos *tablespaces* permanentes são armazenados em *datafiles*. Os *tablespaces* temporários são armazenados em *temp files*.

Assim, uma Base de Dados Oracle é uma coleção de *datafiles* que constituem os *tablespaces* da mesma.

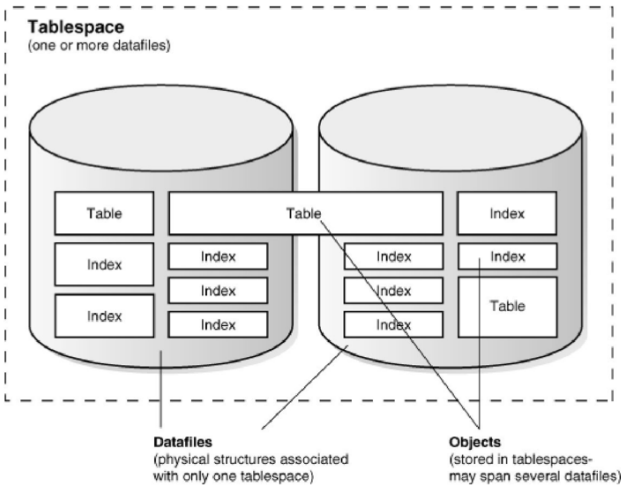

Figura 2: Arquitetura de uma Base de Dados Oracle

4 Vistas e Tabelas de Administração utilizadas

Uma das primeiras grandes etapas deste trabalho surgiu com a decisão das vistas de administração a aceder. Cada uma destas vistas contêm informação fundamental para a obtenção de bons indicadores de monitorização e a sua escolha baseou-se em alguns exemplos da *Enterprise Manager Console* analisados.

Com o auxílio da documentação oferecida pela Oracle[1], foi considerada a obtenção de toda a informação relevante para a monitorização através das colunas das seguintes vistas:

- **Roles**: Descreve todos os *Roles* existentes na Base de Dados.

  - *DBA_ROLES* - ROLE_ID , ROLE , AUTHENTICATION_TYPE;

  - *DBA_ROLE_PRIVS* - GRANTED_ROLE, GRANTEE;

- **Tablespaces**: Descreve todos os *tablespaces* acessíveis ao utilizador atual.

  - *DBA_TABLESPACES* - TABLESPACE_NAME, STATUS , ALLOCATION_TYPE , CONTENTS , SEGMENT_SPACE_MANAGEMENT;

  - *DBA_TABLESPACE_USAGE_METRICS* - TABLESPACE_SIZE , USED_SPACE , USED_PERCENT;

- **Datafiles**: Descreve todos os ficheiros existentes na Base de Dados.

  - *DBA_DATA_FILES* - FILE_NAME , FILE_ID , TABLESPACE_NAME , BYTES , BLOCKS , STATUS , AUTOEXTENSIBLE , MAXBYTES , MAXBLOCKS , ONLINE_STATUS;

- **Users**: Apresenta todos os utilizadores da Base de Dados bem como alguns dos seus dados.

  - *DBA_USERS* - USERNAME , ACCOUNT_STATUS , EXPIRY_DATE , DEFAULT_TABLESPACE , TEMPORARY_TABLESPACE , PROFILE , CREATED , COMMON;

- **Sessions**: Lista informações para cada sessão.

  - *V$SESSION* - SID , USERNAME , STATUS , SERVER , SCHEMANAME , OSUSER , MACHINE , PORT , TYPE , WAIT_TIME_MICRO , LOGON_TIME, SERIAL#;

  - *V$SESSTAT* - STATISTIC#, SID, VALUE;

  - *V$STATNAME* - STATISTIC#;

- **Memory** : Exibe estatísticas do uso da memória.

  - *V$PGASTAT* - NAME, VALUE;

  - *V$SGA* - NAME, VALUE;

5 Criação da PDB, respetivo schema com tablespaces e datafiles associados

Apresentado as características de uma Base de Dados da Oracle, conclui-se que é necessário um ou mais *tablespaces*, cada um destes composto por um ou mais *datafiles*, para armazenar a informação. Desta forma, foi criada uma nova PDB com o respetivo *schema*, *tablespaces* e *datafiles* associados. Para este efeito, executaram-se as seguintes operações:

1. Criaram-se as diretorias para acomodar os ficheiros da nova PDB, e alteraram-se as permissões:

```
[root@localhost ~]$
        cd /home/uminho/dockers/data/oracle/u02/app/oracle/oradata/ORCL
[root@localhost ~]$ mkdir DBMONITOR
[root@localhost ~]$ chown oracle:oinstall DBMONITOR
```

2. Usando a *bash* do *container*, ligou-se à base de dados de *root* (CDB), criando a PDB através da PDB Seed:

```
[root@localhost ~]$ sudo docker ps -a
[root@localhost ~]$ sudo docker exec -it <_> bash
[oracle@1aeb761d5003 /]$
sqlplus sys/Oradoc_db1@localhost:1521/ORCLCDB.localdomain as sysdba

SQL> create pluggable database dbmonitor
        admin user dbmonitor_admin identified by dbmonitor
        roles = (DBA)
        FILE_NAME_CONVERT=('/u02/app/oracle/oradata/ORCL/pdbseed',
            '/u02/app/oracle/oradata/ORCL/DBMONITOR');
```

3. Verificou-se se a PDB estava montada e passível de ser utilizada:

```
    SQL> select pdb_name, status from cdb_pdbs;
```

4. Para a PDB estar disponível para utilização, foi necessário garantir a sua abertura sempre que a base de dados é ligada:

```
SQL> alter pluggable database DBMONITOR open;

SQL> create or replace trigger Sys.After_Startup after startup on database
    begin
        execute immediate 'alter pluggable database all open';
    end After_Startup; \
```

5. Adicionou-se os *tablespaces* (permanente e temporário) e respetivos *datafiles* à PDB:

```
SQL> connect sys/Oradoc_db1@localhost:1521/dbmonitor.localdomain as sysdba
SQL> create tablespace dbmonitordata datafile
    '/u02/app/oracle/oradata/ORCL/dbmonitordata01.dbf' SIZE 500M;
SQL> create temporary tablespace temp_dbmonitor tempfile
    '/u02/app/oracle/oradata/ORCL/pdb1/tempdbmonitor01.dbf' SIZE 500M;
```

6. Verificou-se a criação do *tablespace* e do *datafile* permanente, respetivamente:

```
SQL> select tablespace_name, con_id from cdb_tablespaces order by con_id;
SQL> select file_name, con_id from cdb_data_files order by con_id;
```

7. Verificou-se, ainda, a criação do *tablespace* temporário e do *tempfile*, respetivamente:

```
SQL> select tablespace_name, con_id from cdb_tablespaces order by con_id;
SQL> select tablespace_name, con_id from cdb_tablespaces
    where contents='TEMPORARY';
```

8. Criou-se um *local user* para a PDB criada com todos os previlégios:

```
SQL> connect sys/Oradoc_db1@localhost:1521/dbmonitor.localdomain as sysdba;
SQL> create user monitor identified by database;
SQL> grant all privileges to monitor;
```

9. Finalmente, é possível estabelecer uma nova conexão no SQL Developer com as seguintes credenciais:

```
utilizador: monitor;
password : database;
service_name : dbmonitor.localdomain
```

6 Modelo relacional de dados

Numa segunda fase, procedeu-se à modelação relacional de dados, de forma a albergar a informação previamente selecionada. Esta etapa permite representar uma Base de Dados como uma série de relacionamentos como forma de armazenar e processar os dados desta. O modelo relacional tem diversas vantagens, tais como permitir um elevado grau de independência de dados; permitir uma vista mais simplificada da estrutura da base de dados; diminuir a complexidade do trabalho de um DBA; eliminar problemas de redundância, consistência, entre outros.

Depois de uma análise cuidada aos dados a serem recolhidos, considerando sempre uma perspetiva histórica, foi construído o seguinte modelo relacional com as seguintes entidades:

- **DB_Monitor**;
- **Tablespaces**;
- **Users**;
- **Roles**;
- **Sessions**;
- **Datafiles**;
- **Program_Global_Area**;
- **System_Global_Area**;

Posteriormente, foi criado o modelo lógico. O modelo lógico é um dos modelos fundamentais para a conceção de uma Base de Dados. Este resulta de processos de normalização, através dos quais se obtém um conjunto de tabelas que permite um adequado registo dos dados, que vão de encontro aos requisitos/funcionalidades pretendidas.

O modelo lógico elaborado é composto pelas entidades anteriormente mencionadas e por novas entidades resultantes dos relacionamentos de N:N necessários (veja-se o exemplo do relacionameto entre as relações "User"e "Roles"). Na figura 3, é possível visualizar o modelo lógico criado que modela a Base de Dados de monitorização.

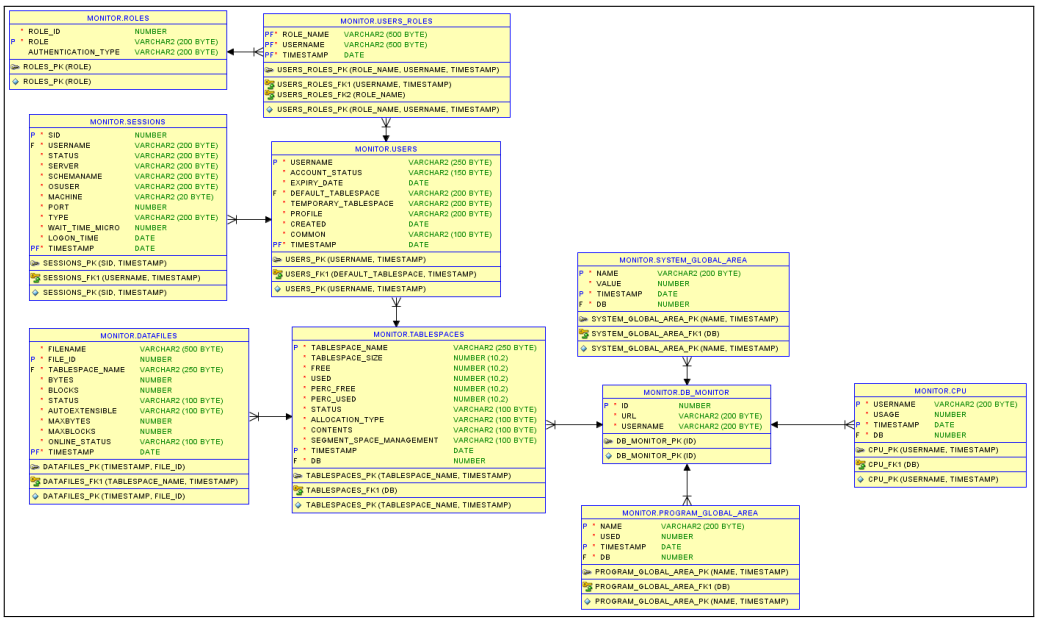

Figura 3: Modelo lógico criado

A construção do modelo físico foi realizada através da interface do **SQLDeveloper**. Para facilitar o acesso a algumas tabelas, optou-se por definir os valores nominais, juntamente com a marca temporal, como chaves primárias compostas de modo a identificar de forma única cada uma das relações (por exemplo, na relação "tablespaces", a chave primária é composta pelo tablespace_name e uma marca temporal, o seu timestamp).

7 Agente

Em seguida, foi necessária a criação de um agente que, periodicamente, efetue a consulta às vistas e tabelas e insera os dados nas tabelas previamente apresentadas.

Este agente foi desenvolvido na linguagem de programação Java, visto ser a linguagem com que o grupo se sente mais à vontade e facilita o processo de conexão. Para tal, usou-se o *JDBC*, uma API da linguagem que permite a comunicação com Bases de Dados SQL. Esta API permite, de uma forma simples, a troca de dados entre a aplicação e a Base de Dados.

7.1 Estabelecimento das conexões

De forma a estabelecer as conexões o agente actua da seguinte forma:

Passo 1: Inicialmente cria uma conexão como *System* de forma a ser possível ler e aceder às vistas e tabelas de administração enumeradas previamente;

Passo 2: Posteriormente cria uma nova conexão, mas agora com o utilizador criado (apresentado anteriormente), de forma a escrever os resultados que leu através da conexão criada no passo 1;

Passo 3: De seguida é tentado estabelecer a comunicação à base de dados através das conexões criadas no passo 1 e 2;

Passo 4: Caso se tenha obtido sucesso nas duas conexões, é criado um *Statement* para a conexão de forma a ser possível efectuar as operações pretendidas;

Passo 5: Após todo processo de construção, o agente entra num ciclo contínuo onde a cada 30 segundos vai efectuar a recolha da informação;

7.2 Recolha da informação

Quando o agente inicia uma nova recolha de informação, retira o *Timestamp*, atribuindo a cada linha a inserir nas tabelas respetivas o *Timestamp* atual.

7.2.1 DB_Monitor

De cada vez que o agente é iniciado, este cria uma nova entrada onde insere o nome da *Pluggable Database* ao qual está ligado e o utilizador pelo qual está ligado. Aqui, é atribuído um identificador utilizado para os relacionamentos com as outras tabelas.

7.2.2 Roles

Um dos parâmetros monitorizados que ajuda na administração das Bases de Dados são os *Roles*(grupo de permissões concedida a utilizadores).

Para este efeito, o agente verifica, através da conexão do utilizador criado, se a tabela utilizada para guardar a informação dos *Roles* já possui informação. Se não existir, quer dizer que se encontra na primeira iteração, por isso efetua a seguinte *query*:

```
SELECT DISTINCT ROLE_ID , ROLE , AUTHENTICATION_TYPE FROM DBA_ROLES;
```

Assim, é obtido o identificador do *role*, o nome deste e o tipo de autenticação. Posteriormente, é inserida esta informação na tabela criada para o efeito (Roles).

7.2.3 Tablespaces

Um dos parâmetros fulcrais de serem monitorizados é os *Tablespaces*. Para isso, efetuou-se a seguinte *query*:

```
SELECT DISTINCT T.TABLESPACE_NAME,
    ROUND((M.TABLESPACE_SIZE*8/1024),2) AS TABLESPACE_SIZE,
    ROUND((M.TABLESPACE_SIZE-M.USED_SPACE)*8/1024,2) AS FREE,
    ROUND(M.USED_SPACE*8/1024,2) AS USED,
    ROUND(100-M.USED_PERCENT,2) AS PERC_FREE,
    ROUND(M.USED_PERCENT,2) AS PERC_USED,
    T.STATUS, T.ALLOCATION_TYPE, T.CONTENTS,T.SEGMENT_SPACE_MANAGEMENT
FROM DBA_TABLESPACES T LEFT JOIN DBA_TABLESPACE_USAGE_METRICS M
    ON T.TABLESPACE_NAME = M.TABLESPACE_NAME
ORDER BY TABLESPACE_NAME ;
```

Através desta *query*, foi possível visualizar o nome do *tablespace*, o seu tamanho ocupado e livre em MB e a respetiva percentagem. Além destes parâmetros, decidiu-se monitoraizar o estado, o tipo de alocação, o conteúdo e a gestão autómatica do espaço livre.

7.2.4 Datafiles

Subsequentemente, de forma a monitorizar os *datafiles*, realizou-se a presente *query*, para recolher informação sobre os mesmos:

```
SELECT FILE_NAME , FILE_ID , TABLESPACE_NAME , BYTES , BLOCKS ,
    STATUS , AUTOEXTENSIBLE , MAXBYTES , MAXBLOCKS , ONLINE_STATUS
FROM DBA_DATA_FILES;
```

Decidiu-se recolher as seguintes informações: nome, identificador, *tablespace* correspondente, *bytes*, *blocks*, estado, se é auto extensível, número máximo de *bytes* e *block*, e se o estado é *online*.

7.2.5 Users

No seguimento da obtenção da informação dos *users*, foi implementada a seguinte *query*:

```
SELECT USERNAME , ACCOUNT_STATUS , EXPIRY_DATE , DEFAULT_TABLESPACE ,
    TEMPORARY_TABLESPACE , PROFILE , CREATED , COMMON
FROM DBA_USERS;
```

Obteve-se o nome, o estado da conta, a data de expiração, o *default tablespace*, o *tablespace* temporário, o tipo de perfil, quando foi criado e se é um utilizador comum.

7.2.6 Sessions

De forma a obter quais os *users* ligados, foi necessário considerar as sessões ligadas. Para tal, aplicou-se a *query* abaixo de forma a obter o SID, o nome do utilizador, o estado, o servidor, o nome do *schema*, o nome do sistema operativo, a máquina, porta, tipo, quantidade de tempo em espera e hora de *login*.

```
SELECT SID , USERNAME , STATUS , SERVER , SCHEMANAME , OSUSER ,
    MACHINE, PORT , TYPE , WAIT_TIME_MICRO , LOGON_TIME
FROM V$SESSION
WHERE USERNAME IS NOT NULL;
```

7.2.7 Roles dos Users

Com o objetivo de associar os roles que estão atribuídos aos utilizadores, efetuou-se a seguinte operação:

```
SELECT GRANTED_ROLE, GRANTEE
FROM DBA_ROLE_PRIVS
WHERE GRANTEE in (select USERNAME from DBA_USERS);
```

7.2.8 Program Global Area

Em relação à memória utilizada, decidiu-se monitorizar a denominada PGA (*Program Global Area*), que é uma região da memória que contém dados e informações de controlo para um processo do servidor. Para isso, acedeu-se à vista devida, encontrando qual o nome e parâmetros que estão inutilizados e alocados.

```
SELECT NAME, VALUE FROM V$PGASTAT
WHERE NAME='total PGA inuse' OR NAME='total PGA allocated';
```

7.2.9 System Global Area

Outra parâmetro referente à memória considerado foi a denominada SGA(*System Global Area*), que forma a parte da memória (RAM) do sistema partilhada por todos os processos pertencentes a uma única instância da Base de Dados Oracle.

```
SELECT NAME, VALUE FROM V$SGA;
```

7.2.10 CPU

Por fim, o último parâmetro recolhido e monitorizado foi o CPU (quantidade de tempo de CPU, em 10s de milissegundos, usada por uma sessão entre o início e o término de uma chamada do utilizador). Assim sendo, combinou-se diferentes vistas de forma a obter o valor deste.

```
    SELECT S.USERNAME, SUM(VALUE/100) as "CPU USAGE (SECONDS)"
    FROM V$SESSION S, V$SESSTAT T, V$STATNAME N
    WHERE T.STATISTIC# = N.STATISTIC#
        AND NAME LIKE '%CPU used by this session%'
        AND T.SID = S.SID
        AND S.STATUS='ACTIVE'
        AND S.USERNAME IS NOT NULL
    GROUP BY USERNAME, T.SID, S.SERIAL#;
```

7.3 Inserção da informação

Apresentado a forma como é recolhida a informação, o agente procede da seguinte forma para a inserção da mesma. Como o procedimento de escrita segue sempre o mesmo raciocínio, é apresentada apenas um excerto do código de forma a exemplificar a atuação do agente. Assim, é apresentada a inserção da informação dos *Users*. Para os parâmetros, o processo é idêntico. Apenas a informação recolhida varia, como a tabela de origem onde é escrita é diferente.

```
query = "SELECT USERNAME , ACCOUNT_STATUS , EXPIRY_DATE , DEFAULT_TABLESPACE
    , TEMPORARY_TABLESPACE , PROFILE , CREATED , COMMON FROM DBA_USERS";

ResultSet resultSetUsers = stmt.executeQuery(query);

while (resultSetUsers.next()) {
    String username = resultSetUsers.getString("USERNAME");
    String account_status = resultSetUsers.getString("ACCOUNT_STATUS");
    Date expiry_date = resultSetUsers.getDate("EXPIRY_DATE");
    String default_tablespace = resultSetUsers.getString("DEFAULT_TABLESPACE");
    String temporary_tablespace =
        resultSetUsers.getString("TEMPORARY_TABLESPACE");
    String profile = resultSetUsers.getString("PROFILE");
    Date created = resultSetUsers.getDate("CREATED");
    String common = resultSetUsers.getString("COMMON");

    query = "INSERT INTO USERS VALUES ('" + username + "','" + account_status +
        "','" + expiry_date + "','" + default_tablespace +
        "','" + temporary_tablespace + "','" + profile +
        "','" + created + "','" + common +
        "',TO_DATE('"+ timestamp +"','dd-mm-yyyy hh24:mi:ss'))"
    ;

    stmtwriter.executeQuery(query);
}
```

8 API REST

A apresentação das diversas informações ao utilizador requer uma componente capaz de ler conteúdos da Base de Dados, interpretando-os para a interface gráfica. Para este efeito, desenvolveu-se uma ***API REST*** em *NodeJS*, apoiado pela *framework Express*, que trata dos pedidos de acesso à Base de Dados, através de *queries* sobre a mesma. A API (a receber pedidos em localhost:5501), permite ao utilizador aceder a diversas informações existentes na Base de Dados. De entre estas destacam-se: consultar os utilizadores; visualizar os *tablespaces* juntamente com um histórico de espaço ocupado; entre outras.

8.1 Estabelecimento da conexão

Para se conectar à Base de Dados, o servidor usa as informações do utilizador definido no Capítulo 5, bem como o endereço do *localhost* e o nome da *Pluggable Database*(neste caso *dbmonitor*):

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

oracledb.initOracleClient({ libDir: 'C:\\Oracle\\instantclient_19_9' });

oracledb.getConnection( {
    user : "monitor",
    password : "database",
    connectString : "//localhost:1521/dbmonitor.localdomain"
})
```

8.2 Tratamento dos pedidos

A *API* é composta por duas componentes principais: um *router* e um *controller*.

O *router* é responsável pelo encaminhamento dos pedidos, por parte da interface, para o *controller*. Os resultados destes pedidos são retornados em formato **JSON**, facilitando o manuseamento dos dados na interface.

No que diz respeito ao *controller*, este executa *queries* sobre a Base de Dados, utilizando a conexão anteriormente apresentada. Cada uma destas *queries* recorre à função *queryObject* que estabelece a conexão à Base de Dados e executa o método *execute*, de forma a obter os dados presentes na Base de Dados. Para tal, é passado como parâmetro a *query* SQL a ser executada.

Apresenta-se, em seguida, um pedido de *datafiles*, correspondentes a um *tablespace* num determinado *timestamp*.

```
function queryObject(sql, bindParams, options) {
    options['outFormat'] = oracledb.OBJECT;

    return new Promise(function (resolve, reject) {
        oracledb.getConnection({
            user: "monitor",
            password: "database",
            connectString: "//localhost:1521/dbmonitor.localdomain"})
            .then(function (connection) {
                connection.execute(sql, bindParams, options)
                .then(function (results) {
                    resolve(results);
                    process.nextTick(function () {
                        oracleDbRelease(connection);
                    });
                }).catch(function (err) {
                    reject(err);
                    process.nextTick(function () {
                        oracleDbRelease(connection);
                    });
                });
            }).catch(function (err) {
                reject(err);
            });
        });
}


checkConnection.getDataFilesTablespace = (tName, tTS) => {
    return queryObject("SELECT * FROM DATAFILES D
            WHERE D.TABLESPACE_NAME = '"+ tName + "'
            AND D.TIMESTAMP = to_date('" + tTS + "','DD-MM-YYYY HH24:MI:SS')",
        {}, {outFormat: ""});
}
```

9 Interface Web

O monitor de base de dados assenta na divisão de micro-serviços, isto é, foi implementada uma API em **NodeJS** que efetua a conexão à Base de Dados e retorna os resultados em *JSON* (Capítulo 8). Posteriormente, foi implementada uma interface, com recurso à *framework* **Vue.js**, capaz de interpretar respostas em *JSON* provenientes da API. A escolha desta *framework* recaiu na facilidade e capacidade de personalização dos componentes, permitindo a criação de uma *interface* agradável e intuitiva.

A divisão em micro-serviços permitiu a distinção clara entre os serviços *backend* e *frontend* da aplicação.

Um dos objectivos delimitados consistiu no desenvolvimento de uma interface que permitisse a apresentação dos dados mais relevantes para a monitorização de uma Base de Dados. Assim sendo, foi implementado um menu dinâmico que permite a seleção da informação que se pretende visualizar/monitorizar. A seleção de um elemento deste menu, reencaminha para um conjunto de dados e indicadores relativos a esse parâmetro. De forma a facilitar a monitorização recorreu-se a tabelas, gráficos, e outras representações visuais.

9.1 Arquitectura

O fluxo de pedidos e respostas realizado entre a API REST e a Interface Web é o seguinte:

1. Quando é realizado um pedido à interface, é iniciado um fluxo de informação entre vários componentes;
2. A interface inicia a página correspondente ao pedido mediante o que se encontra definido nas *routes* da interface;
3. No entretanto, a interface envia os pedidos necessários à API, através das rotas definidas;
4. A API recebe estes pedidos e invoca o método *queryObject* com a *query* correspondente;
5. Os dados resultantes desta execução são enviados para a interface;
6. Finalmente, a interface trata os dados recebidos (em formato *JSON*), para posterior apresentação.

9.2 Resultados Obtidos

Figura 4: Menu inicial do monitor.

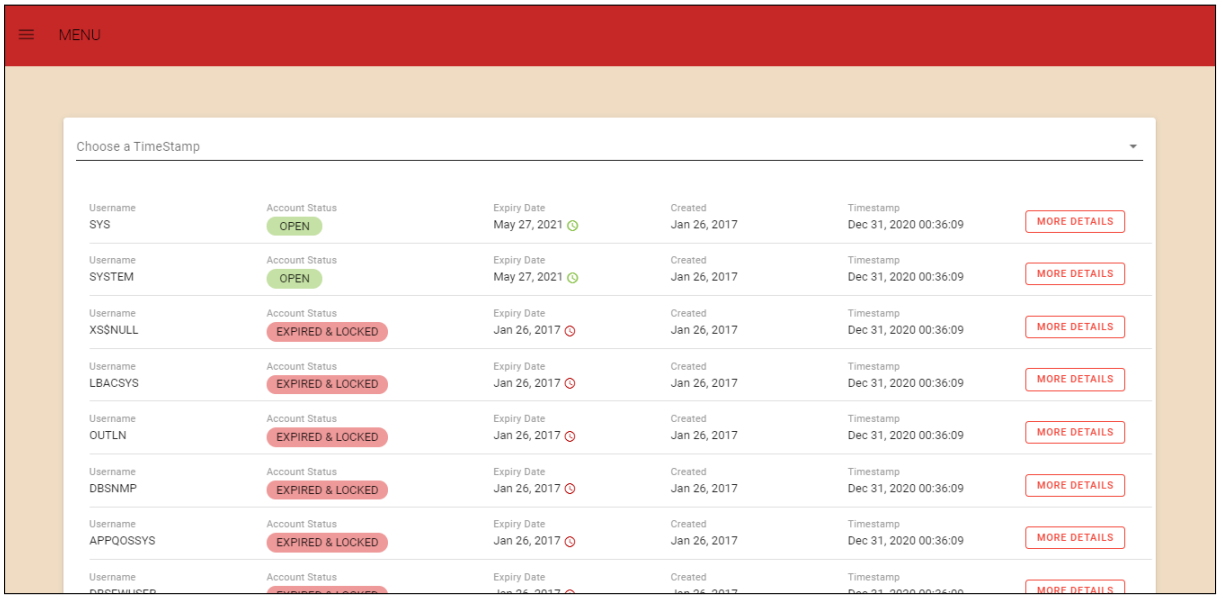

Figura 5: Página com a lista de utilizadores.

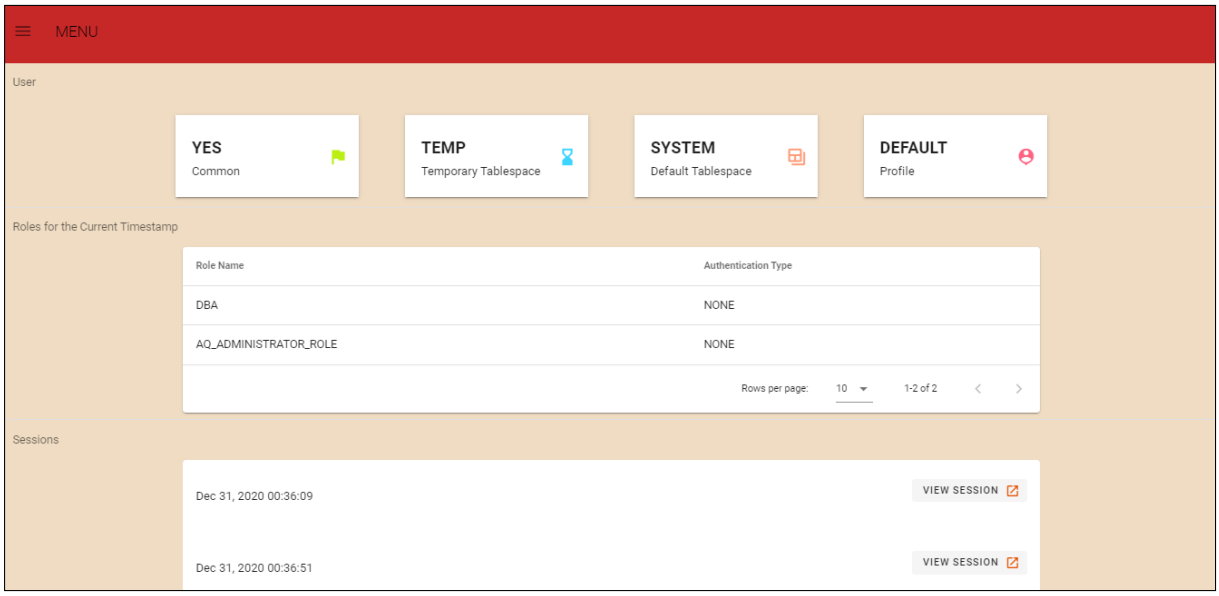

Figura 6: Informações de um utilizador.

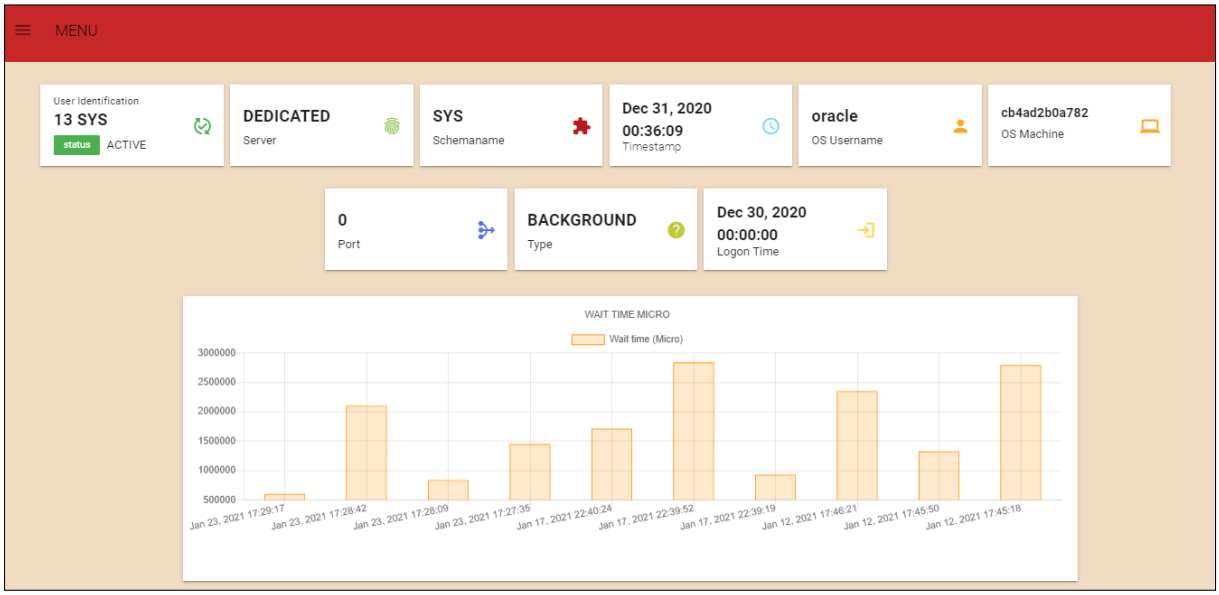

Figura 7: Informações de uma sessão.

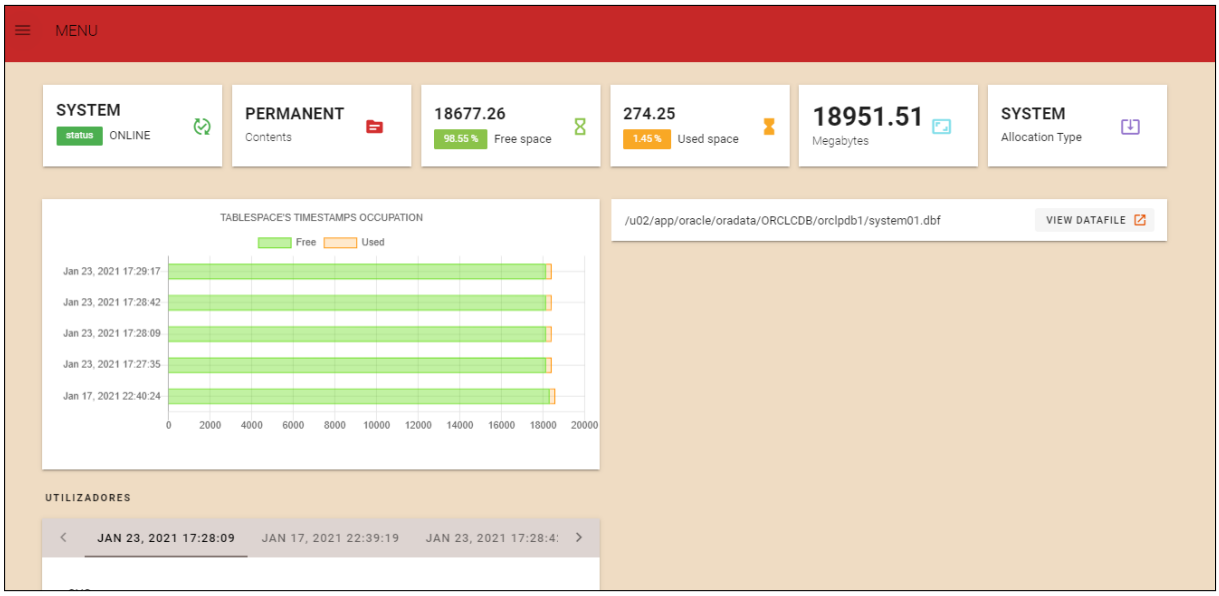

Figura 8: Detalhes de um *tablespace*.

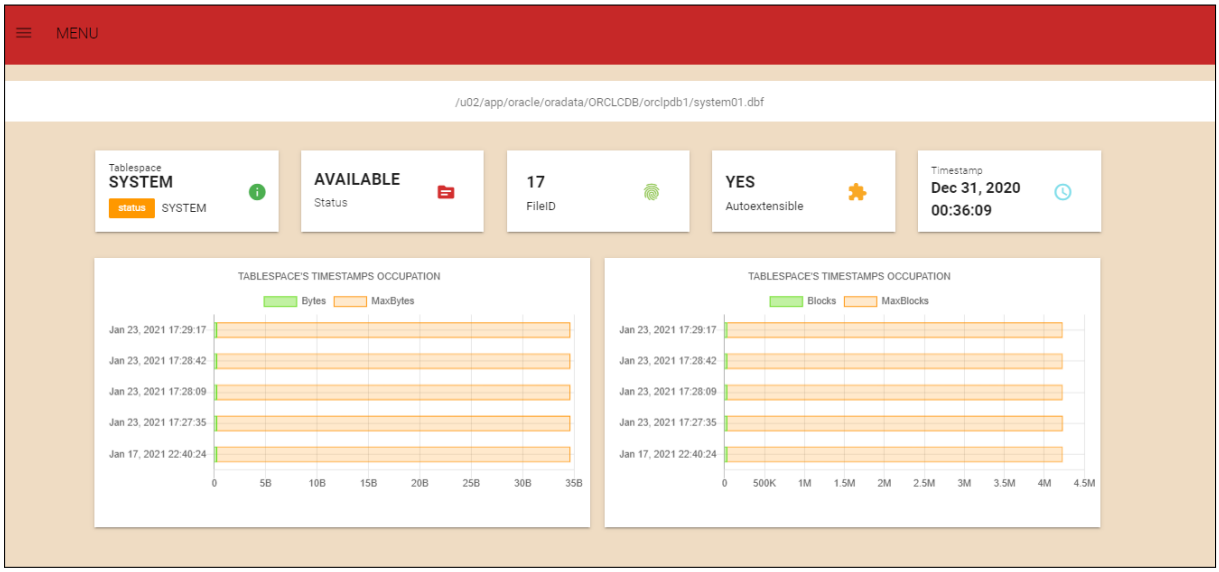

Figura 9: Detalhes de um *datafile*.

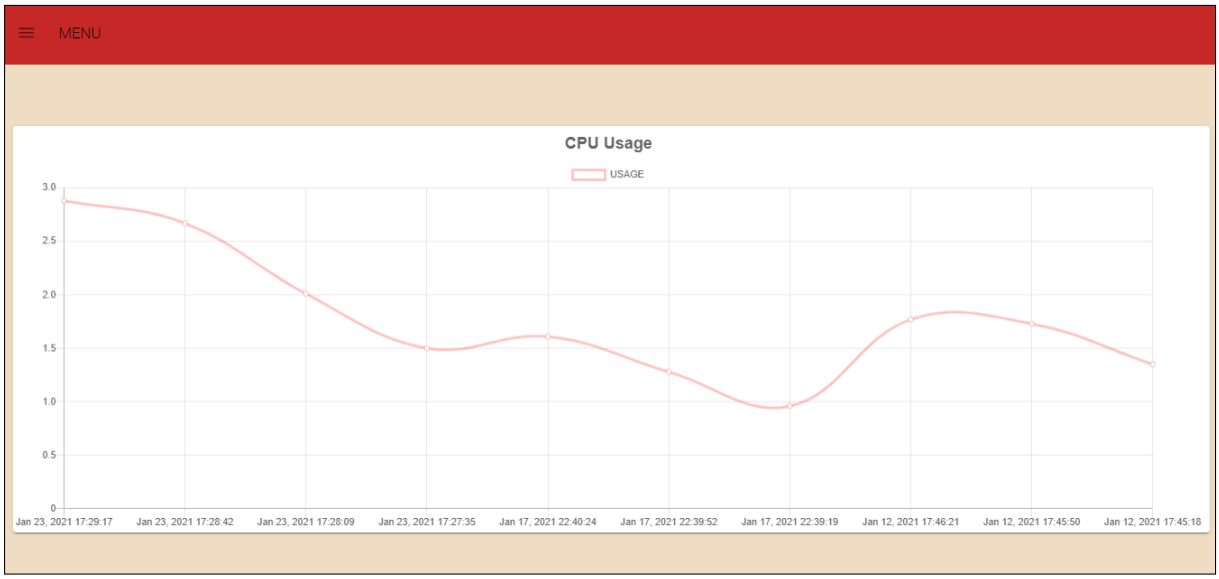

Figura 10: Utilização do CPU.

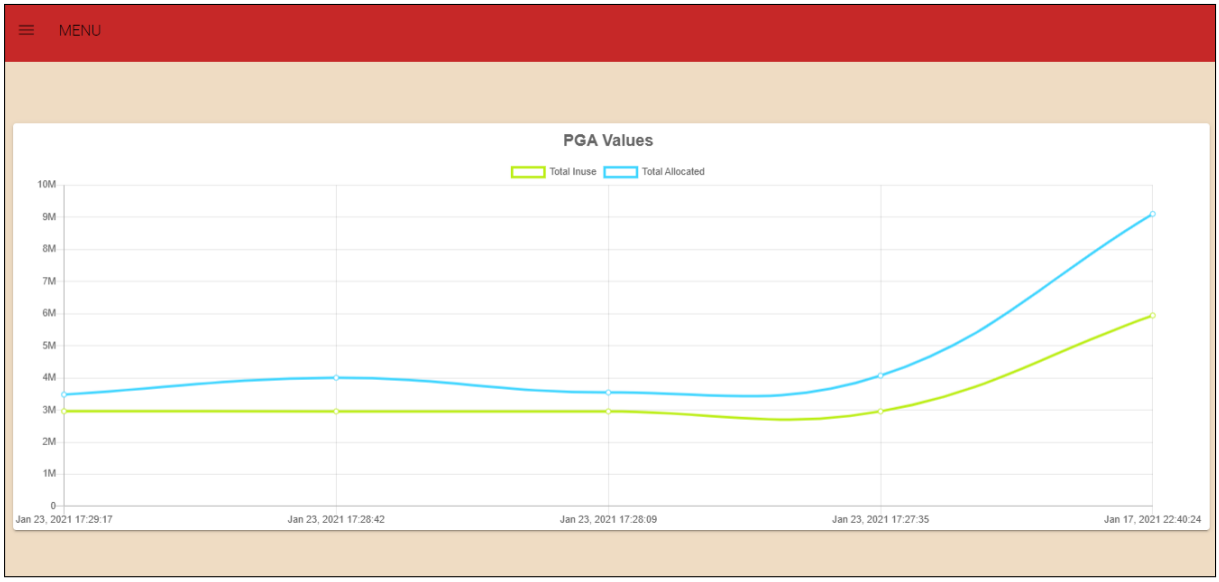

Figura 11: Utilização da área especial de memória para processos, PGA.

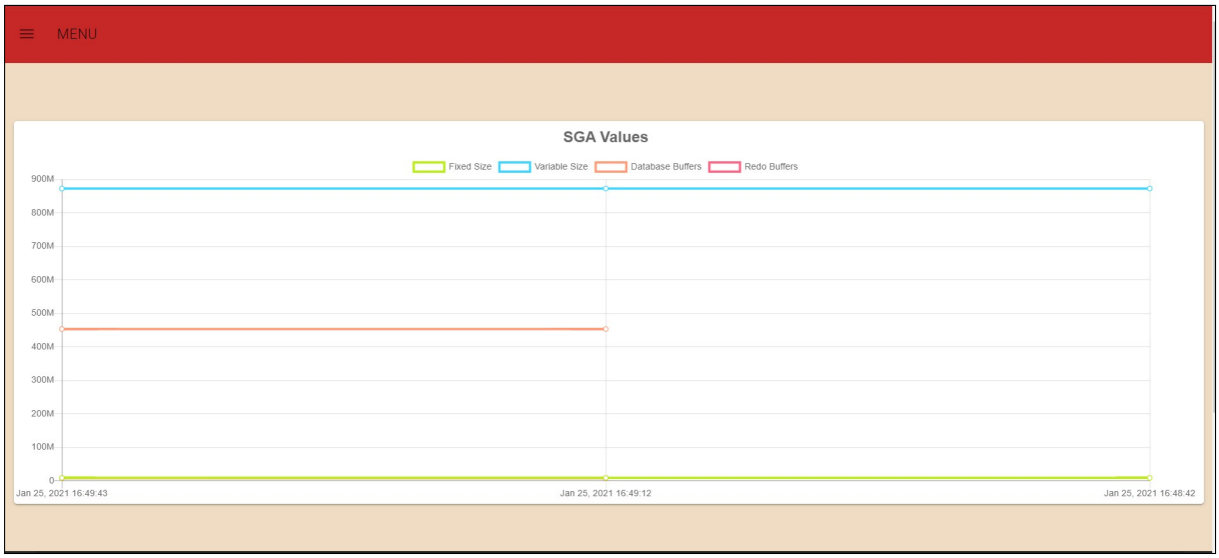

Figura 12: Utilização da área de memória compartilhada por todos os processos, SGA.

10 Conclusão

O presente relatório descreveu, de forma sucinta, o processo de implementação de um monitor de base de dados que apresenta de forma simples os principais parâmetros de avaliação de performance de uma base de dados Oracle.

Durante a elaboração deste trabalho surgiram algumas dificuldades que, com o esforço e entusiasmo do grupo pelo resultado final, acabaram por ser ultrapassadas. Entre as quais destacam-se o desafio que foi a realização da interface por não ser uma área em que os elementos do grupo se sentissem muito confortáveis dado o percurso académico até ao momento; a dificuldade de tratamento de dados em formatos de data na transição da Base de Dados para a API; e a seleção da informação mais pertinente dada a quantidade elevada de dados disponíveis para tarefas de monitorização.

Numa perspectiva futura, considera-se que seria interessante a atualização automática de determinados componentes presentes na interface (gráficos, por exemplo), bem como a criação de um sistema de *login* que permitisse acesso a diferentes Bases de Dados, ou até a adição de mais indicadores para complementar o processo de monitorização.

Após a realização deste trabalho, o grupo ficou consciente dos diferentes parâmetros presente numa de bases de dados da Oracle utilizados para à sua administração e exploração.

Considerou-se que os objectivos propostos com a realização deste trabalho foram cumpridos, bem como a consolidação dos conhecimentos na área de administração e exploração de bases de dados.

Por fim, o grupo espera que os conhecimentos obtidos e consolidados sejam de enorme utilidade tendo uma perspectiva futura.

Referências

[1] *Database Reference*. 2021. URL: https://docs.oracle.com/cd/B19306_01/server.102/b14237/toc.htm (acedido em 20/01/2021).

[2] *DB-engines Ranking 2021*. 2021. URL: https://db-engines.com/en/ranking (acedido em 20/01/2021).