ESTGOH
Escola Superior de Tecnologia e Gestão de Oliveira do Hospital
Instituto Politécnico de Coimbra

Licenciatura em Engenharia Informática

Programação para a Internet II

2022/2023

Relatório Avaliação Periódica 2

Realizado em: 31/05/2023

Vasco Tomás Ramos Rodrigues [a2021154888]

Índice

Índices de figuras

1. Introdução

Este é o Trabalho nº2 realizado a 31/05/2023, no âmbito da unidade curricular Programação para a Internet II, com o objetivo de desenvolver uma pequena aplicação PHP e **MySQL** denominada **GymLog**, com o objetivo mais concreto do desenvolvimento da área de administração do sistema na qual o administrador deste possa gerir os exercícios disponibilizados na *APP*. Neste trabalho foram abordados temas como **programação orientada a objetos**, **programação PHP**, **bases de dados relacionais** e internacionalização de sites. Neste relatório especificadas as escolhas e opções tomadas durante a realização do código para a boa realização deste trabalho. Este trabalho estará dividido em principalmente um único subcapítulo onde será feita a especificação do código e escolhas.

A metodologia usada para este trabalho foi principalmente a utilização de resumos, apontamentos, material disponibilizado e exercícios realizados em aulas com a ajuda do docente para auxiliar na conclusão bem-sucedida de todos os requisitos exigidos, para além deste também houve a necessidade de uma pesquisa mais aprofundada para perceber alguns comandos e algumas peculiaridades da programação com a linguagem **PHP**.

2. Métodos e Materiais

2.1. Métodos

Para a realização deste trabalho foi usado apenas o software *Visual Studio Code* onde foi feito todo o código **css**, **javascript**, **html** e PHP e o site **Bootstrap** para a edição visual de alguns conteúdos do site. Como já foi referido a metodologia usada foi principalmente o uso de resumos, apontamentos, material disponibilizado e exercícios realizados em aulas com a ajuda do docente para assim obter o melhor trabalho possível

Os requisitos implementados foram os seguintes:

Requisitos específicos

RE-1 Página de autenticação do administrador (a base de dados disponibilizada já inclui um administrador com o username = admin e password = admin. Convém mencionar que a password foi encriptada com a função password_hash).

RE-2 Alteração da password do administrador (nota: a password deverá ser guardada encriptada). Deverá ser solicitada a password atual e a nova password, sendo que a password apenas será alterada se tiver sido inserida a password antiga correta. Obviamente, só deverá ser possível aceder a esta funcionalidade depois de se ter autenticado.

RE-3 Funcionalidade permita terminar a sessão (logout).

RE-4 Após ser autenticado, o administrador deverá ser redirecionado para uma página na qual deverá surgir uma tabela contendo os nomes de todos os exercícios disponibilizados.

RE-5 Para cada item da tabela deverá ser ainda apresentado:

RE-5.1 Um botão "ver detalhes" que permita aceder a uma página com todos os detalhes daquele exercício;

RE-5.2 Um botão "editar" que permita a uma página que permita editar os detalhes daquele exercício; nos casos em que tal seja possível, os campos de formulário deverão surgir preenchidos com os valores atuais.

RE-5.3 Um botão que permita ativar ou desativar aquele exercício. Se o exercício estiver ativo, o botão deverá ter o texto "desativar" e vice-versa.

RE-6 Deverá ainda ser disponibilizado um botão que permita aceder a uma página onde possam ser inseridos nossos exercícios. Deverá ter em atenção que um novo exercício deverá ficar automaticamente ativo. Além disso, no caso da imagem do exercício deverá passar a ser efetuado o upload do ficheiro, muito embora da base de dados deva ser apenas guardado o nome do ficheiro devidamente modificado para evitar duplicações.

Requisitos genéricos

RG-1 A aplicação desenvolvida deverá estar preparada para ser facilmente disponibilizada em diferentes línguas (não simultaneamente).

RG-2 Deverá garantir que não seja possível aceder às páginas de acesso restrito sem ter sido previamente feito corretamente a autenticação.

RG-3 Todos os formulários deverão ser devidamente validados com PHP e HTML5 e/ou JavaScript.

RG-4 Na resolução do trabalho terá de ser utilizada programação orientada a objetos e para os acessos à base de dados terá de ser utilizado PDO (PHP Data Objetas) e prepared statements (sempre que existirem inputs);

RG-5 Poderá utilizar livremente frameworks de IU como o Bootstrap e outras que de alguma forma considere úteis.

Todos os requisitos foram realizados com sucesso não tendo ficado nenhum por fazer ou incompleto.

Este trabalho foi bastante rápido de se fazer tendo em conta o esforço já tido em aula onde foram abordados todos os temas e recursos necessários para o trabalho. Onde pode ter sido gasto algum tempo foi a perceber a dinâmica de alteração entre *actions* para tratar os objetos, a implementação de várias línguas no site que apesar de não ser uma tarefa difícil mostrou-se demorada e a aplicação de estilos ao site. Este demorou cerca de 14 horas.

2.2. Materiais

Para a realização deste trabalho foram utilizados:

- **Visual Studio Code;**
- **Bootstrap.**

3. Resultados

Ao longo deste capítulo será feita a especificação das escolhas, algoritmos e pedaços de código únicos usados para a realização deste trabalho de uma forma sucinta e breve.

Para o correto funcionamento da aplicação, esta deve ser iniciada através do ficheiro **PHP** "**formulário.php**".

Antes da especificação de escolhas, foram implementados aspetos para proteger a aplicação como a verificação de dados, para impossibilitar a inserção de dados inválidos, como a verificação de dados vazios ou não numéricos. Também foram implementados aspetos informativos para o utilizador como, alertas de inserção de dados bem-sucedidos.

Isto através de mensagens em **HTML** normais ou através de *toastes*, **Figura 3-1** e **Figura 3-2**.

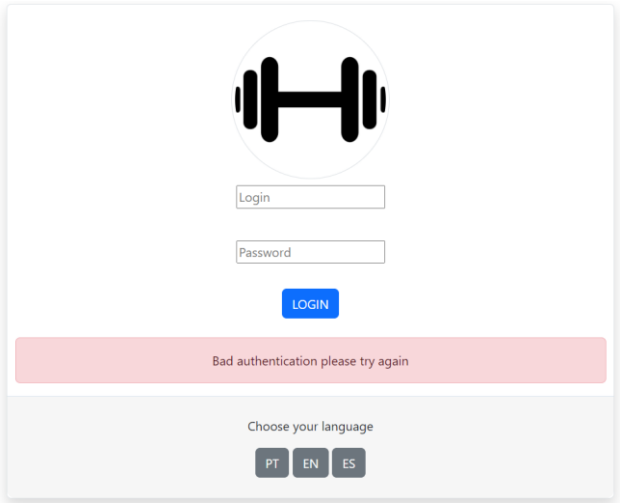

Figura 3-1 - Mensagem de erro

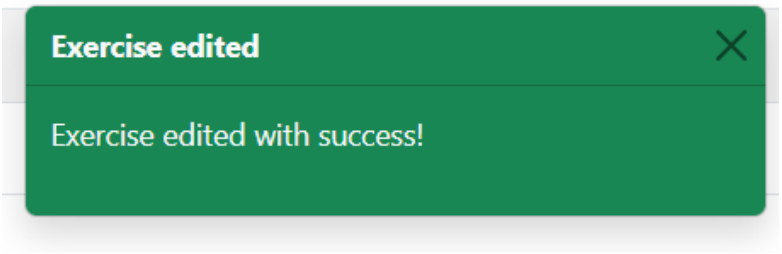

Figura 3-2 - Toast

Vale também mencionar que todo o visual desta aplicação foi orientado para dispositivos como o portátil e computador fixo, e, portanto, o seu aspeto num telemóvel não foi tomado muito em conta.

3.1. Algoritmos

3.1.1. Apresentação de conteúdos

Para a apresentação de conteúdos no site foi utilizada sempre a mesma forma, ou seja, para evitar a redundância do código seria criada sempre uma *string* na qual estariam todos os elementos **HTML** que iriam ser apresentados, **Figura 3-3**.

```
$strForm='<!DOCTYPE html>
    <html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">
        <title>GymLog</title>
    </head>
    <style>
        .container{
            border-color: #32383e;
            padding-top: 5rem;
            padding-bottom: 5rem;
            margin-top: 1rem;
            border-radius: 0.375rem;
        }
        .formFooter {
            background-color: #f6f6f6;
            border-top: 1px solid #dce8f1;
            padding: 25px;
            border-radius: 0 0 10px 10px;
        }
        .wrapper {
            align-items: center;
            flex-direction: column;
            justify-content: center;
            border-top: var(--bs-border-width) solid var(--bs-border-color);
            width: 50%;
            margin-left: auto;
```

Figura 3-3 - Exemplo de código para apresentação de conteúdos

Para a alteração da língua na qual estavam os conteúdos recorreu-se ao método i18n, que consiste na criação de ficheiros de constantes para diferentes línguas substituindo os conteúdos apresentados por estas constantes.

Para isto foi criada um variável de sessão onde está guardada a língua escolhida pelo utilizador, caso esta variável não esteja definida a língua padrão será o inglês, depois é testada a existência de um ficheiro **i18n** na língua escolhida caso este exista será feito o *include* do mesmo caso contrário será feito o *include* do ficheiro padrão inglês, **Figura 3-4** e **Figura 3-5**.

```
include("i18n.php");
session_start();
if(isset($_SESSION['idioma'])){
    $lang=$_SESSION['idioma'];
}
else{
    $lang='en';
}
if (file_exists ("i18n_" . $lang . ".php")){
    include "i18n_" . $lang . ".php";
}
else{
    include "i18n_en.php";
}
```

Figura 3-4 - Include dos ficheiros i18n

```php
<?php
    const LOGIN_BUTTON = "LOGIN";
    const LANGUAGE_LABEL = "Choose your language";
    const MSG_NO_EXERCISE = "There is no exercises, please add one";
    const DETAILS = "Details";
    const EDIT = "Edit";
    const DEACTIVATE = "Deactivate";
    const ACTIVATE = "Activate";
    const ADD_EXERCISE_BUTTON = "Add exercises";
    const LOGOUT_BUTTON = "Logout";
    const CH_PASSWORD_BUTTON = "Change password";

    const LABEL_NAME_EXE = "Name";
    const LABEL_DESC_EXE = "Description";
    const LABEL_PHOTO_EXE = "Photo";
    const LABEL_MUSCLES_EXE = "Muscles";
    const LABEL_REP_EXE = "Repetitions";
    const LABEL_STATE_EXE = "State";
    const LABEL_STATE_ACTIVE = "Active";
    const LABEL_STATE_DEACTIVE = "Inactive";
```

Figura 3-5 - Exemplo ficheiro i18n

Sendo feita a apresentação de conteúdos do site feita toda desta forma.

3.1.2. Obtenção de dados dos formulários

Para a obtenção de dados dos formulários é necessário o uso da variável **$_POST** como aprendido em aula, antes de trabalhar com valor desta variável é primeiro feito a verificação de se esta existe ou não está a **null**, inicialmente era também verificado se não estava vazia através da função **empty()** mas com o campo **required** nos *inputs* do formulário isso já não se mostrava necessário pois era garantido que ele pelo menos teria um valor, após estas verificações serão atribuídos estes valores a variáveis e tratados para um dado fim, **Figura 3-6**.

```php
if(isset($_POST["exerciseAdded"])){
    if(isset($_POST["nameExercise"]) && isset($_POST["descExercise"]) && isset($_FILES["photoExercise"])
    && isset($_POST["musclesExercise"]) && isset($_POST["repsExercise"])){
        if(!(empty($_POST["nameExercise"]) || empty($_POST["descExercise"]) || empty($_FILES["photoExercise"])
                || empty($_POST["musclesExercise"]) || empty($_POST["repsExercise"]))){
```

Figura 3-6 - Exemplo de obtenção de dados dos formulários

Em alguns casos a passagem de dados através dos formulários necessitava de outros dados para além dos inseridos, um caso será na edição de exercícios por exemplo, para isto foi utilizado um método em **php** que, apesar de não ser o mais seguro funciona, após alguma discussão com o professor foi percebido que seria mais seguro, apesar de não infalível, a utilização de *inputs* do tipo **hidden** que desta maneira seria impossível ter acesso via html ao **ID** de um exercício, **Figura 3-7**.

```php
<input type="submit" class="btn btn-primary" name="exerciseEdited['.$valor[0].']" value="'.i18n(EDIT_EXE_BUTTON).'">
```

Figura 3-7 - Passagem de dados

3.1.3. Algoritmos de inserção e edição de dados

Todas as inserções são feitas de maneiras idênticas seguindo um conjunto de passos, posto isto, às edições também aconteceria o mesmo pois estas são apenas um caso especial das inserções.

Primeiramente todas as *queries* feitas à base de dados que utilizem *inputs* introduzidos pelo utilizador, ou neste caso administrador, são protegidas através do uso de *prepared*

statements método este que consiste na preparação de uma *query* que será enviada à base de dados para assim evitar *SQL injection*.

Utilizando a inserção de exercícios como exemplo, para a sua inserção é primeiro necessário obter os dados, isto é feito através do método especificado em cima, após isto através das variáveis que serão obtidas irá ser criado o objeto pretendido, e, em seguida, será inserido através do método do próprio objeto na base de dados, **Figura 3-8** e **Figura 3-9**.

```
$exe = new exercicio($_POST["nameExercise"],$_POST["descExercise"],$pho,$_POST["musclesExercise"],$_POST["repsExercise"]);
$exe->insereExercicio();
```

Figura 3-8 - Inserção de dados na base de dados (1)

```
public function insereExercicio(){
    $DBH=bd::get_instance();
    $STH = $DBH->prepare("INSERT INTO exercise (e_name,e_desc,e_photo,e_muscles,e_repetitions,e_active) values (?,?,?,?,?,?);");
    $STH->execute(array($this->name,$this->desc,$this->photo,$this->muscles,$this->reps,$this->active));
}
```

Figura 3-9 - Inserção de dados na base de dados (2)

No caso da edição de dados, será feita a pesquisa deste através de uma classe, **gereExercicios** que permitirá pesquisar, através do **ID**, um exercício, **Figura 3-10**, **ID** este que será obtido através do campo *input*, após alguma pesquisa foi concluído que ao utilizar o método de passagem de dados por *input* no próprio nome deste é criado um *array* de chaves com o valor selecionado, então através da função *array_keys()* é possível obter o seu valor, **Figura 3-11**, em seguida irão ser feitos os mesmo passos especificados em cima para a inserção de dados alterando apenas a *query* para um **UPDATE**, **Figura 3-12**.

```
public function pesquisaExercicios($eId){

    $DBH=bd::get_instance();
    $STH=$DBH->prepare('SELECT * FROM exercise
                        WHERE e_id=?;');
    $STH->execute(array($eId));
    $STH->setFetchMode(PDO::FETCH_OBJ);

    $record = $STH->fetch();


    $exe=new exercicio($record->e_name,$record->e_desc,$record->e_photo,
                       $record->e_muscles,$record->e_repetitions,$record->e_active,$record->e_id);
    return $exe;

}
```

Figura 3-10 - Método de pesquisa de exercícios

```
$valor=array_keys($_POST['edit']);
$gereEx = new gereExercicio();
$exe=$gereEx->pesquisaExercicios($valor[0]);
```

Figura 3-11 - Obter ID do exercício

```
public function editaExercicio($nome,$desc,$photo,$muscles,$repetitions){
    $DBH=bd::get_instance();
    $STH = $DBH->prepare("UPDATE exercise SET e_name = ?, e_desc = ?, e_photo = ?, e_muscles = ?, e_repetitions = ? WHERE e_id = ?;");
    $STH->execute(array($nome,$desc,$photo,$muscles,$repetitions,$this->id));
}
```

Figura 3-12 - Método de edição de exercício

3.1.4. Listagem de múltiplos elementos

Todas as listagens neste trabalho, que neste caso é apenas uma, seguem o mesmo método.

Novamente através da classe **gereExercicios** irá ser feita a contagem de objetos na base de dados, caso não exista nenhuma é passada uma mensagem de informação, caso contrário é

chamado um método que obterá todos os objetos, criando um *array* com estes e retornando-o, **Figura 3-13**, em seguida será feito um ciclo **foreach** onde irão ser apresentados um a um cada um dos exercícios, **Figura 3-14**.

```php
public function listaExercicios(){
    $DBH=bd::get_instance();
    $STH=$DBH->query('SELECT * FROM exercise;');
    $STH->setFetchMode(PDO::FETCH_OBJ);

    $records = $STH->fetchAll();

    foreach($records as $record){
        $exe=new exercicio($record->e_name,$record->e_desc,$record->e_photo,
                           $record->e_muscles,$record->e_repetitions,$record->e_active,$record->e_id);
        array_push($this->exercicio,$exe);
    }
    return $this->exercicio;
}
```

Figura 3-13 – Obtenção dos vários objetos

```php
if($gereExe->contaExercicios()==0){
    $listaExe=i18n(MSG_NO_EXERCISE).'<br>';
}
else{
    $exercicios=$gereExe->listaExercicios();
    $listaExe=$strBoot.'<form method="POST" action="formularioExercicios.php">
                        <table class="table table-striped">
                            <thead class="MYthead-dark">
                                <tr>
                                    <th scope="col">'.i18n(LABEL_TABLE_EXE_NAME).'</th>
                                    <th scope="col">'.i18n(LABEL_TABLE_EXE_ACTIONS).'</th>
                                </tr>
                            </thead>
                            <tbody>';
    $estado='';
    foreach($exercicios as $e){
        if($e->getActive()){
            $estado=i18n(DEACTIVATE);
        }
        else{
            $estado=i18n(ACTIVATE);
        }

        $listaExe.='
        <tr>
            <td>'.$e->getNome().'</td>
            <td>
                <input type="submit" name="details['.$e->getId().']" value="'.i18n(DETAILS).'">
                <input type="submit" name="edit['.$e->getId().']" value="'.i18n(EDIT).'">
                <input type="submit" name="deactivate['.$e->getId().']" value="'.$estado.'"><br>
            </td>
        </tr>';
    }
    $listaExe.='
                    </tbody>
                </table>
            </form>';
}
```

Figura 3-14 - Listagem dos conteúdos

3.2. Métodos

3.2.1. Variáveis de sessão

Na questão de variáveis de sessão foram escolhidas o login apenas. Numa fase inicial foram escolhidas as variáveis login e password, mas obviamente por questões de segurança foi removida a password. Estas mantêm-se sempre ativas até existir um *logout* assim sempre que

um utilizador entre no site desde que não tenho feito o *logout*, ele será sempre redirecionado automaticamente para a página inicial do site.

4. Conclusão

Neste relatório foi feita a descrição das escolhas tomadas para a realização da aplicação proposta, oque pode ser concluído é que PHP torna todo o processo de tratamento de dados e a apresentação destes mais simples, tendo sido a primeira vez que é feito um site com acesso a base de dados relacionais, têm de ser tidas em conta vários aspetos, principalmente em programação orientada a objetos, modelo **mvc**, segurança e privacidade que deve ser garantida a todos os utilizadores que apesar do que já foi incluído com este aspeto ainda existe muito mais, como por exemplo algo que também foi discutido com o docente ao esclarecer uma dúvida, a forma usada neste trabalho para passar os **Ids** dos exercícios não é segura isto porque qualquer administrador presente na base de dados poderia ter acesso a eles através do código fonte do site e via **URL**, então foi discutido um método melhor que seria o uso de inputs *hiden*, isto apesar de ser mais seguro continuo a não ser a melhor opção pois, apesar de o utilizador comum já não ter acesso através do código fonte, este pode aceder aos detalhes de um exercício através do **URL** e do **ID** do exercício, então uma maneira verdadeiramente segura seria verificando no link, neste caso do detalhes de um exercício se este utilizador tinha permissão para os ver, apenas não foi implementado pois não era um requisito.

Cada um dos requisitos propostos pelo professor foi concluído com sucesso, devido ao esforço feito em aula pela minha parte e também pelos materiais disponibilizados pelo professor, por consequência não tomando muito tempo para a sua realização total.

A realização deste trabalho foi bastante interessante pois abriu portas para a aprendizagem de uma nova linguagem que apesar de já não ser tão usada para a criação de sites como em outros tempos, ainda é por larga maioria a linguagem mais utilizada atualmente, eu nunca gostei muito de programação orientada à internet, mas a inclusão de base de dados relacionais e a introdução da programação orientada a objetos nesta tornou tudo um pouco mais interessante, tornando também a realização deste trabalho mais prazerosa e não tão carregada.

Concluindo a realização deste trabalho foi bem-sucedida.

5. Referências

I. Enunciado do trabalho prático
II. Material de apoio disponibilizado pelo professor
III. contributors, Mark Otto, Jacob Thornton, and Bootstrap. «Get Started with Bootstrap». Acedido 6 de maio de 2023. https://getbootstrap.com/docs/5.3/getting-started/introduction/.
IV. «HTML Tutorial». Acedido 6 de maio de 2023. https://www.w3schools.com/html/.