

Este guia ajuda a configurar um espaço de trabalho para completar as aulas práticas, que inclui um sistema de gestão de bases de dados (SGBD) e outros serviços auxiliares em *containers* para o Curso de Sistemas de Informação e Bases de Dados no Instituto Superior Técnico.

- PostgreSQL um sistema de gestão de base de dados relacional de código aberto.
- pgAdmin4 uma plataforma de administração e desenvolvimento para o PostgreSQL.
- bdist/notebook Jupyter Notebook Databases Stack

1 Pré-requisitos

1.1 Instalar Docker Desktop

Docker Desktop é uma aplicação para máquinas MacOS, Linux e Windows para a construção e partilha de aplicações e microsserviços em containers. Instalar Docker Desktop em Mac, Windows or Linux

1.2 Instalar Git

Git é um sistema de controlo de versões distribuído de código aberto projetado para lidar com projetos pequenos e muito grandes com velocidade e eficiência. Instalar Git em Mac Windows or Linux

2 A primeira vez que você iniciar o Workspace

1. Abra o Terminal.
2. Altere o diretório de trabalho atual para o local onde você deseja clonar o diretório.
3. Execute o seguinte comando para criar o clone local:

```
git clone https://github.com/bdist/bdist-workspace.git
```

4. Altere o diretório de trabalho atual para a localização do diretório clonado.

```
cd bdist-workspace/
```

5. A partir do diretório clonado, inicie o `bdist-workspace` executando

```
docker compose up
```

Nota: Os serviços estão anexados a este Terminal, portanto, os logs de todos os serviços fornecidos serão impressos na janela do Terminal. Para encerrar de forma segura, você precisa pressionar CTRL+C na janela do Terminal e aguardar até que todos os serviços parem corretamente.

2.1 Usando o Jupyter Notebook (Docs)

O serviço Jupyter Notebook é executado na porta 9999. A autenticação por token está ligada:

`http://127.0.0.1:9999/lab?token=SEU_TOKEN_DE_AUTENTICACAO`

1. Você precisa encontrar o seu Token de Autenticação para fazer login sempre que o `bdist-workspace` for iniciado (por exemplo, após reiniciar).
2. Encontre a seção dos logs que está no final da janela do Terminal:

```
bdist-workspace-notebook-1 | Or copy and paste one of these URLs:
bdist-workspace-notebook-1 |
↪ http://7fd8c38e99bd:9999/lab?token=SEU_TOKEN_DE_AUTENTICACAO
bdist-workspace-notebook-1 |
↪ http://127.0.0.1:9999/lab?token=SEU_TOKEN_DE_AUTENTICACAO
```

Nota: Você também pode visualizar os logs do `bdist-workspace-notebook-1` na guia Containers na aplicação Docker Desktop.

3. Siga o link impresso por último com o host `127.0.0.1`.

3 Serviços incluídos

3.1 pgAdmin

pgAdmin é a plataforma de administração e desenvolvimento de código aberto mais popular e rica em recursos para o PostgreSQL, o sistema de gestão de bases de dados utilizado.

1. Login here.

Nome de utilizador: `pgadmin@tecnico.pt` Senha: `pgadmin`

2. Clique no botão Adicionar Novo Servidor.
3. Defina o Nome na guia Geral como `postgres`.
4. Defina o Nome do Host na guia de Conexão como `postgres`.
5. Utilize o mesmo nome de utilizador e senha que você forneceria para o `psql`.

Nome de utilizador: `postgres` Senha: `postgres`

3.2 Flask Web App

1. Execute o seguinte comando para criar o clone local:

```
git clone https://github.com/bdist/app.git
```

2. Certifique-se de que o diretório `app/` esteja ao lado do diretório `bdist-workspace/`. Se necessário, mova o diretório `app/` para fora do diretório `bdist-workspace/`.

```
$ ls
app/  bdist-workspace/
```

3. O arquivo `docker-compose..app.yml` contém as instruções para iniciar o workspace. Relance o workspace com o seguinte comando para usá-lo:

```
docker compose -f docker-compose..app.yml up
```

4. Verifique se o aplicativo está em execução e abra ping.

```
http://127.0.0.1:8080/ping
```

5. Você recebe uma resposta HTTP no formato JSON semelhante a uma API, como esta?

```
{
  "message": "pong!"
}
```

6. Tente modificar a mensagem em `../app/app.py` enquanto ele estiver em execução.
7. Nos logs, verifique se é acionado um *reload* quando você guarda as alterações.

4 Utilizando o cliente de linha de comandos

Regra geral, todos os sistemas de gestão de base de dados dispõem de uma interface de linha de comando através da qual é possível executar instruções SQL e também outros comandos de administração e manutenção do sistema.

Nota: Todos os comandos de administração começam com `\` (contra-barra ou barra invertida).

Nota: Todos os comandos terminam com `;` (ponto e vírgula) e começam com uma instrução SQL

CREATE, **SELECT**, **UPDATE**, **INSERT**, **DELETE**, etc.

1. Clique no botão **New Launcher** no canto superior esquerdo anotado com um sinal `+`.
2. Na página seguinte, selecione a opção **Terminal** na secção *Other*.

3. Ligue-se ao PostgreSQL usando o cliente de linha de comandos `psql`.

```
psql -h postgres -U postgres
```

4. Introduza a password do utilizador `postgres`.

```
postgres ↵
```

Use o comando `\h` ↵ para obter informação sobre os comandos SQL disponíveis.

Use o comando `\?` ↵ para obter informação sobre os comandos de administração disponíveis.

5 Criação da base de dados exemplo – ‘Bank’

Para criar uma base de dados, precisa de criar as tabelas e carregar os dados em cada tabela. A criação de uma base de dados é feita com a instrução `CREATE DATABASE`. A base de dados `bank` pode ser criada usando as instruções seguintes:

1. Crie um utilizador `bank` sem privilégios.

```
CREATE USER bank WITH PASSWORD 'bank';
```

2. Crie a base de dados `bank` e defina o utilizador `bank` como dono da base de dados.

```
CREATE DATABASE bank  
WITH  
OWNER = bank  
ENCODING = 'UTF8';
```

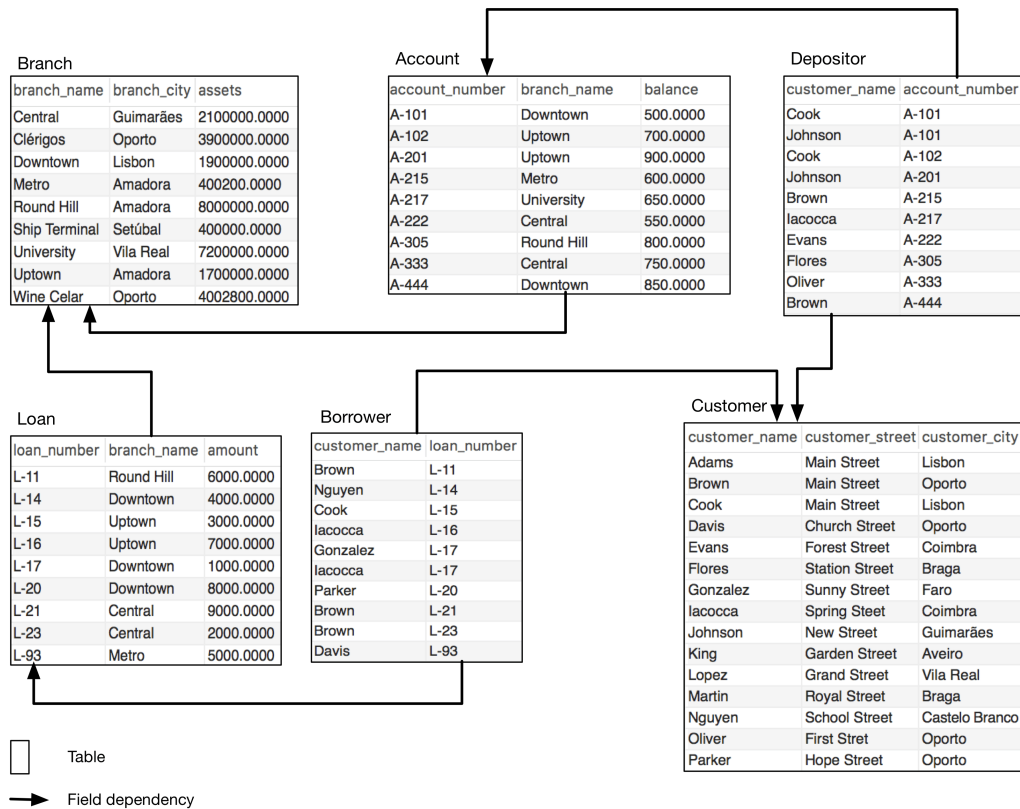
Note: Defina o *encoding* de caracteres para UTF-8 explicitamente.

3. Conceda ao utilizador `bank` todos os privilégios para a base de dados `bank`.

```
GRANT ALL ON DATABASE bank TO bank;
```

4. Exit the program using the command `\q` ↵.

Table Diagram for the BANK Example



NOTE: Values are only illustrative. The actual database can be different.

Figure 1: Figure 1

1. Ligue-se ao PostgreSQL usando o cliente de linha de comandos `psql`.

```
psql -h postgres -U bank
```

2. Introduza a password do utilizador bank.

```
bank ↵
```

O ficheiro `bank.sql` contém um conjunto de instruções SQL para criar a base de dados de exemplo ilustrada na Figura 1.

A criação de tabelas é feita usando a instrução `CREATE TABLE`. Por exemplo, a tabela `customer` pode ser criada usando a instrução seguinte:

```
CREATE TABLE customer
(customer_name varchar(80) not null unique,
 customer_street varchar(255) not null,
 customer_city varchar(30) not null,
 CONSTRAINT pk_customer PRIMARY KEY(customer_name));
```

Esta instrução especifica o nome da tabela, os nomes das três colunas, o tipo de cada coluna. Também especifica restrições tais como os valores não poderem ser **NULL** e o facto da chave primária da tabela ser o nome do cliente.

Os registos de cada tabela são carregados através de instruções SQL **INSERT**. Por exemplo:

```
INSERT INTO customer VALUES ( 'Luis' , 'Rua do Cima' , 'Musgueira' );
```

Nesta instrução são especificados, respetivamente, os valores de cada coluna pela mesma ordem em que estes foram definidos aquando da criação da tabela. Esta instrução resulta na criação de um novo registo na tabela de clientes.


Note-se que o ficheiro bank.sql inclui instruções para inserir mais registos na base de dados do que aqueles que se encontram exemplificados na Figura 1. Estes registos serão usados para realizar vários testes sobre a base de dados. Em aulas de laboratório futuras, iremos utilizar esta base de dados para demonstrar vários dos conceitos da disciplina.

Nota: Muito embora também possa ser utilizada a interface gráfica pgAdmin, como forma de interagir com o sistema de gestão de bases de dados Postgres, nas aulas de laboratório iremos sobretudo utilizar a interface de linha de comandos.

Para criar a base de dados de exemplo do Bank, execute o comando que irá carregar e executar as instruções SQL que estão no ficheiro bank.sql. O Postgres irá produzir algumas mensagens à medida que executa as instruções do ficheiro.

```
\i ~/data/bank.sql
```

No final da execução, a base de dados de exemplo está criada.

Para listar as tabelas da base de dados, use o comando: `\d` 

O tempo que o sistema demora a executar algumas consultas é um fator importante quando o volume de dados é considerável.

Execute o comando:

```
\timing 
```

para ativar ou desativar a cronometragem da execução dos comandos SQL.

Uma vez na sessão Postgres, poderá executar algumas consultas introduzindo comandos SQL, nomeadamente:

- ver a lista completa de clientes:

```
SELECT * FROM customer;
```

- ver a lista completa de contas:

```
SELECT * FROM account;
```

- ver o saldo da conta A-101:

```
SELECT balance FROM account  
WHERE account_number='A-101';
```

- ver todos os clientes que não são depositantes (i.e. não têm contas):

```
SELECT * FROM customer  
WHERE customer_name NOT IN (  
SELECT customer_name FROM depositor);
```

Nota: Em aulas futuras irá aprender a obter a resposta para consultas mais complexas.