RELATÓRIO

Programação e Integração de Serviços

IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Tecnologia e Programação de Sistemas de Informação

Carlos Carmo, 202000733
Ryan Silva, 202002433

Índice

Introdução

Este projeto desenvolvido pelos alunos de Tecnologia e Programação de Sistemas de Informação tem como objetivo de criar uma aplicação de adoção de animais de estimação utilizando as tecnologias de servidor para a gestão das mesmas. Utilizando o **NodeJs** como servidor aplicacional e o **MySQL** para persistência dos dados. Ou seja, que contenha as operações GET,POST,PUT,DELETE sobre um SGBD (Sistema de Gestão de Base de Dados) MySQL.

Este relatório foi dividido em duas partes:

Manual Técnico: no qual servirá para quem for instalar o produto e possui as seguintes secções:

- Breve descrição das tecnologias base.
- Lista de Tecnologias utilizadas e para que efeito; Ex: JavaScript jquery, MySQL,

Manual de Utilizador: no qual servirá para um utilizador do mesmo (nas várias perspetivas ou papéis) e contem as seguintes informações:

- Lista das várias funcionalidades. Ou seja, a gestão, a consulta , criação e eliminação dos usuários, animais e organizações
- Cada funcionalidade indica as suas características e toda a informação relevante.

Base de Dados

Foi utilizado o programa **MySQL** para a criação da base de dados para o projeto. Foram criadas as seguintes Tabelas:

- **Usuario**

Nome	Tipo de dado
id_usuario	Int (Primary Key)
nome	Varchar(50)
email	Varchar(50)
morada	Varchar(50)
idade	Tinyint
telefone	Varchar(15)
sexo	Varchar(10)
nome_usuario	Varchar(20)
senha	Varchar(25)

- **Animal**

Nome	Tipo de dado
id_animal	Int (Primary Key)
tipo	Varchar(15)
raca	Varchar(15)
cor	Varchar(15)
idade	Tinyint
sexo	Varchar(10)
localidade	Varchar(20)

- **Organização**

Nome	Tipo de dado
id_organizacao	Int (Primary Key)
nome_organizacao	Varchar(15)
localidade	Varchar(20)

Tecnologias Utilizadas

Para a realização deste projeto foram utilizadas as seguintes tecnologias:

- ✓ **HTML** - (Linguagem de Marcação de Hipertexto) é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web.

- ✓ **CSS** - Cascading Style Sheets (CSS) é um mecanismo para adicionar estilo (cores, fontes, espaçamento, etc.) a um documento web.

- ✓ **JavaScript** - (frequentemente abreviado como JS) é uma linguagem de programação interpretada estruturada, de script em alto nível com tipagem dinâmica fraca e multiparadigma (protótipos, orientado a objeto, imperativo e, funcional).

- ✓ **NodeJS** - Node.js é um software de código aberto, multiplataforma, baseado no interpretador V8 do Google e que permite a execução de códigos JavaScript fora de um navegador web.

Figura 1- NodeJS

HTML

Link: https://www.w3schools.com/html/

Neste projeto foi utilizado o HTML para a construção da nossa página Web, juntamento com o CSS (aparência/apresentação) e o JavaScript (funcionalidade/comportamento de uma página da web).

Foram utilizados os métodos GET,POST,PUT,DELETE para os usuários, animais e organizações. As figuras a seguir mostram a utilização destes métodos.

```
<h2>GET</h2>
<input id="getId" placeholder="Id">
<button onclick= "getUsuario()">Usuário</button>
<button onclick="getAnimal()">Animal</button>
<button onclick="getOrganizacao()">Organização</button>
```

Figura 2 – GET

Foi utilizado o elemento **Form** para a construção do formulário, juntamento com o atributo **Button** que enviar os dados recolhidos no formulário para o local indicado e o **Input** que é usado para criar controles interativos para formulários baseados na web para receber dados do usuário.

```
<h2>POST</h2>
<Label>Adicionar Usuário</Label>
<br>
<form class="postForm" action="/login" method="POST">
    <input id="addUsuario_id" placeholder="Id">
    <input id="addNome_id" placeholder="Nome">
    <input id="addEmail_id" placeholder="Email">
    <input id="addMorada_id" placeholder="Morada">
    <input id="addNascimento_id" placeholder="Data de Nascimento" type="date">
    <input id="addTelefone_id" placeholder="Telefone">
    <input id="addSexo_id" placeholder="Gênero">
    <input id="addNome_usuario_id" placeholder="Nome de Usuário">
    <input id="addSenha_id" placeholder="Senha">
    <button type="button" onclick="postUsuario()" >Enviar</button>
</form>
```

Figura 3- POST – Adicionar Usuário

O utilizador insere nos inputs as informações referentes ao que é pedido, após a inserção das informações, prime o botão para enviar essas informações para a base de dados onde fica guardado as informações dos Usuários. A seguir temos o Post da Criação dos animais, funcionando da mesma maneira.

```
<br>
<Label>Adicionar Animal</Label>
<br>
<form class="postForm">
    <input id="addAnimal_id" placeholder="Id">
    <input id="addTipo_id" placeholder="Tipo">
    <input id="addRaca_id" placeholder="Raça">
    <input id="addCor_id" placeholder="Cor">
    <input id="addIdade_id" placeholder="Idade">
    <input id="addSexo_id" placeholder="Sexo">
    <input id="addLocalidade_id" placeholder="Localidade">

    <button type="button" onclick="postAnimal()" >Enviar</button>
</form>
```

Figura 4 – POST - Adicionar Animal

Foi o utilizado o método PUT, para fazer a atualização dos dados que já existiam, no exemplo abaixo verificamos a atualização do Usuário, no qual o utilizador pode alterar qualquer informação, para salvar a alteração ele deve premir o botão enviar para atualizar na base de dados.

```
<h2>PUT</h2>
<Label>Atualizar Usuario</Label>
<br>
<form class="postForm">
    <input id="addUsuario_ID" placeholder="Id">
    <input id="addNome_ID" placeholder="Nome">
    <input id="addEmail_id" placeholder="Email">
    <input id="addMorada_id" placeholder="Morada">
    <input id="addNascimento_id" placeholder="Data de Nascimento" type="date">
    <input id="addTelefone_id" placeholder="Telefone">
    <input id="addSexo_id" placeholder="Gênero">
    <input id="addNome_usuario_id" placeholder="Nome de Usuário">
    <input id="addSenha_id" placeholder="Senha">
    <button type="button" onclick="putUsuario()" >Enviar</button>
</form>
```

Figura 7 – PUT – Atualizar Usuário

Foi o utilizado o método DELETE, para fazer a eliminação dos elementos que já existiam, para fazer a eliminação, ele deve escolher o ID e qual o tipo de elemento que pretende eliminar, após isso deve premir o botão referente ao elemento para eliminar o elemento da base de dados.

```
<h2>DELETE</h2>
<input id="deleteId" placeholder="id">
<button onclick="deleteUsuario()">Usuario</button>
<button onclick="deleteAnimal()">Animal</button>
<button onclick="deleteOrganizacao()">Organização</button>
```

Figura 8 - DELETE

CSS

Link: https://www.w3schools.com/css/

O CSS foi utilizado para alterar a cor do fundo, margem, fonte e espaçamento entre parágrafos e outras coisas. Não foi tão utilizado para a resolução do nosso projeto.

```css
footer {
    position: fixed;
    bottom: 0;
    text-align: center;
    padding: 10px 0;
    background-color: brown;
    width: 100%;
    color: white;
}

body{
    margin: 0;
    padding: 0;
}

#titleheader{
    text-align: center;
}

#content{
    margin: 0 auto ;
    display: flex;
    justify-content: space-around;
    width: 85%;
}
#result{
    margin-left: 3em;
    background-color: rgba(122, 176, 247, 0.226);
    min-height: calc(100vh - 11em);
    word-wrap: break-word;
}
```

JavaScript

Link: https://www.w3schools.com/js/

Utilizando o JavaScript, foram criadas as funções: getUsuario, getAnimal, getOrganizacao, postUsuario, postAnimal, postOrganizacao, putUsuario, putAnimal, putOrganizacao, deleteUsuario, deleteAnimal, deleteOrganizacao.

Abaixo temos um exemplo de cada tipo de função que foi referida acima. O processo é semelhante em quase todos, só mudando o elemento que vai ser trabalhado.

Funções

- getUsuario() - Esta função recebe o id que foi inserido pelo utilizador e vai buscar na base de dados toda a informação referente a esse ID. No caso a informação referente ao usuário.

```
function getUsuario() {
    let id = document.getElementById("getId").value
    var xmlhttp = new XMLHttpRequest();

    xmlhttp.onreadystatechange = () => {
        if(xmlhttp.readyState == 4 && xmlhttp.status == 200){
            let str = "";
            JSON.parse(xmlhttp.response).forEach(element => {
                str += `
                    ID> ${element.id_usuario}<br>
                    Nome> ${element.nome}<br>
                    Email> ${element.email}<br>
                    Morada> ${element.morada}<br>
                    Idade> ${element.idade}<br>
                    Telefone> ${element.telefone}<br>
                    Sexo> ${element.sexo}<br>
                    Nome de Usuario> ${element.nome_usuario}<br>
                    Senha> ${element.senha}<br>
                    <br>
                    `
            });
            changeResultText(str)
        }
    }
    xmlhttp.open("GET", `/usuario/${id}`);
    xmlhttp.setRequestHeader("Content-type", "application/json");
    xmlhttp.send();
}
```

- postAnimal() - Esta função recebe as informações que foram inseridas pelo utilizador e vai enviar para a base de dados toda a informação para ser armazenada. No caso abaixo a informação referente a criação de um Animal.

```
function postAnimal(){
    let idAnimal = document.getElementById("addAnimal_id").value;
    let tipo = document.getElementById("addTipo_id").value;
    let raca = document.getElementById("addRaca_id").value;
    let cor = document.getElementById("addCor_id").value;
    let idade = document.getElementById("addIdade_id").value;
    let sexo = document.getElementById("addSexo_id").value;
    let localidade = document.getElementById("addLocalidade_id").value;
    let obj = {idAnimal,tipo,raca,cor,idade,sexo,localidade}
    var xmlhttp = new XMLHttpRequest();

    xmlhttp.onreadystatechange = () => {
        if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            let str = "";
            let element = JSON.parse(xmlhttp.response)
                str += `
                ID> ${element.id_animal}<br>
                Tipo> ${element.tipo}<br>
                Raca> ${element.raca}<br>
                Cor> ${element.cor}<br>
                Idade> ${element.idade}<br>
                Sexo> ${element.sexo}<br>
                Localidade> ${element.localidade}<br>
                <br>
                `
        }
        changeResultText(str)
    }
    xmlhttp.open("POST", `/animal`);
    xmlhttp.setRequestHeader("Content-type", "application/json");
    xmlhttp.send(JSON.stringify(obj));
}
```

- putOrganizacao() - Esta função recebe as informações que foram inseridas pelo utilizador e vai atualizar as informações antigas e enviar novamente para a base de dados para ser armazenada. No caso abaixo a informação referente a atualização de uma organização.

```
function putOrganizacao(){
    let idOrganizacao = document.getElementById("addOrganizacao_id").value;
    let nomeOrganizacao = document.getElementById("addNomeOrganizacao_id").value;
    let localidade = document.getElementById("addLocalidadeOrganizacao_id").value;
    let obj = {idOrganizacao,nomeOrganizacao,localidade}
    var xmlhttp = new XMLHttpRequest();

    xmlhttp.onreadystatechange = () => {
        if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            let str = "";
            let element = JSON.parse(xmlhttp.response)
                str += `
                ID> ${element.id_organizacao}<br>
                    Nome> ${element.nome}<br>
                    Localidade> ${element.localidade}<br>
                    <br>

                `
        }
        changeResultText(str)
    }
    xmlhttp.open("PUT", `/organizacao`);
    xmlhttp.setRequestHeader("Content-type", "application/json");
    xmlhttp.send(JSON.stringify(obj));
}
```

- deleteUsuario() - Esta função recebe o id que foi inserido pelo utilizador e vai apagar toda a informação referente a esse ID. No caso a informação referente ao usuário.

```
function deleteUsuario(){
    let id = document.getElementById("deleteId").value;
    var xmlhttp = new XMLHttpRequest();

    xmlhttp.onreadystatechange = () => {
        if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            let str = "";
            let element = JSON.parse(xmlhttp.response)
                str += `
                ID> ${element.id_usuario}<br>
                Nome> ${element.nome}<br>
                Email> ${element.email}<br>
                Morada> ${element.morada}<br>
                Idade> ${element.idade}<br>
                Telefone> ${element.telefone}<br>
                Sexo> ${element.sexo}<br>
                Nome de Usuario> ${element.nome_usuario}<br>
                Senha> ${element.senha}<br>
                <br>
                `
                changeResultText(str)
        }
    }
    xmlhttp.open("DELETE", `/usuario`);
    xmlhttp.setRequestHeader("Content-type", "application/json");
    xmlhttp.send();
}
```

NodeJS

Link: https://www.w3schools.com/nodejs/

Node.js é uma plataforma para construir aplicativos de servidor rápidos e escaláveis usando JavaScript. Node.js é o tempo de execução e npm é o gerenciador de pacotes para módulos Node.js.

O Visual Studio Code oferece suporte para as linguagens JavaScript e TypeScript prontas para usar, bem como para depuração Node.js. No entanto, para executar um aplicativo Node.js, você precisará instalar o tempo de execução Node.js em sua máquina.

Utilizamos o NodeJS e o Express para criar nosso servidor web.

Package.json

O package.json é um repositório central de configuração para ferramentas criado para o NodeJS.

```
{
  "name": "projeto-pis",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  ▷ Depurar
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.2",
    "jsonwebtoken": "^8.5.1",
    "mysql": "^2.18.1"
  }
}
```

Request-handler

O request-handler é um script que é usado para definições das funções que serão executadas em resposta a cada pedido (uma por cada caminho). O request-handler utiliza o mysql para fazer a gestão dos dados. Foram criadas varias funções para a recolha de dados, como : getUsuario, getAnimais, getOrganizacao, etc. As funções recebem os dados do mysql e com isso utilizam e fazem as verificações e em seguida exportam os objetos. Conseguimos ver isso com as funções abaixo.

Função – getUsuario

```
function getUsuario(req, res) {
    var connection = mysql.createConnection(options);
    console.log(options);
    connection.connect();
    var query = "SELECT id_usuario AS 'Id', nome AS 'Nome', email AS 'email', morada AS 'morada',

    connection.query(query, function (err, rows) {
        if (err) {
            res.json({"message": "error", "error": err });
        } else {
            res.json({"message": "success", "usuario": rows });
        }
    });
}
module.exports.getUsuario = getUsuario;
```

Função - postUsuario

```
function postUsuario(req,res) {
    var connection = mysql.createConnection(options);
    let nome = req.body.nome;
    let email = req.body.email;
    let morada = req.body.morada;
    let idade = req.body.idade;
    let telefone = req.body.telefone;
    let sexo = req.body.sexo;
    let nome_usuario = req.body.nome_usuario;
    let senha = req.body.senha;


        if (nome == undefined , email == undefined , morada == undefined,
            idade == undefined , telefone == undefined,
            sexo == undefined , nome_usuario == undefined, senha == undefined) {
            res.send("Undefined");
        }
        else {
            let sqlquery =  `INSERT INTO usuario(nome, email, morada, idade, telefone,
            connection.query(sqlquery, (err, result) => {
                if (err) throw err;
                res.send(result);
            })
        }

}

module.exports.postUsuario = postUsuario;
```

Função - putAnimal

```
function putAnimal(req, res) {
    let connection = mysql.createConnection(options);
    let tipo = req.body.tipo;
    let raca = req.body.raca;
    let cor = req.body.cor;
    let idade = req.body.idade;
    let sexo = req.body.sexo;
    let localidade = req.body.localidade;

    let sql = "UPDATE animal SET tipo= ?,raca= ?,cor= ?,idade= ?,sexo= ?,localidade =?, WHERE id_animal= ?";
    connection.connect(function (err) {
        if (err) throw err;
        connection.query(sql, [tipo,raca,cor,idade,sexo,localidade, req.params.id_animal], function (err, rows) {
            if (err) {
                res.sendStatus(500);
            } else {
                connection.query("Select id_animal, tipo, raca, cor, idade, sexo, localidade FROM animal where id_animal = ?", [req.params.id_animal], (err, result) => {
                    res.send(result)
                })
            }
        });
    });
}
module.exports.putAnimal = putAnimal;
```

Função - deleteUsuario

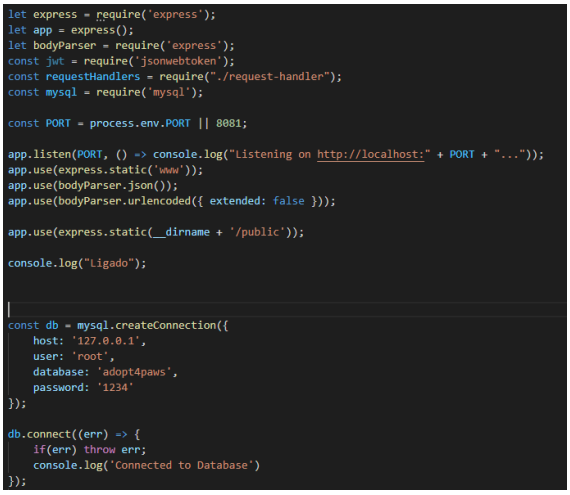
```
function deleteUsuario(req, res) {
    let query = 'DELETE FROM usuario WHERE id_usuario = ?';
    let connection = mysql.createConnection(options);
    connection.connect(function (err) {
        if (err) throw err;
        connection.query(query, [req.params.id_usuario], function (err) {
            if (err) {
                res.sendStatus(404);
            } else {
                res.sendStatus(200);
            }
        });
    });
}

module.exports.deleteUsuario = deleteUsuario;
```

Express

Express é uma estrutura de aplicativo da web Node.js mínima e flexível que fornece um conjunto robusto de recursos para desenvolver aplicativos da web e móveis. Ele facilita o rápido desenvolvimento de aplicativos da Web baseados em Node. A seguir estão alguns dos principais recursos da estrutura Express -

- ✓ Permite configurar middlewares para responder a solicitações HTTP.
- ✓ Define uma tabela de roteamento que é usada para realizar diferentes ações com base no método HTTP e URL.
- ✓ Permite renderizar páginas HTML dinamicamente com base na passagem de argumentos para modelos



```
let express = require('express');
let app = express();
let bodyParser = require('express');
const jwt = require('jsonwebtoken');
const requestHandlers = require("./request-handler");
const mysql = require('mysql');

const PORT = process.env.PORT || 8081;

app.listen(PORT, () => console.log("Listening on http://localhost:" + PORT + "..."));
app.use(express.static('www'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));

app.use(express.static(__dirname + '/public'));

console.log("Ligado");


const db = mysql.createConnection({
    host: '127.0.0.1',
    user: 'root',
    database: 'adopt4paws',
    password: '1234'
});

db.connect((err) => {
    if(err) throw err;
    console.log('Connected to Database')
});
```

Server.js

```
app.get("/id_usuario", requestHandlers.getUsuario);

app.get("/id_animal", requestHandlers.getAnimal);

app.get("/id_organizacao", requestHandlers.getOrganizacao);

/*POST */

app.post("/",requestHandlers.postUsuario);

app.post("/",requestHandlers.postAnimal);

app.post("/",requestHandlers.postOrganizacao);


/*PUT */

app.put("/id_usuario", requestHandlers.putUsuario);

app.put("/id_animal", requestHandlers.putAnimal);

app.put("/id_organizacao", requestHandlers.putOrganizacao);

/*DELETE */

app.delete("/id_usuario", requestHandlers.deleteUsuario);

app.delete("/id_animal", requestHandlers.deleteAnimal);

app.delete("/id_organizacao", requestHandlers.deleteOrganizacao);
```

Server.js

Para a utilização do Express precisamos chamar o módulo express, como foi usado na linha 1 (Figura Server.js) **let express = require (“express”);**

Foi utilizado o roteamento que é como um aplicativo responde a uma solicitação do cliente para um ponto de extremidade específico, que é um URI (ou caminho) e um método de solicitação HTTP específico (GET, POST e assim por diante).

- Get para chama a função para receber os dados referentes ao pedido , como exemplo: app.get(“/id_usuario”,requestHanldlers.getUsuario).

- Post para chamar a função e para enviar os dados referentes ao pedido , como exemplo: app.post(“/”,requestHanldlers.postUsuario).

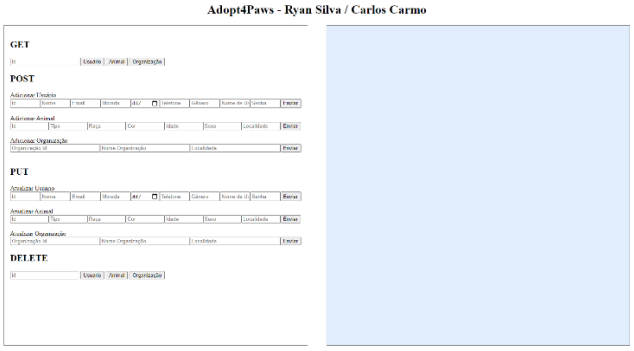
- Put para chamar a função e para enviar os dados referentes ao pedido , como exemplo: app.post(“/id_usuario”,requestHanldlers.putUsuario).

- Delete para chama a função e para eliminar os dados referentes ao pedido , como exemplo: app.delete(“/id_usuario”,requestHanldlers.deleteUsuario).

A função app.listen(8081, function()) do Express inicia um socket UNIX e escuta as conexões em um caminho fornecido. Ou seja, quando o servidor for ligado, será ligado na **localhost: 8081**. Não é possível iniciar o Express sem dizer onde escutar.

Manual de Utilizador

Página Principal



Na Página Principal apresentam-se 2 Formulários:

- Formulário a esquerda – Neste formulário podemos tanto: Ver, Adicionar, atualizar ou Eliminar um dos elementos que possuem, no caso os elementos são: usuários, animais e organização.
- Formulário a direita – Neste formulário conseguimos verificar todas as ações que foram realizadas no formulário a esquerda, como por exemplo os GET,POST,PUT,DELETE referente aos elementos.

GET

- O utilizador insere o ID referente ao seu desejo, após isso prime o botão referente ao elemento que quer observar. Após isso será mostrado no formulário a direita o seu pedido.

POST

- O utilizador insere as informações referente ao elemento, respeitando o tipo de dados, após a inserção da informação, prime o botão enviar para que a informação inserida seja enviada para a base de dados.

PUT

- O utilizador pode atualizar as suas informações que foram inseridas anteriormente, atualizando os campos que deseja , prime o botão enviar para atualizar a informação na base de dados.

DELETE

- O utilizador insere o ID referente ao seu desejo de eliminação, após isso prime o botão referente ao elemento que quer eliminar. Após isso o elemento será apagado.

Conclusão

Neste relatório abordamos os Manuais Técnicos e de Utilizador que possuíram os seguintes tópicos: Base de Dados, Técnicas Utilizadas, HTML, CSS, NodeJS, JavaScript, Express e como utilizar o nosso servidor web.

Cumprimos os objetivos que nos foram propostos, porém tivemos dificuldades em algumas partes, como na montagem do servidor web e na ligação com o MySQL, porém conseguimos resolver esses problemas.

Este trabalho foi muito importante para a nossa compreensão sobre a matéria de PIS uma vez que nos permitiu compreender melhor e desenvolver nossas competências de investigação.