Guião Laboratorial - Balanceamento de carga em Servidores Web com HAProxy e Keepalived

Disponibilidade e Desempenho

2021 - 2022

Bruno Teixeira
a2019100036@isec.pt

Licenciatura de Engenharia Informática
21 de janeiro de 2021

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O objetivo deste guião é fazer com que a pessoa que o seguir seja capaz de criar todas as experiências feitas de modo a perceber melhor os vários assuntos abordados no relatório.

1.1 Ambiente para as experiências

Para serem feitas algumas experiências foi criado um ambiente com várias máquinas virtuais, estando estas agregadas a um virtualizador ESXi, ou seja, todas as máquinas estão na mesma LAN.

- **webserver01** - 192.168.1.180
- **webserver02** - 192.168.1.181
- **mariadb01** - 192.168.1.182
- **mariadb02** - 192.168.1.186
- **haproxy01** - 192.168.1.183
- **haproxy02** - 192.168.1.184
- **haproxy03** - 192.168.1.185
- **haproxy04** - 192.168.1.187

1.1.1 Aplicação Web

Como descrito anteriormente, foi criada uma aplicação em Flask. Esta aplicação funciona como uma espécie de lista de compras em que o utilizador depois de fazer o *login*, consegue adicionar e eliminar produtos do seu cesto.

Figura 1.1: Index - Aplicação Web

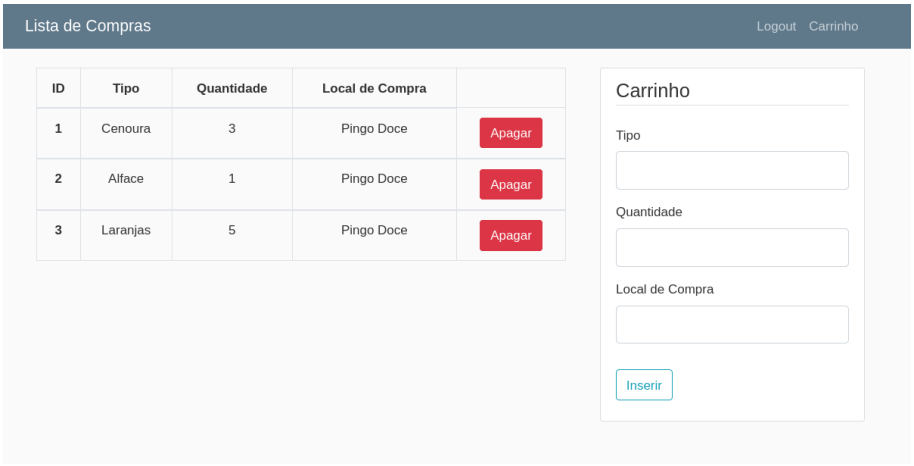

Figura 1.2: Carrinho - Aplicação Web

1.1.2 Configuração do HAProxy

No ficheiro de configuração do HAProxy (localizado em ***/etc/haproxy/haproxy.cfg***) existem 5 secções, sendo que estas definem como é que o servidor se comporta, quais são as definições por omissão, e como é que o cliente faz pedidos e recebe respostas.

- **global**
  - Nesta primeira secção estão definidas as medidas em que o processo vai operar, sendo estas medidas de um nível mais baixo, ou seja, relacionadas com o sistema operativo.
- **defaults**
  - Esta secção não é obrigatória, no entanto permite reduzir a duplicação de comandos, uma vez que as configurações feitas aqui são aplicadas na secção ***frontend*** e ***backend***.
- **listen**

- Aqui podemos combinar o **frontend** e **backend** ao mesmo tempo. Isto é útil, pois é aqui feito o redirecionamento para o ***endpoint*** de estatisticas.

- **frontend**
  - Nesta secção definimos como é que os pedidos dos utilizadores irão ser encaminhados para o **backend**.

- **backend**
  - Aqui definimos os ***webservers*** que vão operar na infraestrutura, definindo tambem o algoritmo de ***load balancing*** a ser utilizado

```
# O comando "stats socket" ativa a API do HAProxy sendo assim possível gerar um "endpoint" com todas as estatisticas do proxy e dos servidores web
global
        stats socket /run/haproxy/admin.sock mode 660 level admin

# Neste caso usamos o modo "http"
defaults
        mode http

# Configuração referente ao "endpoint" de estatisticas
# Por motivos de simplicidade o "auth" está sem  encriptação
listen stats
       bind 192.168.1.183:9999
       stats enable
       stats hide-version
       stats refresh 10s
       stats uri /stats
       stats auth haproxy:haproxy

# "http-in" é apenas um nome para o frontend
# Qualquer pessoa que se conecte a este servidor irá ser redirecionado para os "webservers" e fazemos isso usando "default_backend <nome do backend>"
frontend http-in
         bind 192.168.1.183:80
         default_backend webservers

# Damos o mesmo nome ao backend que demos no "default_backend"
# No "balance" escolhemos o algoritmo que quisermos utilizador na atribuição de "webservers" aos clientes (round-robin ou leastcoon)
# Fazemos um "check" com um intervalo de 500 milisegundos, se falhar 3 vezes retira esse "webserver" do grupo
backend webservers
        balance roundrobin
        server webserver01 192.168.1.180:5000 check inter 500 fall 3
        server webserver02 192.168.1.181:5000 check inter 500 fall 3
```

Figura 1.3: Configuração - HAProxy

Conforme foi configurado o HAProxy, ao acedermos a ***http://192.168.1.183:9999/stats*** conseguimos visualizar uma página *web* com várias estatísticas sobre os *webservers*.

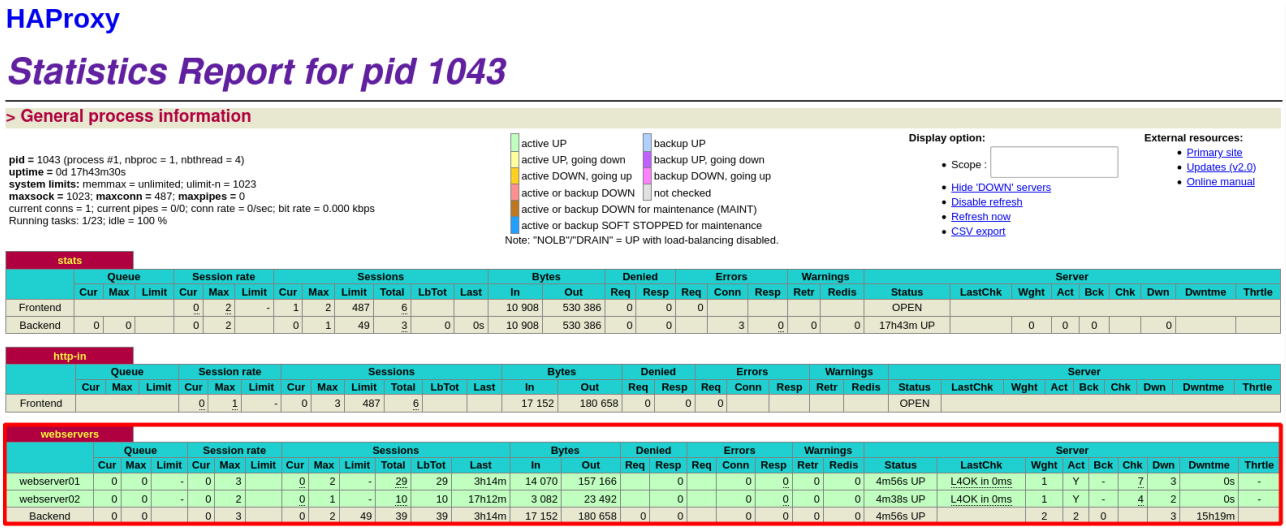

Figura 1.4: Estatísticas - HAProxy

1.1.3 Configuração do KeepAlived

No ficheiro de configuração do Keepalived (localizado em */etc/keepalived/keepalived.conf*), foi criado um **vrrp_script** com o intuito de verificar, com um intervalo de 2 em 2 milisegundos, se o haproxy está a funcionar corretamente. Se o mesmo não estiver a funcionar, o peso dele diminui em 10 reduzindo assim a sua prioridade tornando o BACKUP num MASTER.
Depois disto, foi criado um **vrrp_instance** que define uma instância individual do protocolo VRRP com alguns atributos.

```
# Define definições globais do processo
global_defs{
        enable_script_security
        script_user root
}

# Script que verifica o estado do HAProxy
vrrp_script check_haproxy {
            script "service haproxy status"
            interval 1
            weight -10
}

# Instancia que executa o script, nomeando o HAProxy01 (192.168.1.183) como MASTER
# Escolhemos a interface que queremos usar
# O estado inicial
# Identificação do virtual_router_id sendo que esta tem de ser a mesma no BACKUP
# Prioridade do servidor, esta será anunciada no grupo VRRP
# IP virtual do grupo VRRP
vrrp_instance V1_1{
              interface ens160
              state MASTER
              virtual_router_id 11
              priority 101
              authentication {
                auth_type PASS
                auth_pass algumacoisamuitocomplicada
              }
              virtual_ipaddress {
                192.168.1.200
              }
              track_script {
                check_haproxy
              }
}
```

Figura 1.5: Configuração - Keepalived

O mesmo foi feito para o segundo servidor de HAProxy, no entanto foi alterada a prioridade e o estado para definir que este seria o BACKUP.

Foi tambem necessário alterar o Ip, para onde faziamos ***bind*** inicialmente (**Ip do servidor HAProxy**), para o novo IP virtual (**192.168.1.200**) na secção de ***frontend*** do ficheiro de configuração do HAProxy.

Por fim foi preciso colocar ***net.ipv4.ip_nonlocal_bind=1*** no ficheiro */etc/sysctl.conf* uma vez que no segundo servidor de HAProxy o IP virtual ainda não está ativo (só fica ativo quando esse for o MASTER), logo não é possível iniciar o *bind*.

Todo este processo apenas foi feito a partir da experiência 02.

Capítulo 2

Guião

Em baixo estão descritos os passos para que seja possível criar todo o código da aplicação *web* assim como a base de dados de modo a conseguir-se replicar as experiências que foram mostradas no relatório do trabalho prático.
Estão tambem descritas as pequenas alterações feitas nos servidores ao longo das experiências.

2.1 Guião - Experiência 01

2.1.1 HAProxy

Para instalar o HAProxy, bastou fazer ***sudo apt install haproxy*** em ambos os servidores de HAProxy. Depois foi feita a configuração do ficheiro do HAProxy (***/etc/haproxy/haproxy.cfg***) como descrito na secção de **Configuração do HAProxy**.

2.1.2 Aplicação Web

Estrutura da aplicação

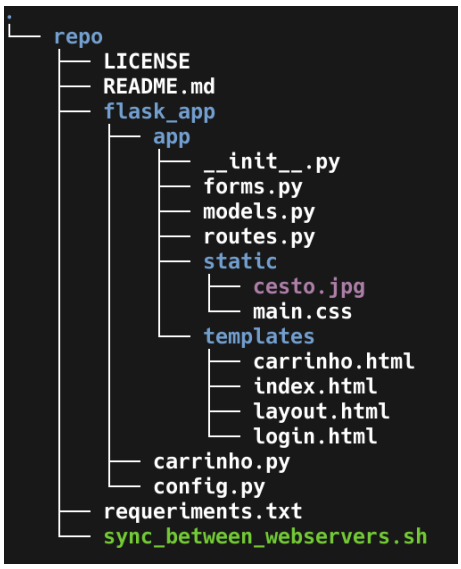

Figura 2.1: Estrutura da aplicação web - Guião

Variáveis de ambiente usadas

```
brun0@webserver01:~$ cat /etc/environment
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"

FLASK_APP=carrinho.py
FLASK_DEBUG=true
RSYNC_PASSWORD="toor"
DB_HOST="mysql+pymysql://brun0:toor@192.168.1.185:3306/carrinho"
DB_USER="brun0"
DB_PW="toor"
DB_NAME="carrinho"
```

Figura 2.2: Variáveis de ambiente - Guião

requirements.txt

Para esta aplicação Web são precisas algumas dependências, dependências estas descritas no ficheiro *requeriments.txt*. A maneira mais simples de instalar todas as dependências é criar um ficheiro chamado *requeriments.txt* e depois executar o comando *pip3 install -r requirements.txt*

```
email-validator==1.1.3
entrypoints==0.3
Flask==1.1.1
Flask-Login==0.5.0
Flask-SQLAlchemy==2.5.1
Flask-WTF==0.15.1
httplib2==0.14.0
importlib-metadata==1.5.0
incremental==16.10.1
Jinja2==2.10.1
mariadb==1.0.8
MarkupSafe==1.1.0
more-itertools==4.2.0
oauthlib==3.1.0
pexpect==4.6.0
pyasn1==0.4.2
pyasn1-modules==0.2.1
PyGObject==3.36.0
PyHamcrest==1.9.0
pyinotify==0.9.6
PyJWT==1.7.1
pymacaroons==0.13.0
PyMySQL==1.0.2
PyNaCl==1.3.0
pyOpenSSL==19.0.0
pyrsistent==0.15.5
pyserial==3.4
python-apt==2.0.0+ubuntu0.20.4.6
python-debian===0.1.36ubuntu1
PyYAML==5.3.1
requests==2.22.0
requests-unixsocket==0.2.0
SecretStorage==2.3.1
simplejson==3.16.0
SQLAlchemy==1.4.26
ssh-import-id==5.10
systemd-python==234
Twisted==18.9.0
ufw==0.36
urllib3==1.25.8
```

```
wadllib==1.3.3
Werkzeug==0.16.1
WTForms==2.3.3
```

sync_between_webservers.sh

Este script foi útil para que quando fosse feito um *commit* no *github*, o *webserver01* sincronizasse o código atualizado com o *webserver02*

```
#!/usr/bin/env bash

# automating git stuff

echo "Whats the commit message?"
read message
git add .
git commit -m "${message}"
echo "Pushing data ... "
git push


echo "Syncing to webserver02 ..."
# sync a folder from webserver01 to webserver02
rsync -rt /home/brun0/repo/ brun0@192.168.1.181:/home/brun0/repo --delete-after
```

flask_app

carrinho.py

```
from app import app
```

config.py

Criar variável de ambiente com o nome **DB_HOST**

```
DB_HOST=mysql+pymysql://<username>:<password>@<db_ip>:<db_port>/<db_name>
```

```
import os

basedir = os.path.abspath(os.path.dirname(__file__))

class Config(object):
   SECRET_KEY = "something"
   SQLALCHEMY_DATABASE_URI = os.environ.get('DB_HOST')
   SQLALCHEMY_TRACK_MODIFICATIONS = False
```

app

__init__.py

```
from flask import Flask
from config import Config
from flask_sqlalchemy import SQLAlchemy
```

```
from flask_login import LoginManager


app = Flask(__name__)
app.config.from_object(Config)
db = SQLAlchemy(app)

login = LoginManager(app)
login.login_view = "login"

from app import routes,models
```

forms.py

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, BooleanField, IntegerField
from wtforms.validators import DataRequired, Length, Email, EqualTo


class LoginForm(FlaskForm):
    email = StringField('Email',validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    submit = SubmitField('Login')

class ProductForm(FlaskForm):
       product_type = StringField("Tipo", validators=[DataRequired()])
       quantity = IntegerField("Quantidade", validators=[DataRequired()])
       local = StringField("Local de Compra", validators=[DataRequired()])
       submit = SubmitField('Inserir')
```

models.py

```
from app import db, login
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import UserMixin

@login.user_loader
def load_user(id):
    return Clientes.query.get(int(id))

class Clientes(UserMixin, db.Model):
       id_cliente = db.Column(db.Integer, primary_key=True)
       nome = db.Column(db.String(64), index=True, unique=True)
       email = db.Column(db.String(120), index=True, unique=True)
       password = db.Column(db.String(128))
       compras = db.relationship("Compras", backref="cliente")

       def __repr__(self):
              return f"{self.nome}"

       def set_password(self, password):
              self.password = generate_password_hash(password)

       def check_password(self, password):
              return check_password_hash(self.password,password)
```

```
        def get_id(self):
                return self.id_cliente

class Compras(db.Model):
        id_compras = (db.Column(db.Integer, primary_key=True))
        tipo = db.Column(db.String(140))
        quantidade = db.Column(db.Integer)
        local = db.Column(db.String(140))
        id_cliente = db.Column(db.Integer, db.ForeignKey("clientes.id_cliente"))

        def __repr__(self):
                return f"{self.id_compras}"
```

routes.py

```
from flask import render_template, flash, redirect, url_for, request, jsonify, make_response
from app import app,db
from app.forms import LoginForm, ProductForm
from flask_login import current_user, login_user, logout_user, login_required
from app.models import Clientes, Compras


# index
@app.route("/")
def index():
    return render_template("index.html")

# login
@app.route('/login', methods=['GET', 'POST'])
def login():
        if current_user.is_authenticated:
                return redirect(url_for("carrinho"))

        form = LoginForm()
        if form.validate_on_submit():
                # query a bd
                user = Clientes.query.filter_by(email=form.email.data).first()
                # verifica o resultado da query, se nao existir ...
                if user is None or not user.check_password(form.password.data):
                        flash("Login invalido","danger")
                        return redirect(url_for("login"))
                # se existir faz login
                else:
                        login_user(user)
                return redirect(url_for("carrinho"))
        return render_template("login.html", title="Login", form=form)

# logout
@app.route("/logout")
def logout():
        logout_user()
        flash("Logout com sucesso", "info")
        return redirect(url_for("index"))


# carrinho
```

```python
@app.route("/carrinho/", methods=["GET", "POST"])
@login_required
def carrinho():
        # lista produtos atuais do cliente
        customer_list = db.session.query(Compras.id_compras.label("id_compras"),
                                         Compras.tipo.label("Tipo"),
                                         Compras.quantidade.label("Quantidade"),
                                         Compras.local.label("Local"))\
                                         .join(Clientes, Compras.id_cliente == Clientes.id_cliente)\
                                         .filter(Compras.id_cliente==current_user.get_id()).all()

    # adicionar produtos ao carrinho
        form = ProductForm()
        if form.validate_on_submit():
                compras = Compras(tipo = form.product_type.data,
                                  quantidade = form.quantity.data,
                                  local = form.local.data,
                                  id_cliente = current_user.get_id())
                db.session.add(compras)
                db.session.commit()

                return redirect(url_for("carrinho"))

        return render_template("carrinho.html", title="Lista", form=form,
            customer_list=customer_list)

# apagar artigo
@app.route("/apagar_artigo",methods=["POST"])
@login_required
def delete_item():
        # recebe o POST feito no js
        req = request.get_json()
        # aplica a query
        Compras.query.filter_by(id_compras=int(req["id"])).delete()
        db.session.commit()

        return redirect(url_for("carrinho"))
```

---

static

main.css

```css
  body {
  background: #fafafa;
  color: #333333;
  margin-top: 5rem;
}

h1, h2, h3, h4, h5, h6 {
  color: #444444;
}

.bg-steel {
  background-color: #5f788a;
}

.site-header .navbar-nav .nav-link {
```

```
  color: #cbd5db;
}

.site-header .navbar-nav .nav-link:hover {
  color: #ffffff;
}

.site-header .navbar-nav .nav-link.active {
  font-weight: 500;
}

.content-section {
  background: #ffffff;
  padding: 10px 20px;
  border: 1px solid #dddddd;
  border-radius: 3px;
  margin-bottom: 20px;
}

.article-title {
  color: #444444;
}

a.article-title:hover {
  color: #428bca;
  text-decoration: none;
}

.article-content {
  white-space: pre-line;
}

.article-img {
  height: 65px;
  width: 65px;
  margin-right: 16px;
}

.article-metadata {
  padding-bottom: 1px;
  margin-bottom: 4px;
  border-bottom: 1px solid #e3e3e3
}

.article-metadata a:hover {
  color: #333;
  text-decoration: none;
}

.article-svg {
  width: 25px;
  height: 25px;
  vertical-align: middle;
}
```

templates

layout.html

```
<!DOCTYPE html>
<html>
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="https://use.fontawesome.com/releases/v5.6.3/css/all.css" rel="stylesheet">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
        integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
        crossorigin="anonymous">

    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='main.css') }}">
    <title>Lista de compras 01</title>
</head>
<body>
    <header class="site-header">
      <nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">
        <div class="container">
          <a class="navbar-brand mr-4" href="/">Lista de Compras</a>
          <button class="navbar-toggler" type="button" data-toggle="collapse"
              data-target="#navbarToggle" aria-controls="navbarToggle" aria-expanded="false"
              aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarToggle">
            <div class="navbar-nav mr-auto">
            </div>
            <!-- Navbar Right Side -->
            <div class="navbar-nav">
              {% if current_user.is_anonymous %}
              <a class="nav-item nav-link" href="{{ url_for('login') }}">Login</a>
              {% else %}
              <a class="nav-item nav-link" href="{{ url_for('logout') }}">Logout</a>
              <a class="nav-item nav-link" href="{{ url_for('carrinho') }}">Carrinho</a>
              {% endif %}
            </div>
          </div>
        </div>
      </nav>
    </header>
    <main role="main" class="container">
      <div class="row">

          {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
              {% for category, message in messages %}
                <div class="alert alert-{{ category }}" style="margin:auto">
                  {{ message }}
                </div>
              {% endfor %}
            {% endif %}
        {% endwith %}
```

```
      {% block content %}{% endblock %}
    </div>
  </main>


  <!-- Optional JavaScript -->
  <!-- jQuery first, then Popper.js, then Bootstrap JS -->
  <script src="https://code.jquery.com/jquery-3.1.1.min.js"
      integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
      crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
      integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
      crossorigin="anonymous"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
      integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
      crossorigin="anonymous"></script>
</body>
</html>
```

index.html

```
{% extends "layout.html" %}
{% block content %}

<div class="container ">
<div class="d-flex justify-content-center p-5">
  <img class="mt-5" src="static/cesto.jpg" alt="icon" style="width:350px">
  </div>
  <div class="d-flex justify-content-center text-center p-5">
    <h1>A sua lista de compras<h1/>
    </div>
</div>
{% endblock content %}
```

carrinho.html

```
{% extends "layout.html" %}
{% block content %}

<script>
  function remove_entry(){
      var entry = {id:event.srcElement.id};

      fetch(`${window.origin}/apagar_artigo`,{
          method: "POST",
          credentials: "include",
          body: JSON.stringify(entry),
          cache: "no-cache",
          headers: new Headers({
              "content-type":"application/json"
          })
      })

  }
</script>
```

```
{% if customer_list|length > 0 %}
<div class="col-8">
<div class="container">
     <table class="table table-bordered text-center" id="mytable">
       <thead>
         <tr>
           <th scope="col">ID</th>
           <th scope="col">Tipo</th>
           <th scope="col">Quantidade</th>
           <th scope="col">Local de Compra</th>
           {% for item in customer_list %}
         </tr>
       </thead>
       <tbody>
         <tr>
           <th scope="row">{{ loop.index }}</th>
           <td>{{ item[1] }}</td>
           <td>{{ item[2] }}</td>
           <td>{{ item[3] }}</td>
           <td>
             <form>
                 <button id={{ item[0] }} type="submit" class="btn btn-danger"
                     onclick="remove_entry();">Apagar</button>
               </form>
           </td>
           {% endfor %}
           </tr>
       </tbody>
     </table>
</div>
</div>
{% else %}

<div class="d-flex justify-content-center">
<div class=" d-flex alert alert-dark text-center" style="max-height:62px">Neste momento nao tem
    nenhum produto no seu cesto
</div>
</div>
{% endif %}


  <div class="col-4">
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Carrinho</legend>
                <div class="form-group">
                    {{ form.product_type.label(class="form-control-label") }}
                    {% if form.product_type.errors %}
                        {{ form.product_type(class="form-control form-control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.product_type.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.product_type(class="form-control form-control-lg") }}
```

```
                {% endif %}
            </div>
            <div class="form-group">
                {{ form.quantity.label(class="form-control-label") }}
                {% if form.quantity.errors %}
                    {{ form.quantity(class="form-control form-control-lg is-invalid") }}
                    <div class="invalid-feedback">
                        {% for error in form.quantity.errors %}
                            <span>{{ error }}</span>
                        {% endfor %}
                    </div>
                {% else %}
                    {{ form.quantity(class="form-control form-control-lg") }}
                {% endif %}
            </div>
            <div class="form-group">
                {{ form.local.label(class="form-control-label") }}
                {% if form.local.errors %}
                    {{ form.local(class="form-control form-control-lg is-invalid") }}
                    <div class="invalid-feedback">
                        {% for error in form.local.errors %}
                            <span>{{ error }}</span>
                        {% endfor %}
                    </div>
                {% else %}
                    {{ form.local(class="form-control form-control-lg") }}
                {% endif %}
            </div>
        </fieldset>
        <div class="form-group">
          {{ form.submit(class="btn btn-outline-info") }}
        </div>
    </form>
  </div>
  </div>

{% endblock content %}
```

login.html

```
{% extends "layout.html" %}
{% block content %}
  <div class="container">
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Log In</legend>
                <div class="form-group">
                    {{ form.email.label(class="form-control-label") }}
                    {% if form.email.errors %}
                        {{ form.email(class="form-control form-control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.email.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
```

```
                {% else %}
                    {{ form.email(class="form-control form-control-lg") }}
                {% endif %}
            </div>
            <div class="form-group">
                {{ form.password.label(class="form-control-label") }}
                {% if form.password.errors %}
                    {{ form.password(class="form-control form-control-lg is-invalid") }}
                    <div class="invalid-feedback">
                        {% for error in form.password.errors %}
                            <span>{{ error }}</span>
                        {% endfor %}
                    </div>
                {% else %}
                    {{ form.password(class="form-control form-control-lg") }}
                {% endif %}
            </div>
        </fieldset>
        <div class="form-group">
          {{ form.submit(class="btn btn-outline-info") }}
        </div>
    </form>
</div>

{% endblock content %}
```

2.1.3 Base de Dados

Criação da base de dados

Como a conexão é feita a partir de dois servidores remotos(**haproxy01** e **haproxy02**), foi preciso garantir todos os privilegios a esses dois servidores

```
mysql -u root -p
mysql> GRANT ALL ON *.* to <username>@'<ip_haproxy01>' IDENTIFIED BY '<password>';
mysql> GRANT ALL ON *.* to <username>@'<ip_haproxy02>' IDENTIFIED BY '<password>';
mysql> FLUSH PRIVILEGES;
mysql> exit;
sudo service mariadb restart
mysql -u root -p
mysql> create database carrinho;
```

Criação das tabelas

```
create table clientes(
    id_cliente int auto_increment,
    nome varchar(255) not null,
    email varchar(255) not null,
    password varchar(255) not null,
    primary key(id_cliente)
);


create table compras(
    id_compras int auto_increment,
    id_cliente int,
```

```
    tipo varchar(255) not null,
    quantidade int not null,
    local varchar(255) not null,
    primary key(id_compras),
    foreign key(id_cliente)
    references clientes(id_cliente)
);
```

Inserção de dados nas tabelas

```
insert into clientes(nome, email,password) values ("Bruno", "a2019100036@isec.pt",
    "passwordsegura");
insert into clientes(nome, email,password) values ("Teixeira", "brunoalexandre3@hotmail.com",
    "passwordsegura");

insert into compras(id_cliente,tipo,quantidade,local) values (1, "Laranjas", 5,"PingoDoce");
insert into compras(id_cliente,tipo,quantidade,local) values (2, "Macas", 2,"PingoDoce");
insert into compras(id_cliente,tipo,quantidade,local) values (1, "Peras", 3,"PingoDoce");
```

Selecionar todos os artigos do carrinho do Cliente 1

```
select compras.id_compras "Id Compras", compras.id_cliente "Id Cliente", clientes.nome "Nome
    Cliente", compras.tipo, compras.quantidade, compras.local from compras inner join clientes on
    compras.id_cliente = clientes.id_cliente and clientes.id_cliente = 1;
```

2.2 Guião - Experiência 02

Nesta segunda experiência apenas foi acrescentado mais um servidor de balanceamento de carga e o serviço de KeepAlived em ambos os servidores de HAProxy.O objetivo da mesma era entender como seria feito o *fail-over* do KeepAlived e o que sucedia depois de um servidor *MASTER* tornar-se *BACKUP* e vice-versa.

Para instalar o KeepAlived, bastou fazer ***sudo apt install keepalived*** em ambos os servidores de HAProxy.
A sua configuração está tambem descrita na secção **Configuração do Keepalived**.

2.3 Guião - Experiência 03

Nesta terceira experiências foi preciso utilizar mais um servidor HAProxy(**192.168.185**) e mais uma base de dados(**192.168.1.186**). Esta base de dados agora vai-se juntar à base de dados (**192.168.1.182**) já existente fazendo assim um *Galera Cluster*.

No primeiro servidor de base de dados (**192.168.1.182**) foi criado o ficheiro **galera.cnf** localizado em ***/etc/mysql/mariadb.conf.d/galera.cnf*** contendo todas as configurações necessárias para a configuração do *cluster*.

```
[mysqld]
bind-address=0.0.0.0
default_storage_engine=InnoDB
binlog_format=row
innodb_autoinc_lock_mode=2
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

 ;; IP da mariadb01 e da mariabd02
wsrep_cluster_address="gcomm://192.168.1.182,192.168.1.186"
wsrep_cluster_name="mariadb-galera-cluster"
wsrep_sst_method=rsync

;; IP do servidor local
wsrep_node_address="192.168.1.182"
wsrep_node_name="galera-db-01"
```

Figura 2.3: Configuração do *Galera Cluster* no mariadb01 - Experiência 03

No segundo servidor de base de dados (**192.168.1.186**) foi tambem criado o ficheiro **galera.cnf** com as configurações necessárias ao mesmo fazer parte do *cluster*.

```
[mysqld]
bind-address=0.0.0.0
default_storage_engine=InnoDB
binlog_format=row
innodb_autoinc_lock_mode=2
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

 ;; IP da mariadb01 e da mariabd02
wsrep_cluster_address="gcomm://192.168.1.182,192.168.1.186"
wsrep_cluster_name="mariadb-galera-cluster"
wsrep_sst_method=rsync

;; IP do servidor local
wsrep_node_address="192.168.1.186"
wsrep_node_name="galera-db-01"
```

Figura 2.4: Configuração do *Galera Cluster* no mariadb02 - Experiência 03

Depois de ambos os servidores de base de dados configurados, foi visto que o tamanho do *cluster* era de 2, para isso foi utilizado o comando ***sudo mysql -u root -p -e "show status like 'wsrep_cluster_size'"***.

```
brun0@mariadb01:~$ sudo mysql -u root -p -e "show status like 'wsrep_cluster_size'"
[sudo] password for brun0:
Enter password:
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 2     |
+--------------------+-------+
```

Figura 2.5: Tamanho do *Galera Cluster* - Experiência 03

De seguida o novo servidor de HAProxy (**192.168.1.185**) foi configurado de modo a receber e encaminhar tráfego dos clientes e das base de dados, fazendo assim um balanceamento de carga entre as bases de dados existentes no *Galera Cluster*.
Para isto, foi editado o ficheiro **haproxy.cfg** localizado em ***/etc/haproxy/haproxy.cfg***.

```
frontend mariadb_cluster_frontend
    bind *:3306
    mode tcp
    option tcplog
    default_backend galera_cluster_backend

;; modo TCP
;; utilizando roundrobin entre o mariabd01 e mariadb02
backend galera_cluster_backend
    mode tcp
    option tcpka
    balance roundrobin
    server mariadb01 192.168.1.182:3306  check weight 1
    server mariadb02 192.168.1.186:3306  check weight 1
```

Figura 2.6: Configuração do HAProxy - Experiência 03

Por fim, foi necessário alterar o IP presente na variável de ambiente **DB_HOST**, em ambos os *webservers*, de **192.168.1.182** para o IP do HAProxy (**192.168.1.185**).

2.4 Guião - Experiência 04

Por fim, na experiência final foi preciso utilizar mais um servidor de HAProxy (**192.168.1.187**) e acrescentar o KeepAlived em ambos os servidores HAProxy que fazem o balanceamento de carga para o *Galera Cluster*.

```
brun0@haproxy03:~$ sudo cat /etc/haproxy/haproxy.cfg
frontend mariadb_cluster_frontend
    bind *:3306
    mode tcp
    option tcplog
    default_backend galera_cluster_backend

backend galera_cluster_backend
    mode tcp
    option tcpka
    balance roundrobin
    server mysql-01 192.168.1.182:3306  check weight 1
    server mysql-02 192.168.1.186:3306  check weight 1
```

```
brun0@haproxy04:~$ sudo cat /etc/haproxy/haproxy.cfg
frontend mariadb_cluster_frontend
    bind *:3306
    mode tcp
    option tcplog
    default_backend galera_cluster_backend

backend galera_cluster_backend
    mode tcp
    option tcpka
    balance roundrobin
    server mysql-01 192.168.1.182:3306  check weight 1
    server mysql-02 192.168.1.186:3306  check weight 1
```

Figura 2.7: Configuração do HAProxy - Experiência 04

A configuração do KeepAlived foi exatamente igual, no entanto para estes dois servidores de HAProxy foi utilizado o ip virtual **192.168.1.250**, tendo na mesma um ***vrrp_script*** que faz o controlo da "vida"dos servidores de HAProxy.

```
brun0@haproxy03:~$ sudo cat /etc/keepalived/keepalived.conf
global_defs{
        enable_script_security
        script_user root
}

vrrp_script check_haproxy {
           script "service haproxy status"
           interval 1
           weight -10
}

vrrp_instance V1_1{
             interface ens160
             state MASTER
             virtual_router_id 11
             priority 101
             authentication {
               auth_type PASS
               auth_pass algumacoisamuitocomplicada
             }
             virtual_ipaddress {
               192.168.1.250
             }
             track_script {
               check_haproxy
             }
}
```

```
brun0@haproxy04:~$ sudo cat /etc/keepalived/keepalived.conf
global_defs{
        enable_script_security
        script_user root
}

vrrp_script check_haproxy {
           script "service haproxy status"
           interval 2
           weight -10
}

vrrp_instance V1_1{
             interface ens160
             state BACKUP
             virtual_router_id 11
             priority 100
             authentication {
               auth_type PASS
               auth_pass algumacoisamuitocomplicada
             }
             virtual_ipaddress {
               192.168.1.250
             }
             track_script {
               check_haproxy
             }
}
```

Figura 2.8: Configuração do KeepAlived - Experiência 04

Por fim, foi necessário alterar o IP presente na variável de ambiente **DB_HOST**, em ambos os *webservers*, de **192.168.1.185** (IP do HAProxy03) para o IP virtual (**192.168.1.185**).

Figura 2.9: Variável de ambiente DB_HOST - Experiência 04