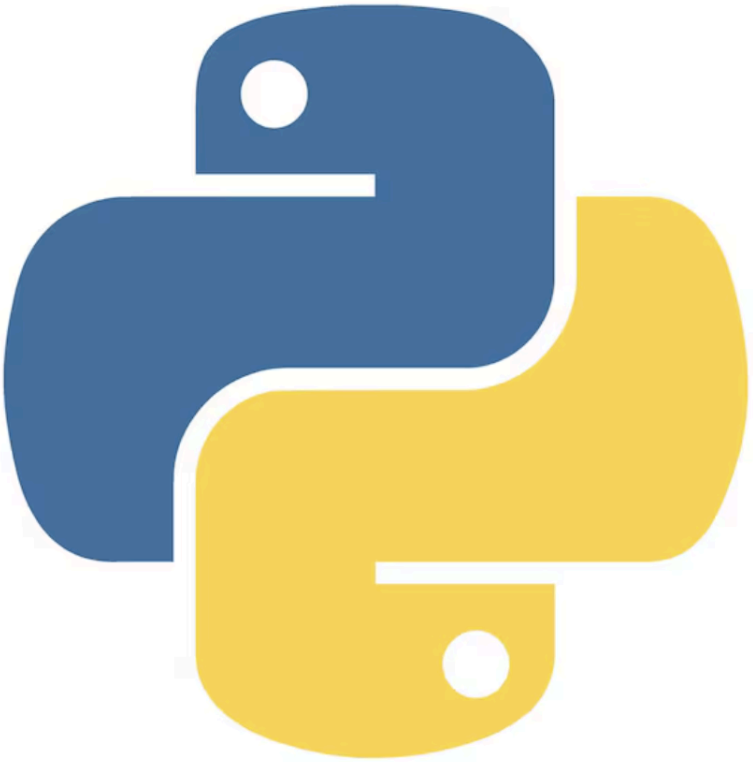
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL, I.P.
CENTRO DE EMPREGO E FORMAÇÃO PROFISSIONAL DE BRAGA

EFA T – PROGRAMADOR/A DE INFORMÁTICA

UFCD 10794

Python Avançado



Leonardo Souza, 06.

Trabalho sobre: Plataforma Web para uma Clínica Veterinária.

Formador: Marcos Alvarães

PESSOAS 2030 · PORTUGAL 2030 · Cofinanciado pela União Europeia
Os Fundos Europeus mais próximos de si.

Resumo do Projeto

Sistema de Clínica Veterinária

Este projeto consiste no desenvolvimento de um **sistema web para gestão de uma clínica veterinária**, com o objetivo de facilitar o controle de utilizadores, clientes, animais e consultas, centralizando todas as informações em uma única plataforma.

O sistema foi desenvolvido utilizando o **framework Flask (Python)**, seguindo o padrão **MVC (Model–View–Controller)**, onde:

- o **backend** é responsável pelo processamento das regras de negócio,
- o **frontend** apresenta as interfaces ao utilizador através de templates HTML,
- e o **banco de dados MySQL** armazena todas as informações do sistema.

A aplicação possui um sistema de **autenticação e controle de acesso**, permitindo que apenas utilizadores autenticados acessem áreas restritas. Os utilizadores são organizados por **perfis (admin, staff e cliente)**, onde cada perfil possui permissões específicas dentro do sistema.

Entre as principais funcionalidades do sistema destacam-se:

- Registo e autenticação de utilizadores
- Gestão de clientes
- Cadastro e listagem de animais associados a cada cliente
- Registo e consulta de consultas veterinárias

- Área exclusiva do cliente (“Minha Área”), onde é possível visualizar seus dados, seus animais e suas consultas
- Proteção de rotas utilizando sessões para garantir a segurança da aplicação

O projeto tem como foco **organização, segurança e usabilidade**, permitindo que a clínica tenha um controle eficiente das informações, reduzindo erros manuais e melhorando o atendimento aos clientes.

Documentação das Funções - Clínica Veterinária

Função: ligar_bd

Descrição: Realiza a conexão com o banco de dados MySQL.

Print da função:

```python
# ------------------------------------------------------
# FUNÇÃO PARA CONEXÃO COM MYSQL
# ------------------------------------------------------
def ligar_bd():
    """
    Cria e retorna a conexão com o servidor MySQL
    """
    return mysql.connector.connect(
        host="62.28.39.135",
        user="efa0125",
        password="123.Abc",
        database="efa0125_06_Leonardo_clinica_veterinaria"
    )
```

Função: executar_query

Descrição: Executa comandos SQL de forma segura.

Print da função:

```python
# ------------------------------------------------------
# HELPER DE EXECUÇÃO SEGURA (TRY / EXCEPT)
# ------------------------------------------------------
def executar_query(query, params=None, fetchone=False, fetchall=False, commit=False):
    cnx = None
    cursor = None
    try:
        cnx = ligar_bd()
        cursor = cnx.cursor(dictionary=True)
        cursor.execute(query, params or ())

        if fetchone:
            return cursor.fetchone()
        if fetchall:
            return cursor.fetchall()

        if commit:
            cnx.commit()

    except mysql.connector.Error as err:
        if cnx:
            cnx.rollback()
        print("ERRO BD:", err)
        raise err

    finally:
        if cursor:
            cursor.close()
        if cnx:
            cnx.close()
```

Função: coleta_user_role

Descrição: Disponibiliza a role do utilizador nos templates.

Print da função:

```python
# ------------------------------------------------------------
# CONTEXT PROCESSOR
# ------------------------------------------------------------
@app.context_processor
def coleta_user_role():
    """
    Disponibiliza a role do usuário logado para todos os templates
    """
    return dict(user_role=session.get("user_role"))
```

Função: base

Descrição: Página inicial do sistema.

Print da função:

```python
# ------------------------------------------------------------
# ROTA BASE
# ------------------------------------------------------------
@app.route("/")
def base():
    """
    Página inicial: redireciona para login se não estiver logado
    """
    if "user_id" not in session:
        return redirect(url_for("login"))
    return render_template("base.html")
```

Função: login

Descrição: Autenticação de utilizadores.

Print da função:

```python
# ------------------------------------------------------
# LOGIN
# ------------------------------------------------------
@app.route("/login", methods=["GET", "POST"])
def login():
    """
    Login de usuários
    """
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]

        try:
            user = executar_query(
                "SELECT id, username, password, role FROM users WHERE username=%s",
                (username,),
                fetchone=True
            )
        except:
            flash("Erro ao acessar a base de dados.")
            return redirect(url_for("login"))

        if user and user["password"] == password:
            session["user_id"] = user["id"]
            session["username"] = user["username"]
            session["user_role"] = user["role"]
            session["password"] = user["password"]
            return redirect(url_for("base"))

        flash("Username ou password incorretos.")
        return redirect(url_for("login"))

    return render_template("users/login.html")
```

Função: logout

Descrição: Finaliza a sessão do utilizador.

Print da função:

```python
# ------------------------------------------------------------
# LOGOUT
# ------------------------------------------------------------
@app.route("/logout")
def logout():
    """
    Limpa sessão e redireciona para login
    """
    session.clear()
    return redirect(url_for("login"))
```

Função: home

Descrição: Página principal após login.

Print da função:

```python
# ------------------------------------------------------------
# HOME
# ------------------------------------------------------------
@app.route("/home")
def home():
    if "user_id" not in session:
        flash("Faça login primeiro!")
        return redirect(url_for("login"))
    return render_template("global/home.html")
```

Função: minha_area

Descrição: Área do cliente com dados, animais e consultas.

Print da função:

```
# ------------------------------------------------------
# MINHA ÁREA (CLIENTE)
# ------------------------------------------------------
@app.route("/minha_area")
def minha_area():
    """
    Mostra área do cliente
    """
    if "user_id" not in session:
        flash("Faça login primeiro!")
        return redirect(url_for("login"))

    try:
        user = executar_query(
            "SELECT * FROM users WHERE id=%s",
            (session["user_id"],),
            fetchone=True
        )

        cliente = executar_query(
            "SELECT * FROM clientes WHERE id=%s",
            (user["cliente_id"],),
            fetchone=True
        )

        animais = executar_query(
            "SELECT * FROM animais WHERE cliente_id=%s",
            (cliente["id"],),
            fetchall=True
        )
```

```
        consultas = executar_query("""
            SELECT c.*, a.nome AS animal_nome
            FROM consultas c
            JOIN animais a ON c.animal_id = a.id
            WHERE a.cliente_id=%s
        """, (cliente["id"],), fetchall=True)

    except:
        flash("Erro ao carregar dados.")
        return redirect(url_for("login"))

    return render_template(
        "cliente/minha_area.html",
        user=user,
        cliente=cliente,
        animais=animais,
        consultas=consultas
    )
```

Função: tabela_utilizadores

Descrição: Lista todos os utilizadores.

Print da função:

```python
# ------------------------------------------------------------
# TABELAS
# ------------------------------------------------------------
@app.route("/tabela_utilizadores")
def tabela_utilizadores():
    if "user_id" not in session:
        return redirect(url_for("login"))

    try:
        utilizadores = executar_query(
            "SELECT id, username, role, created_at FROM users ORDER BY id ASC",
            fetchall=True
        )
    except:
        flash("Erro ao carregar utilizadores.")
        return redirect(url_for("base"))

    return render_template("admin/tabela_utilizadores.html", utilizadores=utilizadores)
```

Função: editar_users

Descrição: Edita dados de um utilizador.

Print da função:

```python
# ------------------------------------------------------
# EDITAR UTILIZADOR
# ------------------------------------------------------
@app.route("/editar_utilizador/<int:id>", methods=["GET","POST"])
def editar_users(id):
    """
    Edita dados de um usuário
    """
    if "user_id" not in session:
        return redirect(url_for("login"))

    if request.method == "POST":
        try:
            executar_query(
                "UPDATE users SET username=%s, role=%s WHERE id=%s",
                (request.form["username"], request.form["role"], id),
                commit=True
            )
        except:
            flash("Erro ao atualizar utilizador.")
            return redirect(url_for("tabela_utilizadores"))

        return redirect(url_for("tabela_utilizadores"))

    try:
        usuarios = executar_query(
            "SELECT id, username, role FROM users WHERE id=%s",
            (id,),
            fetchone=True
        )
    except:
        flash("Erro ao carregar utilizador.")
        return redirect(url_for("tabela_utilizadores"))

    return render_template("admin/editar_users.html", usuarios=usuarios)
```

Função: deleta_utilizador

Descrição: Remove um utilizador do sistema.

Print da função:

```python
# ------------------------------------------------------------
# DELETAR UTILIZADOR
# ------------------------------------------------------------
@app.route("/deleta_utilizador/<int:id>", methods=["POST"])
def deleta_utilizador(id):
    if "user_id" not in session:
        return redirect(url_for("login"))

    try:
        executar_query(
            "DELETE FROM users WHERE id=%s",
            (id,),
            commit=True
        )
    except:
        flash("Erro ao apagar utilizador.")
        return redirect(url_for("tabela_utilizadores"))

    flash("Utilizador apagado com sucesso.")
    return redirect(url_for("tabela_utilizadores"))

# ------------------------------------------------------------
```