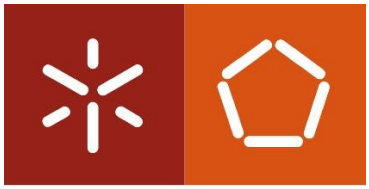

Universidade do Minho

Departamento de Informática

Administração e Exploração de Base de Dados

Trabalho Prático

Maria Silva (A83840), Mariana Marques (A85171), Miguel Solino (A86435), Paulo Lima (A89983)

27 de janeiro de 2021

Conteúdo

Capítulo 1

Introdução

Utilizando os conhecimentos adquiridos na componente prática e teórica da Unidade Curricular de Administração e Exploração de Base de Dados o grupo construiu um monitor de Base de Dados que apresenta de forma simples os principais parâmetros de avaliação de performance de uma Base de Dados Oracle.

A plataforma criada permite visualizar os dados mais importantes para manutenção e gestão de um sistema de Bases de Dados utilizando gráficos e tabelas claras e sucintas, o que permite uma avaliação mais centrada nos dados do que na compreensão da disposição dos mesmos.

Capítulo 2

Implementação

2.1 Ferramentas utilizadas na implementação

Tendo em conta a escolha de base de dados que nos foi proposta como desafio deste trabalho, decidimos utilizar como principal ferramenta para implementação das diferentes partes do projeto o NodeJS. Esta escolha foi tomada graças à excelente documentação disponível e simplicidade de utilização presente na biblioteca utilizada para realizar a comunicação entre a base de dados.

Sendo assim, para recolha dos dados foi utilizado NodeJS juntamente com a biblioteca *node-oracledb*, para RESTFUL API foi utilizado também NodeJS e Express e para interface web foi utilizada a framework *VueJS* e a biblioteca *ChartJS*, sendo estas todas assentes na linguagem de programação *JavaScript*.

Também como um dos objetivos do trabalho e também a pensar no futuro, tudo isto foi incluído em docker containers utilizando a ferramenta Docker e Docker Compose, tornando o processo de correr a aplicação apenas com um comando, sem preocupações a nível de networking facilitando o deployment mais tarde.

2.2 Desenvolvimento do Projeto

2.2.1 Criação da PDB

Com o objetivo de guardar e utilizar com maior eficiência os dados relativos à BD a monitorizar, foi necessário criar uma BD de monitorização. A ideia é que nesta BD sejam guardados dados referentes à outra de 20 em 20 segundos. Todas as queries SQL utilizadas na criação da PDB estão apresentadas nos anexos no final do relatório. Assim sendo, esta apresenta as seguintes tabelas criadas com o utilizador GRUPO10:

- Database
- Activity
- Users
- Resources
- Performance

Apresenta-se, em seguida, o modelo relacional, para facilitar a compreensão do esquema e respetivos relacionamentos e restrições.

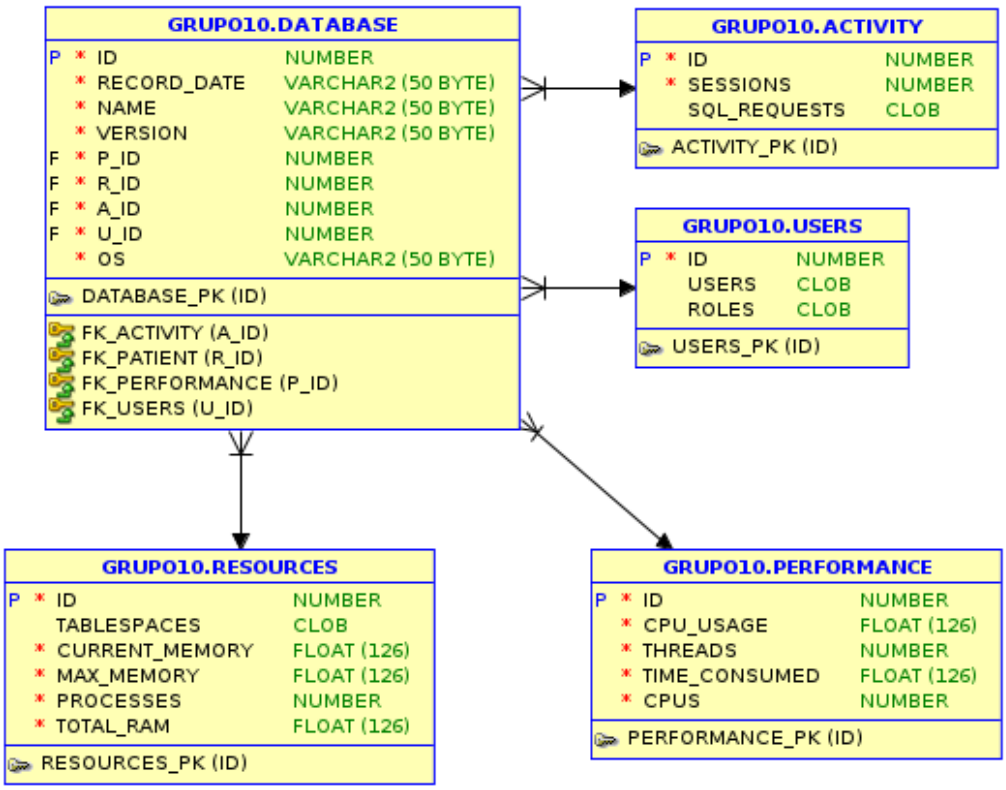

Figura 2.1

2.2.2 Conexão da base de dados e Recolha dos Dados

Após instalação e configuração da PDB, passamos à integração desta com a aplicação em desenvolvimento. Para o processo de conexão entre ambas as partes utilizamos o *Node-Oracledb*, pacote oficial de NodeJS para ligação a bases de dados Oracle.

Tendo a conexão configurada entre as bases de dados (à de recolha e à qual onde vamos guardar os dados), partimos para a escrita de queries que obtivesse os dados todos que pretendíamos guardar e, de seguida através da segunda conexão guardar, também através de queries, os dados nas tabelas respetivas.

2.2.3 RESTFUL API

Para que mais tarde pudesse ser feita uma interface web na qual os dados seriam apresentados, tivemos que desenvolver uma RESTFUL API que possui vários endpoints responsáveis por devolver as informações de cada tabela e extras úteis para as views da interface implementadas. Abaixo fica a documentação dos endpoints existentes:

- /database
- /performance
- /resources
- /activity
- /users

/database endpoint

Este endpoint fornece o último registo das informações gerais acerca da base de dados. Esta informação é dada por um pedido do tipo **GET**. O servidor respondendo com o estado 200 (OK) fornecerá então um *JSON* com os campos:

- Date
- Name
- OS
- Version

Adicionalmente a este endpoint poderá ser fornecido um número (Ex: /database/30) que fará com que o servidor forneça os últimos 30 registos recolhidos e não apenas um.

/performance endpoint

Este endpoint fornece o último registo das informações da performance relativas à base de dados. Esta informação é dada por um pedido do tipo **GET**. O servidor respondendo com o estado 200 (OK) fornecerá então um *JSON* com os campos:

- Number of CPUs
- CPU Usage
- Number of Threads
- Time Consumed

Adicionalmente a este endpoint poderá ser fornecido um número (Ex: /performance/30) que fará com que o servidor forneça os últimos 30 registos recolhidos e não apenas um.

/resources endpoint

Este endpoint fornece o último registo das informações dos recursos relativas à base de dados. Esta informação é dada por um pedido do tipo **GET**. O servidor respondendo com o estado 200 (OK) fornecerá então um *JSON* com os campos:

- Tablespaces
- Current Memory
- Max Memory
- Processes
- Total Ram

/activity endpoint

Este endpoint fornece o último registo das informações da atividade relativas à base de dados. Esta informação é dada por um pedido do tipo **GET**. O servidor respondendo com o estado 200 (OK) fornecerá então um *JSON* com os campos:

- Number of Sessions
- SQL Requests

Adicionalmente a este endpoint poderá ser fornecido um número (Ex: /activity/30) que fará com que o servidor forneça os últimos 30 registos recolhidos e não apenas um.

/users endpoint

Este endpoint fornece o último registo das informações dos users e roles relativas à base de dados. Esta informação é dada por um pedido do tipo **GET**. O servidor respondendo com o estado 200 (OK) fornecerá então um *JSON* com os campos:

- Users
- Roles

2.2.4 Interface Web

Com a RESTFUL API operacional, utiliando os endpoints que criamos mostramos os dados em caixas de informação, tabelas, gráficos de porções e gráficos de linhas com base no tempo. Nos gráficos de porções são considerados os tamanhos totais e são divididos pelas porções a que cada indicador corresponde. Nos gráficos de linhas por tempo são mostrados os últimos trinta registos. No capitulo do Produto Final serão mostradas as três views criadas: Página inicial, Users e Activity.

2.3 Arquitetura Final do Projeto

Após implementação da aplicação, a arquitetura final tem o seguinte aspeto:

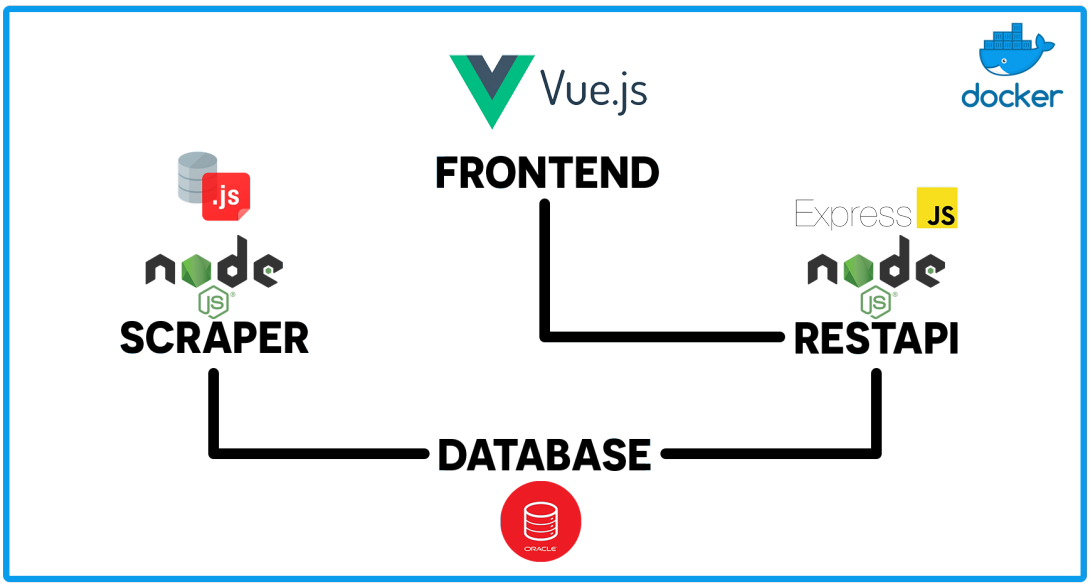

Figura 2.2

No "backend"o scraper e a RESTFUL API acedem diretamente às bases de dados através de queries (Todas as queries SQL utilizadas nesta API para recolher os dados estão presentes nos anexos no final deste relatório.) enquanto que frontend nunca o faz, tendo acesso aos dados apenas através da API documentada. Desta forma, o backend é independente do frontend.

2.4 Produto Final

Abaixo, iremos demonstrar onde se encontra e como estão apresentadas todas as informações recolhidas da base de dados.

2.4.1 Página inicial

Na página inicial é possível ver todas as principais informações acerca da base de dados como o nome, informações de espaço, recursos e data do último registo recolhido. Além disso em todas as páginas é apresentado um menu lateral esquerdo que serve para podermos navegar nas diferentes views existentes.

Figura 2.3

Figura 2.4

2.4.2 View Users

Na view Users são apresentados o número de users, o número de roles e os users e roles existentes como as informações de cada um.

Figura 2.5

2.4.3 View Activity

Na página Activity estão apresentados dados referentes a sessões atualmente ativas, sessões ativas ao longo do tempo, número de pedidos sql nas últimas 24 horas e o conteúdo dos mesmos.

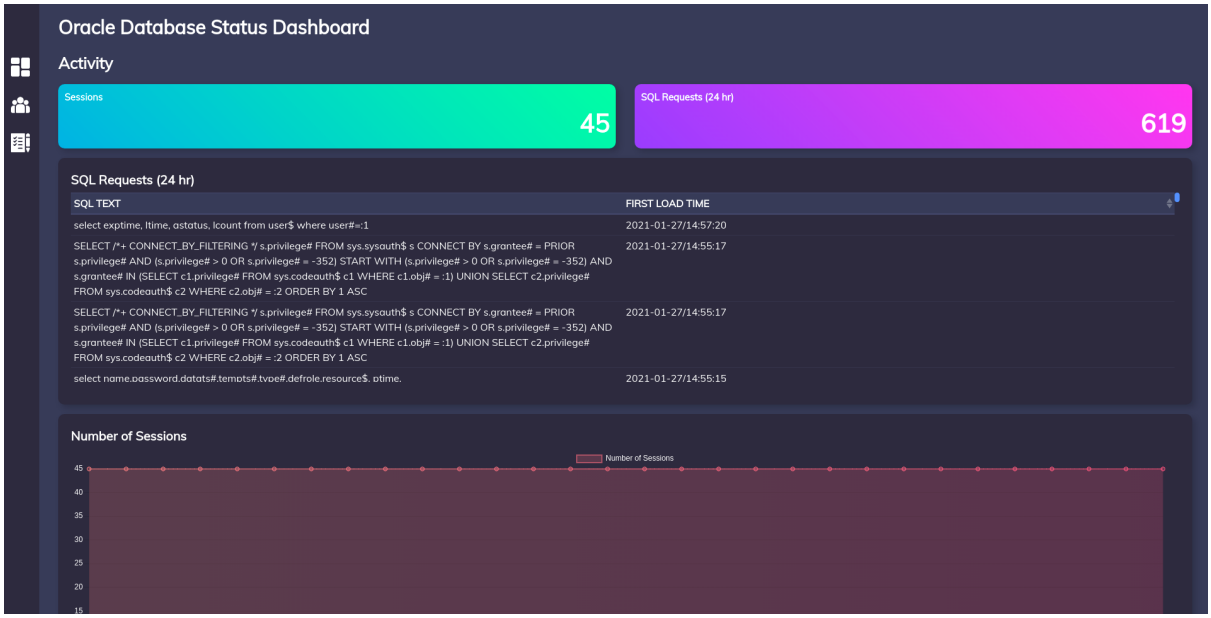

Figura 2.6

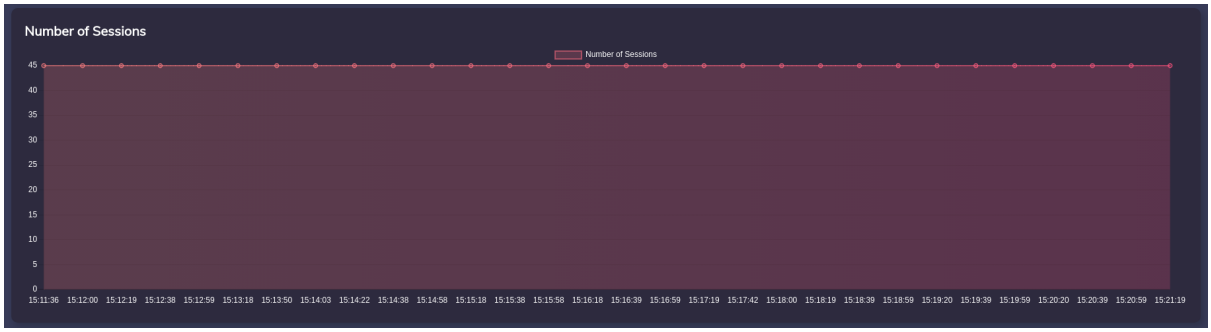
Number of Sessions
Number of Sessions
45
40
35
30
25
20
15
10
5
0
15:11:36
15:12:00
15:12:19
15:12:38
15:12:59
15:13:18
15:13:50
15:14:03
15:14:22
15:14:38
15:14:58
15:15:18
15:15:38
15:15:58
15:16:18
15:16:39
15:16:59
15:17:19
15:17:42
15:18:00
15:18:19
15:18:39
15:18:59
15:19:20
15:19:39
15:19:59
15:20:20
15:20:39
15:20:59
15:21:19

Figura 2.7

Capítulo 3

Conclusão

Durante o desenvolvimento do projeto procuramos encontrar as melhores soluções para agilizar os principais entraves na criação do schema Oracle. Procuramos também orientar o projeto, de forma a este incidir em funcionalidades com mais potencial para facilitar a visualização de toda a informação recolhida pelo scraper. Esta conceptualização foi essencial para delinear um trajeto orientado aos nossos objetivos.

Inicialmente foi necessário perceber quais as formas corretas de implementar a recolha de dados das views da Base de Dados, este passo foi fundamental para percebermos qual o formato mais indicado para demonstrar os dados no sistema final.

O plano de desenvolvimento, passou pela delineação de três tópicos principais, recolha, processamento, e apresentação gráfica, isto permitiu organizar o desenvolvimento do sistema proposto.

Desta forma, no final da primeira fase, sentimos que a estruturação do nosso projeto ficou ilustrada de uma forma bastante clara e objetiva, consolidando assim a matéria lecionada e facilitando desta forma o processamento de dados.

No terceiro e ultimo tópico do projeto, o design gráfico da aplicação permitiu demonstrar toda a recolha e processamento de dados feitos ate ao momento, o design teve como intuito encadear os vários componentes, e foi concebido, através da análise detalhada dos requisitos recolhidos na base de dados, e com vista no comportamento da interface com o utilizador.

Após todo o processo de implementação, o resultado final é funcional e acessível, satisfaz os objetivos pretendidos, acima de tudo, é flexível e utilizável o que sugere que trará bastante utilidade.

Capítulo 4

Webgrafia

https://github.com/oracle/node-oracledb
https://nodejs.org/en/
https://webpack.js.org/
https://expressjs.com/
https://vuejs.org/
https://www.chartjs.org/

Capítulo 5

Anexos

5.1 Criação da PDB

```
sqlplus sys/Oradoc_db1 as sysdba

create pluggable database aebd_tables
        admin user grupo10 identified by grupo10
        roles = (DBA)
        FILE_NAME_CONVERT=('/u02/app/oracle/oradata/ORCL/pdbseed','/u02/app/oracle/

alter pluggable database all open;

create tablespace aebd_tablespace datafile
   '/u02/app/oracle/oradata/ORCL/aebd/aebd_tables.dbf' SIZE 1000M;
create temporary tablespace aebd_tables_temp tempfile
   '/u02/app/oracle/oradata/ORCL/aebd/aebd_tables_temp.dbf' SIZE 100M;

ALTER USER grupo10 quota unlimited on aebd_tablespace;

CREATE TABLE database (
    id              NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    record_date     DATE NOT NULL,
    name            VARCHAR2(50) NOT NULL,
    os              VARCHAR2(50) NOT NULL,
    version         VARCHAR2(50) NOT NULL,
    p_id            NUMBER NOT NULL,
    r_id            NUMBER NOT NULL,
    a_id            NUMBER NOT NULL,
    u_id            NUMBER NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT FK_PERFORMANCE   FOREIGN KEY (p_id) REFERENCES
       performance(id),
    CONSTRAINT FK_PATIENT       FOREIGN KEY (r_id) REFERENCES
       resources(id),
    CONSTRAINT FK_ACTIVITY      FOREIGN KEY (a_id) REFERENCES
       activity(id),
    CONSTRAINT FK_USERS         FOREIGN KEY (u_id) REFERENCES users(id)
    )
    TABLESPACE aebd_tablespace;

CREATE TABLE performance (
```

```
    id              NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    cpus            NUMBER      NOT NULL,
    cpu_usage       FLOAT       NOT NULL,
    threads         NUMBER      NOT NULL,
    time_consumed   FLOAT   NOT NULL,
    PRIMARY KEY (id)
    )
    TABLESPACE aebd_tablespace;

CREATE TABLE resources (
    id              NUMBER  GENERATED BY DEFAULT ON NULL AS IDENTITY,
    tablespaces    CLOB CHECK (tablespaces IS JSON),
    current_memory  FLOAT   NOT NULL,
    max_memory      FLOAT   NOT NULL,
    processes       NUMBER NOT NULL,
    total_ram       FLOAT  NOT NULL,
    PRIMARY KEY (id)
    )
    TABLESPACE aebd_tablespace;

CREATE TABLE activity (
    id              NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    sessions        NUMBER NOT NULL,
    sql_requests    CLOB CHECK (sql_requests IS JSON),
    PRIMARY KEY (id)
    )
    TABLESPACE aebd_tablespace;


CREATE TABLE users (
    id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
    users CLOB CHECK (users IS JSON),
    roles CLOB CHECK (roles IS JSON),
    PRIMARY KEY (id)
    )
    TABLESPACE aebd_tablespace;
```

5.2 Queries utilizadas na RESTFUL API

```
-----------
-- DATABASE
-----------

-- OS & Name
select name, platform_name
from v$database;

-- Oracle version
SELECT PRODUCT, VERSION FROM SYS.PRODUCT_COMPONENT_VERSION WHERE
   PRODUCT LIKE 'Oracle%';

-- Current Date
SELECT TO_CHAR
    (SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "CURRENT DATE"
```

```
     FROM DUAL;


--------------
-- PERFORMANCE
--------------

-- Number of cpus, cpu time consumed, cpu usage (%)
select
    max(num_cpus),
    sum(cpu_consumed_time),
    sum(avg_cpu_utilization)
from V$RSRCMGRMETRIC ;

-- Number of threads
select count(*) from v$thread;

--------------
-- Resources
--------------

-- Space info
select
   fs.tablespace_name                          "Tablespace",
   (df.totalspace - fs.freespace)              "Used MB",
   fs.freespace                                "Free MB",
   df.totalspace                               "Total MB",
   round(100 * (fs.freespace / df.totalspace)) "Pct. Free"
from
   (select
      tablespace_name,
      round(sum(bytes) / 1048576) TotalSpace
   from
      dba_data_files
   group by
      tablespace_name
   ) df,
   (select
      tablespace_name,
      round(sum(bytes) / 1048576) FreeSpace
   from
      dba_free_space
   group by
      tablespace_name
   ) fs
where
   df.tablespace_name = fs.tablespace_name;

-- Number of tablespaces

SELECT count(DISTINCT sgm.TABLESPACE_NAME) "Number of Tablespaces"
FROM DBA_SEGMENTS sgm

-- Used Ram and Max used ram in MB

SELECT
```

```
sum((se1.value/1024)/1024) "CURRENT SIZE",
sum((se2.value/1024)/1024) "MAXIMUM SIZE"
FROM v$sesstat se1, v$sesstat se2, v$session ssn, v$bgprocess bgp,
   v$process prc,
v$instance ins, v$statname stat1, v$statname stat2
WHERE se1.statistic# = stat1.statistic# and stat1.name = 'session pga
   memory'
AND se2.statistic# = stat2.statistic# and stat2.name = 'session pga
   memory max'
AND se1.sid = ssn.sid
AND se2.sid = ssn.sid
AND ssn.paddr = bgp.paddr (+)
AND ssn.paddr = prc.addr (+);

-- Number of processes

select count(*) from v$process
where execution_type = 'PROCESS'

--------------
-- Activity
--------------

-- Number of sessions
select count(*) from V$session where status='ACTIVE';

-- History of sql requests
select v.SQL_TEXT,
           v.FIRST_LOAD_TIME
      from v$sql v
where to_date(v.FIRST_LOAD_TIME,'YYYY-MM-DD hh24:mi:ss')>sysdate-1

--------------
-- USERS
--------------

-- All users on database
SELECT USERNAME, USER_ID, DEFAULT_TABLESPACE,
   TEMPORARY_TABLESPACE,ACCOUNT_STATUS, CREATED FROM DBA_USERS
ORDER BY created ASC;

-- All roles on database
SELECT ROLE, ROLE_ID FROM DBA_ROLES;
```