



Universidade do Minho

Escola de Engenharia

Administração e Exploração de Base de Dados

Trabalho Prático – Database Manager

Diogo Araújo A78485;

Maria Pinto PG39292;

Mariana Fernandes A81728;

Mateus Silva A81952;

Conteúdo

| | | |
|------|--|----|
| 1. | Contextualização | 3 |
| 2. | Conceptualização da Base de Dados..... | 4 |
| 2.1. | Modelo Conceptual | 5 |
| 2.2. | Tabelas e Atributos | 6 |
| 2.3. | Modelo Lógico | 8 |
| 2.4. | Utilizador | 9 |
| 2.5. | Modelo Físico..... | 9 |
| 3. | Agente de Recolha de Informação..... | 9 |
| 4. | API REST | 11 |
| 5. | Interface Web | 12 |
| 6. | Conclusão | 16 |

1. Contextualização

Neste trabalho prático será apresentada a concepção, implementação e desenvolvimento dum sistema informático de forma a demonstrar um monitor básico da base de dados Oracle, que de forma simples nos fornece gráficos e/ou tabelas sobre vários parâmetros de performance como o número de sessões, a carga do CPU e memória mas também informações sobre as *tablespaces* e *datafiles*, incluindo toda a *metadata* envolvida pelos mesmos, como o seu armazenamento livre e a sua estrutura lógica associada.

Para tal desenvolveu-se um agente de recolha de informação na linguagem *Javascript*, mais concretamente em *Node.js* que através da extração de dados das *views* administrativas da base de dados e posteriormente armazenar esse conhecimento útil num *schema* criado especialmente para este projeto e propósito.

Com esta informação já colocada de forma analítica e filtrada nesse *schema* falado anteriormente utilizou-se uma *API RESTful* fornecida automaticamente pela Oracle que nos facilitou para obter os dados em JSON para a sua apresentação no *front-end* que foi criada em HTML5 e que é responsiva de forma a poder-se visualizar esta monitorização em qualquer tipo de dispositivo sem a ocultação de dados cruciais.

2. Conceptualização da Base de Dados

Neste trabalho prático, o grupo direcionou-se para a Base de Dados *Pluggable orcl*. Inicialmente, sentimos a necessidade de planejar como iríamos realizar o armazenamento e gestão da informação a nível da base de dados Oracle. Tivemos também em conta os users que iríamos utilizar para a monitorização e acesso à base de dados, assim como as suas permissões, sendo que existiam vários utilizadores com permissões diferentes entre si:

- *sys.cdb*: este utilizador é administrador da *Container Database (CDB)*, ou seja, acede e gere todos os dados da base de dados original denominada *root*.
- *sys.orcl*: este utilizador é administrador da *Pluggable DataBase (PDB)*, ou seja, acede e gere todos os dados da respetiva PDB.
- *grupo7.orcl*: temos vários utilizadores comuns desta *Pluggable Database (PDB)*, em que o grupo7 irá ser o utilizador criado pelo nosso grupo do projeto. Este utilizador irá criar e gerir a base de dados que irá ser apresentada numa interface *Web*.

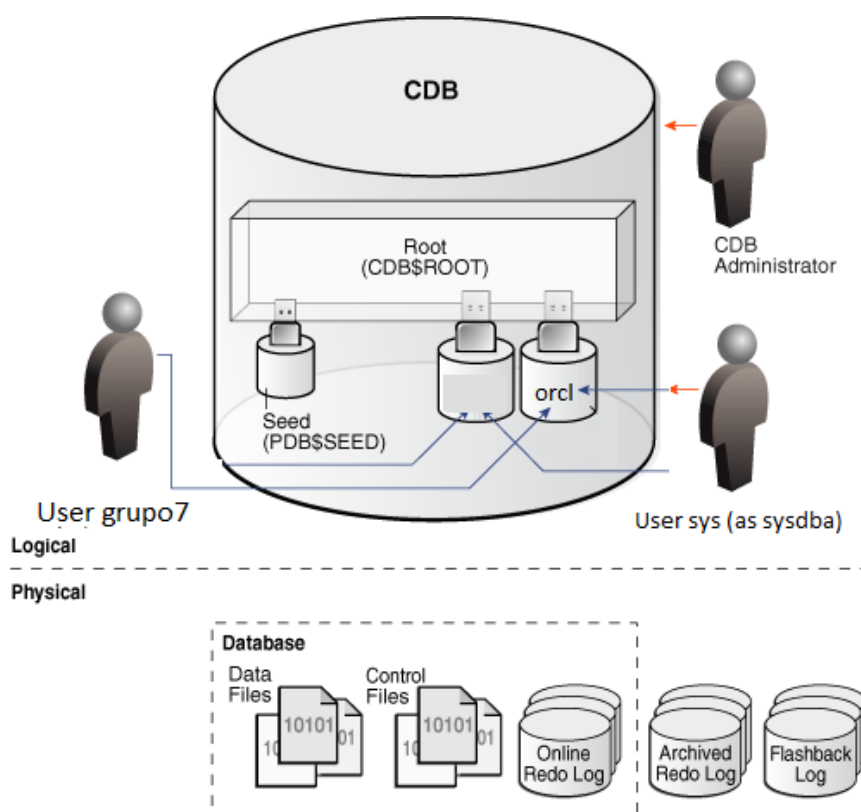
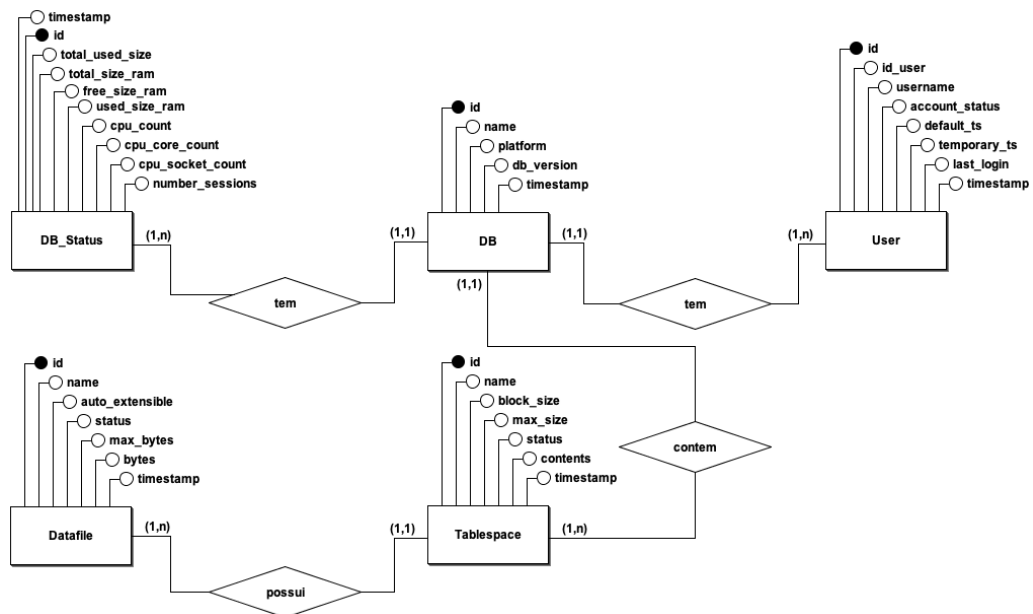


Figure 1 - Estrutura da base de dados utilizada no projeto

2.1. Modelo Conceptual



Para a criação do modelo conceptual tivemos que analisar as várias *views* disponíveis na parte administrativa da Base de Dados, que acedemos tanto pela PDB, através do *user sys* mas também pela CDB, para informações gerais. Fizemos uma análise das *views* para percebermos que informação das tabelas necessitávamos de extrair. O modelo conceptual que desenhamos exhibe-se da seguinte forma:

Concluimos que as tabelas que melhor se adequavam ao resultado pretendido, eram: DB_status, DB, User, Tablespace e Datafile.

Figure 2 - Modelo conceptual da base de dados a criar

2.2. Tabelas e Atributos

O grupo descreveu detalhadamente o significado das entidades e atributos que implementou nas tabelas criadas:

A tabela *DB* possui os seguintes atributos: *id*, *name*, *platform*, *db_version* e *timestamp*.

| Atributos | Tipo | Descrição |
|------------|-----------|--|
| id | Integer | Identificador de uma instância da tabela |
| name | Varchar | Designação da base de dados |
| platform | Varchar | Designação da plataforma onde se encontra a BD |
| db_version | Varchar | Designação da versão da BD |
| timestamp | Timestamp | Timestamp da recolha dos dados |

A tabela *DB_status* possui os seguintes atributos: *id*, *total_size_ram*, *free_size_ram*, *used_size_ram*, *total_used_size*, *cpu_count*, *cpu_core_count*, *cpu_socket_count*, *number_sessions* e *timestamp*.

| Atributos | Tipo | Descrição |
|------------------|-----------|---|
| id | Integer | Identificador de uma instância da tabela |
| total_size_ram | Integer | Total da RAM alocada (MB) |
| free_size_ram | Integer | Total de RAM livre (MB) |
| used_size_ram | Integer | Total de RAM em uso (MB) |
| Total_used_size | Integer | Total de espaço físico em uso (MB) |
| cpu_count | Integer | Número de CPU da base de dados |
| cpu_core_count | Integer | Número de <i>cores</i> de CPU |
| cpu_socket_count | Integer | Número de <i>sockets</i> de CPU |
| number_sessions | Integer | Número de sessões ativas na base de dados |
| timestamp | Timestamp | Timestamp da recolha dos dados |

A tabela *User* possui os seguintes atributos: *id*, *id_user*, *username*, *account_status*, *default_ts*, *temporary_ts*, *last_login* e *timestamp*.

| Atributos | Tipo | Descrição |
|----------------|-----------|---|
| id | Integer | Identificador de uma instância da tabela User |
| id_user | Integer | Identificador de uma instância da tabela User |
| username | Varchar | Nome de um utilizador |
| account_status | Varchar | Estado da conta |
| default_ts | Varchar | Tablespace de padrão |
| temporary_ts | Varchar | Tablespace temporária |
| last_login | Timestamp | Timestamp do último login do utilizador |
| timestamp | Timestamp | Timestamp da recolha dos dados |

A tabela *Tablespace* possui os seguintes atributos: *id*, *name*, *block_size*, *max_size*, *status*, *contents* e *timestamp*.

| Atributos | Tipo | Descrição |
|------------|-----------|---|
| id | Integer | Identificador de uma instância da tabela Tablespace |
| name | Varchar | Designação da tablespace |
| block_size | Integer | Tamanho da tablespace (Bytes) |
| max_size | Integer | Tamanho máximo de uma tablespace (Bytes) |
| status | Integer | Estado da tablespace |
| contents | Integer | Estado do conteúdo da tablespace |
| timestamp | Timestamp | Timestamp da recolha dos dados |

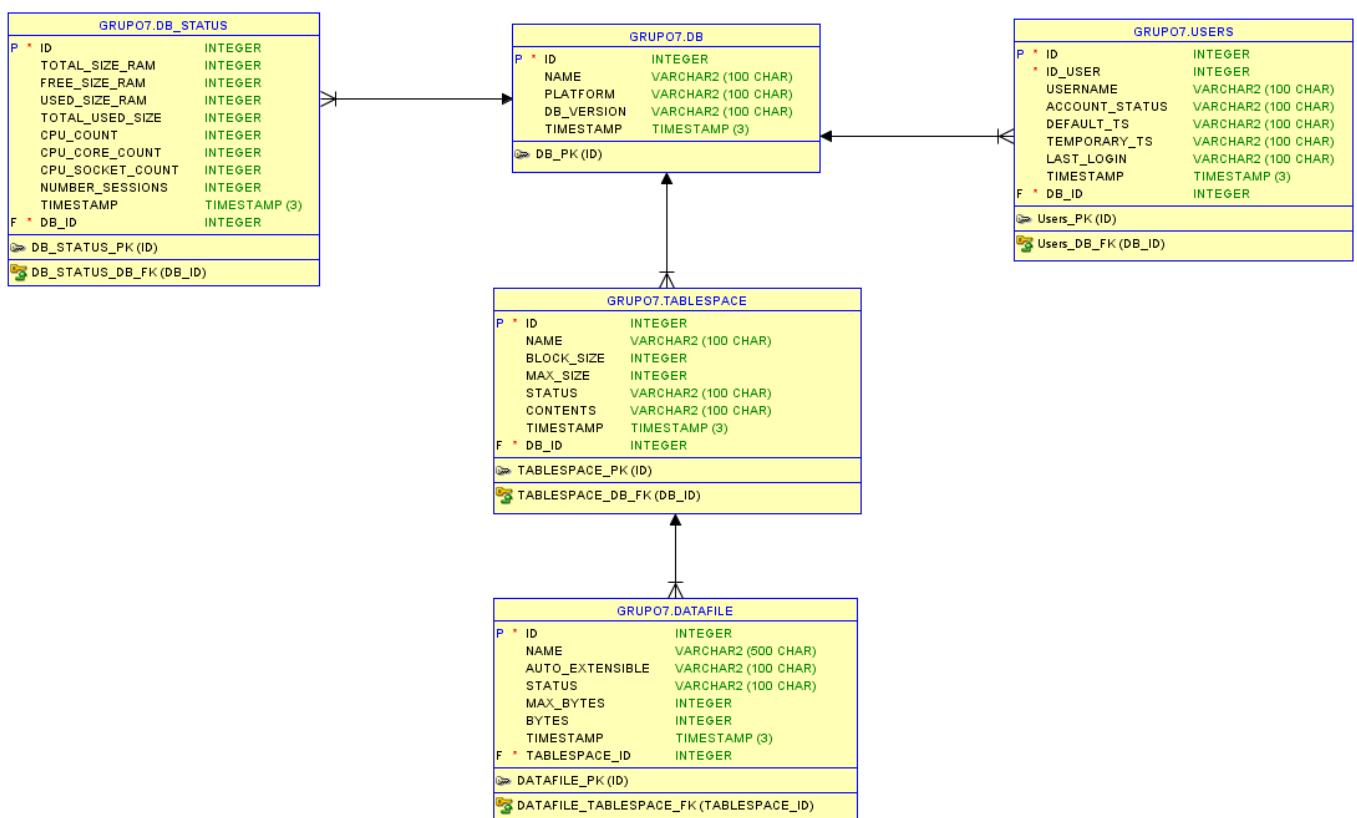
A tabela *Datafile* possui os seguintes atributos: *id*, *name*, *auto_extensible*, *status*, *max_bytes*, *bytes* e *timestamp*.

| Atributos | Tipo | Descrição |
|-----------------|---------|---|
| id | Integer | Identificador de uma instância da tabela Tablespace |
| name | Varchar | Designação do datafile |
| auto_extensible | Varchar | Indica se o datafile tem esta propriedade |
| status | Varchar | Estado do datafile |

| | | |
|-----------|-----------|--|
| max_bytes | Integer | Capacidade máxima de bytes no datafile (Bytes) |
| bytes | Integer | Tamanho do datafile (Bytes) |
| timestamp | Timestamp | Timestamp da recolha dos dados |

2.3. Modelo Lógico

Depois da construção do Modelo Conceptual, o grupo realizou o Modelo Lógico no SQL Developer, tendo em conta as restrições das chaves primárias e estrangeiras de cada tabela. O resultado obtido foi o seguinte:



Com a realização deste Modelo Lógico verificou-se que o modelo é composto por cinco tabelas (*DB*, *DB_status*, *User*, *Tablespace*, *Datafile*) que pertencem a uma única base de dados. Verificamos também que uma Tablespace é composta por vários Datafiles e cada um destes está apenas associado a apenas uma Tablespace. O mesmo acontece com a base de dados (BD), sendo que, a BD pode ser composta por vários Users. A DB também tem associada a sua tabela a DB_status que significa que a DB tem vários estados que nos dão informações muito importantes, como a CPU, memória, tamanho, número de sessões, entre outras.

Durante o processo de criação do modelo lógico, tivemos cuidado por forma a que este se encontre normalizado até à terceira forma normal (3FN)

2.4. Utilizador

Foi criado um utilizador "grupo7" com as permissões necesssárias para que este possa ser o utilizador com o qual vamos estabelecer a conexão entre o agente de recolha de informação e a nossa DB, como explicaremos no próximo capítulo.

```
CREATE USER grupo7 IDENTIFIED BY grupo7
  DEFAULT TABLESPACE grupo7
  TEMPORARY TABLESPACE grupo7_temp
  QUOTA UNLIMITED ON grupo7
  QUOTA UNLIMITED ON grupo7
  ACCOUNT UNLOCK;

GRANT
  CREATE SESSION,
  DELETE ANY TABLE,
  INSERT ANY TABLE,
  SELECT ANY TABLE,
  UPDATE ANY TABLE
TO grupo7;
```

2.5. Modelo Físico

A partir do modelo lógico mostrado anteriormente, desenvolveu-se o modelo físico, presente no ficheiro *oracleSchema.dll*, onde, em primeiro lugar, procedemos à eliminação das tabelas (para facilitar o processo de importar o modelo físico novamente), e de seguida, à criação dos *tablespaces*, do utilizador (como mostrado acima), das tabelas, de acordo com o modelo lógico desenhado, e por fim sequências e *triggers* que permitem que o SGBD faça a gestão própria dos índices, por forma a facilitar a lógica de negócio implementada no agente de recolha de informação.

3. Agente de Recolha de Informação

Para popular a nossa base de dados com informação sobre o status da DB que estamos a monitorizar, desenvolvemos um agente de recolha de informação em

NodeJS, utilizando o package node-oracledb, que permite estabelecer conexões a servidores oracle.

Assim sendo, o primeiro passo foi criar uma função que permitisse estabelecer conexões recebendo como parâmetro os dados do utilizador através do qual nos queremos conectar.

```
function connect(user, pass, cs) {  
  return oracledb.getConnection({  
    user: user,  
    password: pass,  
    connectString: cs  
  });  
}
```

Ao todo, são feitas 3 conexões, uma com o utilizador system à pugglable database, outra com o utilizador grupo7, também à pluggable database, e por fim, com o system à container database. Assim é-nos possível aceder a todos os dados necessários para o povoamento, que é feito unicamente através do utilizador grupo7.

A partir deste ponto, bastou usar as conexões estabelecidas para executar, em primeiro lugar, selects às tabelas com os dados que precisamos, e de seguida updates ou inserts nas nossas tabelas. A estratégia usada foi, depois de obtermos os dados, por cada linha obtida, tentamos fazer update dessa mesma linha na nossa base de dados. Caso nenhuma linha tenha sido afetada, procedemos a inserir os dados.

No entanto, no caso da tabela DB_STATUS não pretendemos atualizar os dados existentes, mas sim criar uma nova linha, com timestamp, pelo que se procede apenas à execução do insert.

Neste processo, alteramos alguns dados para que fossem mais amigáveis de visualizar, nomeadamente o espaço utilizado/livre, que foi convertido de Bytes para MBytes.

Para que os dados sejam atualizados, usou-se a função do NodeJS *setInterval()*, que, de 15 em 15 segundos chama a função *update()*, que efetua as operações indicadas acima.

| Conexão | Tabela de Origem | Campo de Origem | Tabela de Destino | Campo de Destino |
|---------|------------------|-----------------|-------------------|------------------|
| PDB | V\$DATABASE | NAME | DB | NAME |

| | | | | |
|-----|--------------------------|-----------------------------------|------------|------------------|
| PDB | V\$DATABASE | PLATFORM_NAME | DB | PLATFORM |
| CDB | DBA_CPU_USAGE_STATISTICS | VERSION | DB | DB_VERSION |
| PDB | DBA_TABLESPACES | TABLESPACE_NAME | TABLESPACE | NAME |
| PDB | DBA_TABLESPACES | BLOCK_SIZE | TABLESPACE | BLOCK_SIZE |
| PDB | DBA_TABLESPACES | MAX_SIZE | TABLESPACE | MAX_SIZE |
| PDB | DBA_TABLESPACES | STATUS | TABLESPACE | STATUS |
| PDB | DBA_TABLESPACES | CONTENTS | TABLESPACE | CONTENTS |
| PDB | DBA_DATA_FILES | FILE_NAME | DATAFILE | NAME |
| PDB | DBA_DATA_FILES | AUTOEXTENSIBLE | DATAFILE | AUTO_EXTENSIBLE |
| PDB | DBA_DATA_FILES | STATUS | DATAFILE | STATUS |
| PDB | DBA_DATA_FILES | MAXBYTES | DATAFILE | MAX_BYTES |
| PDB | DBA_DATA_FILES | BYTES | DATAFILE | BYTES |
| PDB | DBA_USERS | USER_ID | USERS | ID_USER |
| PDB | DBA_USERS | USERNAME | USERS | USERNAME |
| PDB | DBA_USERS | ACCOUNT_STATUS | USERS | ACCOUNT_STATUS |
| PDB | DBA_USERS | DEFAULT_TABLESPACE | USERS | DEFAULT_TS |
| PDB | DBA_USERS | TEMPORARY_TABLESPACE | USERS | TEMPORARY_TS |
| PDB | DBA_USERS | LAST_LOGIN | USERS | LAST_LOGIN |
| CDB | V\$SGA | ROUND(SUM(VALUE)/1024/1024) | DB_STATUS | TOTAL_SIZE_RAM |
| CDB | V\$SGASTATUS | ROUND(SUM(BYTES)/1024/1024) | DB_STATUS | FREE_SIZE_RAM |
| CDB | V\$SGA/ V\$SGASTATUS | TOTAL_SIZE_RAM- USED_SIZE_RAM | DB_STATUS | USED_SIZE_RAM |
| PDB | DBA_DATA_FILES | ROUND(SUM(BYTES)/1024/1024) | DB_STATUS | TOTAL_USED_SIZE |
| CDB | DBA_CPU_USAGE_STATISTICS | CPU_COUNT | DB_STATUS | CPU_COUNT |
| CDB | DBA_CPU_USAGE_STATISTICS | CPU_CORE_COUNT | DB_STATUS | CPU_CORE_COUNT |
| CDB | DBA_CPU_USAGE_STATISTICS | CPU_SOCKET_COUNT | DB_STATUS | CPU_SOCKET_COUNT |
| CDB | V\$SESSION | COUNT(*) (where username != null) | DB_STATUS | NUMBER_SESSIONS |

4. API REST

O programa SQL Developer tem um excelente serviço incorporado que permite a inicialização de um serviço REST, ou seja, o Oracle REST Data Services permite desenvolver interfaces REST para as bases de dados que nos facilitou imenso na realização deste projeto.

Numa fase inicial, precisamos de nos conectar à base de dados pretendida, neste caso o grupo7.orcl. De seguida, basta clicarmos na opção *"REST Services"* e seguidamente, clicar em *"Enable Rest Services.."* para ativarmos o serviço REST.

Seguidamente, teremos de configurar o *RESTFUL Services Wizard*. Para isso, marcamos a caixa definida como *"Enable Schema"* que, normalmente já vem marcada por defeito. Definimos também o *"Schemas alias"* com o nome da nossa base de dados (também costuma vir implementada por defeito). Por fim, clicamos em *"Seguinte"* onde nos vão ser mostradas as opções selecionadas anteriormente e basta clicar em *"Terminar"*.

Depois de ativar o *REST service* entre o utilizador e a nossa base de dados (grupo7.orcl), teremos também de ativar para todas as tabelas para o qual pretendemos, ter acesso através da API REST. Sendo assim, teremos de clicar com o botão do lado direito do rato em todas as tabelas pretendidas, e clicarmos na opção “*Enable REST Service...*”.

Para facilitar este processo, escolhemos usar o *script ordsActivate*, que faz precisamente a ativação dos serviços REST. Permite também a personalização de rotas, que utilizamos para obter os dados dos *datafiles*, visto que era necessário juntar estes com os dados da tabela *tablespaces*, para que seja possível visualizar a que *tablespace* pertence cada *datafile*.

Depois de ativarmos o REST service para a base de dados e tabelas pretendidas, uma interface REST é criada automaticamente pelos ORDS, assim como a nossa API. Para a utilização da API de forma a obtermos dados, teremos de executar um pedido GET, que retorna um JSON com a informação pretendida. Para a realização destes pedidos, teremos de abrir um separador de browser e escrever a seguinte fórmula:

<http://localhost:8080/ords/nome da BD/nome da Tabela/>

Basta substituir o nome da BD e nome da Tabela, pelos respetivos nomes que pretendemos obter os dados. Um exemplo de um pedido à PDB na tabela DB_status, com o utilizador grupo 7, será:

http://localhost:8080/ords/grupo7/DB_status/

Se quisermos ir buscar os dados da *query* personalizada, acedemos então ao endereço:

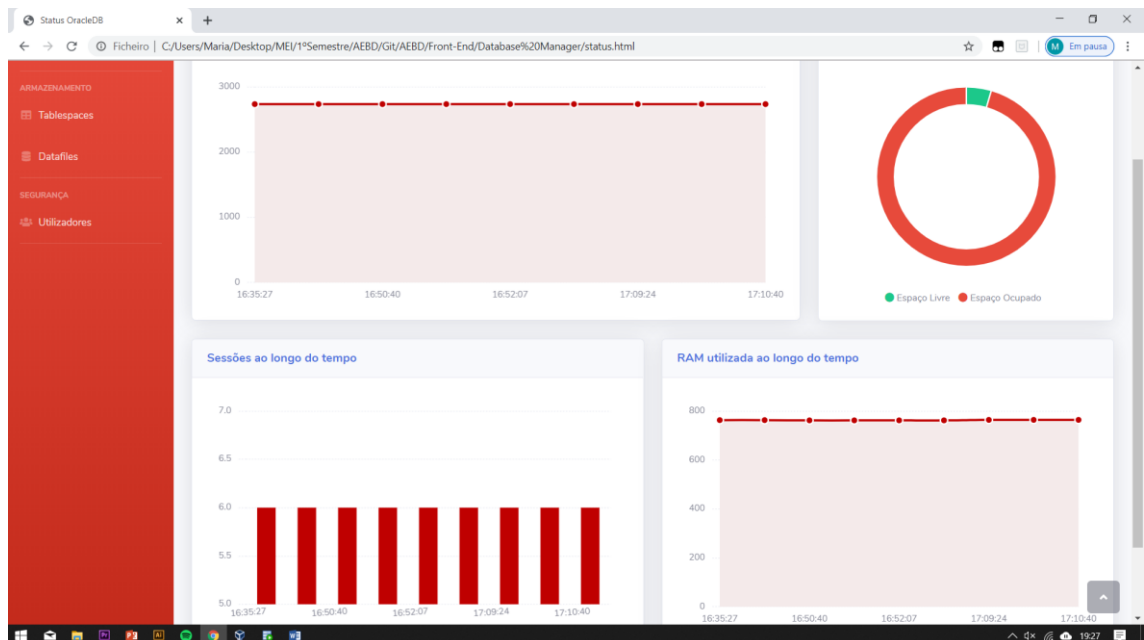
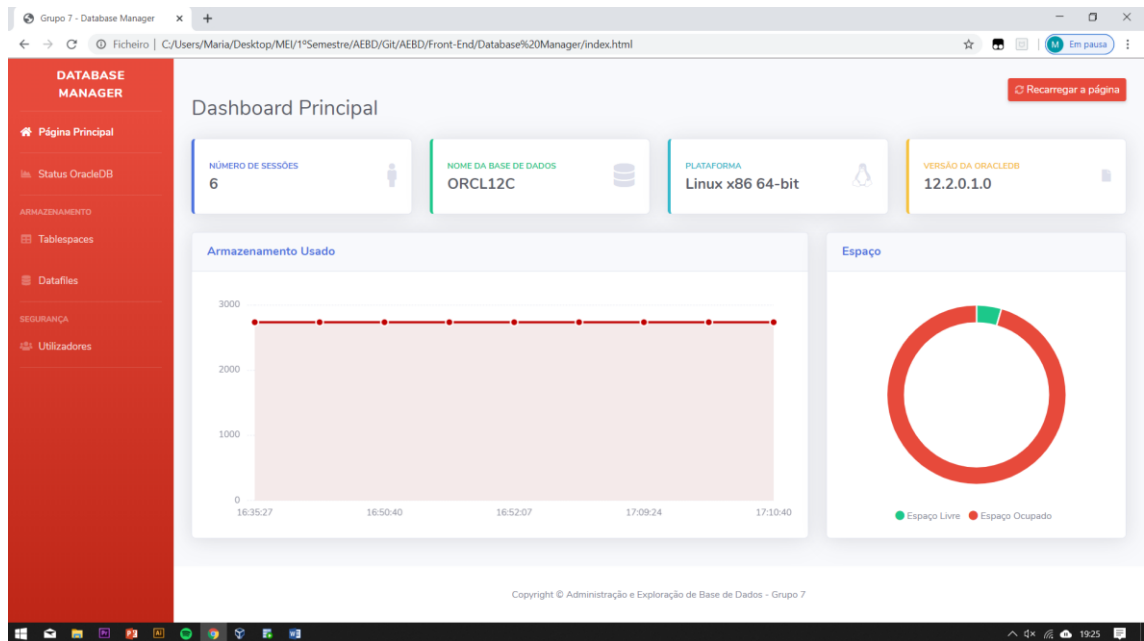
<http://localhost:8080/ords/grupo7/personalizado/1/>

5. Interface Web

Por último, desenvolvemos uma interface *Web*, para que o utilizador possa ter uma visualização geral e intuitiva referentes aos dados da base de dados.

Implementamos esta interface *Web*, a partir do serviço *REST*, em que combinamos o HTML de um *Bootstrap* já existente com a interpretação do código *JSON* enviado pela *API Rest*.

Desenvolvemos uma página *Web* com as informações gerais sobre a base de dados e também criamos uma página individual para cada tabela que pretendíamos visualizar (*Tablespaces*, *Users*, *Datafiles* e *DB Status*).



MANAGER

Página Principal

Status OracleDB

ARMAZENAMENTO

Tablespaces

Datafiles

SEGURANÇA

Utilizadores

Tablespaces

NÚMERO DE TABLESPACES

10

Tabela dos tablespaces da DB

| ID_TABLESPACE | NAME | BLOCK_SIZE | MAX_SIZE | STATUS | CONTENTS |
|---------------|-------------|------------|------------|--------|-----------|
| 1 | SYSTEM | 8192 | 2147483645 | ONLINE | PERMANENT |
| 2 | SYSAUX | 8192 | 2147483645 | ONLINE | PERMANENT |
| 3 | UNDOTBS1 | 8192 | 2147483645 | ONLINE | UNDO |
| 4 | AEBD_TABLES | 8192 | 2147483645 | ONLINE | PERMANENT |
| 5 | AEBD_TEMP | 8192 | 2147483645 | ONLINE | TEMPORARY |
| 6 | USERS | 8192 | 2147483645 | ONLINE | PERMANENT |
| 7 | TEMP | 8192 | 2147483645 | ONLINE | TEMPORARY |
| 8 | GRUPO7_TEMP | 8192 | 2147483645 | ONLINE | TEMPORARY |

DATABASE
MANAGER

Página Principal

Status OracleDB

ARMAZENAMENTO

Tablespaces

Datafiles

SEGURANÇA

Utilizadores

Utilizadores

NÚMERO DE UTILIZADORES

25

Tabela dos utilizadores da DB

| ID_USER | USERNAME | ACCOUNT_STATUS | DEFAULT_TS | TEMPORARY_TS | LAST_LOGIN |
|------------|-----------|------------------|------------|--------------|--------------|
| 0 | SYS | OPEN | SYSTEM | TEMP | 01:00:00 |
| 9 | SYSTEM | OPEN | SYSTEM | TEMP | Invalid Date |
| 2147483638 | XS\$NULL | EXPIRED & LOCKED | SYSTEM | TEMP | 01:00:00 |
| 55 | APPQOSSYS | EXPIRED & LOCKED | SYSAUX | TEMP | 01:00:00 |
| 101 | LBACSYS | EXPIRED & LOCKED | SYSTEM | TEMP | 01:00:00 |
| 54 | DBSNMP | EXPIRED & LOCKED | SYSAUX | TEMP | 01:00:00 |
| 60 | GGSYS | EXPIRED & LOCKED | SYSAUX | TEMP | 01:00:00 |
| 13 | OUTLN | EXPIRED & LOCKED | SYSTEM | TEMP | 01:00:00 |

Datfiles

C:\Users\Maria\Desktop\MEI\1ºSemestre\AEBD\Giu\AEBD\Front-End\Database%20Manager\datafiles.html

DATABASE MANAGER

Página Principal

Status OracleDB

ARMAZENAMENTO

Tablespaces

Datfiles

SEGURANÇA

Utilizadores

Datfiles

NÚMERO DE DATFILES
7

Tabela dos datfiles da DB

| | AUTO_EXTENSIBLE | STATUS | MAX_BYTES | BYTES | TABLESPACE_NAME |
|--|-----------------|-----------|-------------|------------|--------------------|
| :12c/orcl/system01.dbf | YES | AVAILABLE | 34359721984 | 367001600 | SYSTEM |
| :12c/orcl/sysaux01.dbf | YES | AVAILABLE | 34359721984 | 1279262720 | SYSAUX |
| :12c/orcl/undotbs01.dbf | YES | AVAILABLE | 34359721984 | 398458880 | UNDOTBS1 |
| :12c/orcl/users01.dbf | YES | AVAILABLE | 34359721984 | 81264640 | USERS |
| :12c/orcl/APEX_1941389856444596.dbf | YES | AVAILABLE | 26279936 | 7929856 | APEX_1941389856444 |
| .:2/db_1/dbs/u01apporcleoraedataorc12crlaeab_tables_01.dbf | NO | AVAILABLE | 0 | 524288000 | AEBD_TABLES |
| .:2/db_1/dbs/GRUPO7 | YES | AVAILABLE | 34359721984 | 209715200 | GRUPO7 |

6. Conclusão

Com a realização deste projeto, pretendemos criar uma interface simples e interativa que fosse de fácil utilização para qualquer tipo de utilizador, de modo a interpretar as informações mostradas, sem qualquer dificuldade. Pensamos que nesse aspeto, conseguimos um bom resultado. Na nossa opinião, este projeto serviu para o grupo aprender bastante relativamente ao funcionamento e estrutura de uma base de dados Oracle.

Tivemos algumas dificuldades, como por exemplo, o desenvolvimento da estrutura de armazenamento de dados, e que informações escolher guardar e mostrar na interface gráfica.

Em suma, foi um projeto que nos permitiu aprofundar os nossos conhecimentos sobre bases de dados *oracle* e como monitorizar as mesmas. A par disto, as nossas capacidades de programação web foram melhoradas, visto termos implementado o agente em NodeJS, e termos desenvolvido uma interface web para a visualização dos dados.