
# FormaçõesLW
## Trabalho Prático de Avaliação - PHP

Miguel Torres Ramos, nº 20191185

Sebastião Condeixa Sarreira Monteiro, nº 20121814

**Professor**

Carlos Alves

**fevereiro de 2025**

# Índice geral

# 1. Introdução

Este projeto visa desenvolver uma aplicação web utilizando PHP, HTML, MySQL, DTD, XML e XSLT para a gestão de um website de cursos de formação. A plataforma será acessível a diversos tipos de utilizadores, incluindo visitantes, alunos, docentes e administradores, cada um com funcionalidades específicas adaptadas às suas necessidades. O objetivo principal é proporcionar uma experiência personalizada aos utilizadores, oferecendo-lhes acesso a recursos relevantes e simplificando os processos de gestão de inscrições e formações.

# 2. User Cases

**Visitante – Consultar os dados da empresa**

Ator: Visitante

Descrição: O visitante quer ver as informações sobre a empresa, como por exemplo a localização da sede, os horários de funcionamento e os preços.

Pré-condições: A página inicial estar acessível.

Fluxo principal:
1. O visitante acede á página inicial.
2. O website mostra as informações da empresa.

Pós-condições: Nenhuma

**Visitante – Registar-se como aluno**

Ator: Visitante

Descrição: O visitante regista-se no website para se tornar aluno.

Pré-condições: O visitante estar na página de registo.

Fluxo principal:
1. O visitante entra na página de registo.
2. Preenche um formulário com os seus dados pessoais (nome, password e email).
3. Submete o formulário.
4. O sistema guarda os dados e informa se o registo foi bem-sucedido, e avisa que o visitante tem de esperar uma validação por parte de um administrador.

Pós-condições: Os dados do visitante ficam guardados e o visitante fica registado como "visitante não validado".

**Aluno – Fazer login**

Ator: Aluno.

Descrição: O aluno faz login no website após a validação por um administrador.

Pré-condições: O aluno foi validado por um administrador.

Fluxo principal:
1. O aluno entra na página de login.
2. Insere as suas credenciais (nome e password).
3. O sistema verifica se as credenciais estão corretas e autêntica o aluno.

Pós-condições: O aluno está autenticado no sistema.

**Aluno – Gestão das inscrições**

Ator: Aluno.

Descrição: O aluno inscreve-se nas formações disponíveis ou cancela as suas inscrições.

Pré-condições: O aluno está autenticado no sistema.

Fluxo principal:

1. O aluno entra na página de gestão de inscrições.
2. Seleciona uma nova inscrição numa formação ou apaga uma inscrição já feita.
3. O sistema regista a nova inscrição ou o cancelamento.

Pós-condições: A alteração às inscrições é registada.

**Aluno – Gestão de dados pessoais**

Ator: Aluno.

Descrição: O aluno atualiza os seus dados pessoais.

Pré-condições: O aluno está autenticado no sistema.

Fluxo principal:

1. O aluno entra na página de gestão dos dados pessoais.
2. Altera os campos que quer (nome, password e email).
3. Submete as alterações.
4. O sistema atualiza os dados.

Pós-condições: Os dados pessoais do aluno são atualizados.

**Docente – Gestão de inscrições**

Ator: Docente.

Descrição: O docente gere as inscrições dos alunos nas suas formações.

Pré-condições: O docente está autenticado no website.

Fluxo principal:

1. O docente entra na página de gestão de inscrições.
2. Vê a lista de inscrições.
3. Aprova ou rejeita as inscrições conforme o desejado.
4. O sistema atualiza os dados das inscrições.

Pós-condições: Os dados das inscrições são atualizados.

**Docente – Gestão de dados pessoais**

Ator: Docente.

Descrição: O docente atualiza os seus dados pessoais.

Pré-condições: O docente está autenticado no website.

Fluxo principal:

1. O docente entra na página de gestão dos dados pessoais.
2. Altera os campos que quer (nome, password e email).
3. Submete as alterações.
4. O sistema atualiza os dados.

Pós-condições: Os dados pessoais do docente são atualizados.

**Docente – Gestão das suas formações**

Ator: Docente.

Descrição: O docente cria, edita ou remove as suas formações.

Pré-condições: O docente está autenticado no sistema.

Fluxo principal:

1. O docente entra na página de gestão de formações.
2. Cria, edita ou remove as suas formações.
3. Os dados das formações são atualizados no sistema.

Pós-condições: Os dados das formações são atualizados no sistema.

**Administrador – Validar o registo de alunos**

Ator: Administrador.

Descrição: O administrador aprova ou rejeita o registo de novos alunos.

Pré-condições:

1. O administrador está autenticado no sistema.
2. Existem registos à espera de validação.

Fluxo principal:

1. O administrador entra na página de gestão de utilizadores.
2. Vê a lista de utilizadores.
3. Seleciona um registo pendente e valida-o.
4. O sistema atualiza os dados do aluno.

Pós-condições: Os dados do aluno são validados.

**Administrador – Gestão de utilizadores**

Ator: Administrador.

Descrição: O administrador gere os utilizadores do website.

Pré-condições: O administrador está autenticado no sistema.

Fluxo principal:

1. O administrador entra na página de gestão de utilizadores.
2. Vê a lista de utilizadores.
3. Adiciona, edita ou remove utilizadores conforme necessário.
4. O sistema atualiza os dados dos utilizadores.

Pós-condições: Os dados dos utilizadores são atualizados.

**Administrador – Gestão de dados pessoais**

Ator: Administrador.

Descrição: O administrador atualiza os seus dados pessoais.

Pré-condições: O administrador está autenticado.

Fluxo principal:

1. O administrador entra na página de gestão dos seus dados pessoais.
2. Atualiza os campos que deseja.
3. Submete as suas alterações.
4. O sistema atualiza os dados pessoais do administrador.

Pós-condições: Os dados pessoais do administrador são atualizados.

**Administrador – Gestão de formações**

Ator: Administrador.

Descrição: O administrador cria, edita e remove formações dos sistema.

Pré-condições: O administrador está autenticado.

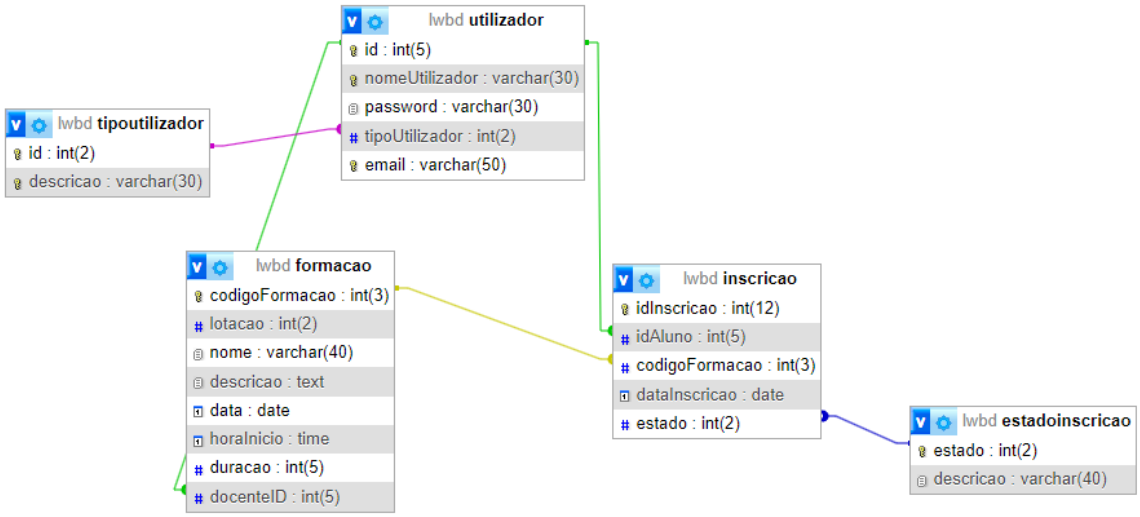
Fluxo principal:

1. O administrador entra na página de gestão de formações.
2. Vê uma lista das formações.
3. Altera as formações como desejado.
4. Os dados das formações são atualizados no sistema.

Pós-condições: Os dados das formações são atualizados no sistema.

# 3. Modelo relacional da base de dados

Para dar suporte ao sistema de gestão de formações, foi desenvolvido o seguinte modelo entidade-relação (E-R). São consideradas diversas tabelas que armazenam a informação das formações, dos utilizadores e respetivos tipos, bem como, das inscrições e respetivos estados.

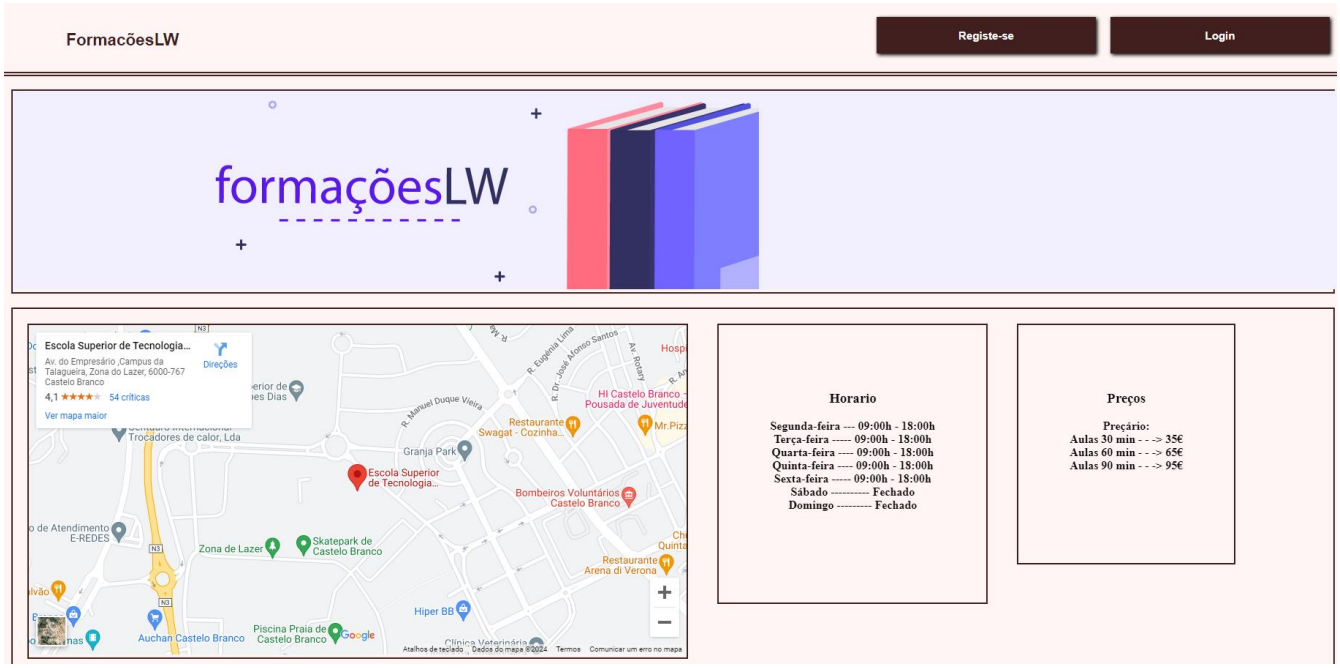


# 4. Screens da aplicação

## 4.1 Página Principal

A página principal apresenta o nome do salão “FormaçõesLW”, com um banner. Este título permite tanto nesta página como nas outras voltar à página principal. Apresenta também uma opção de login e de registo, estes botões são substituídos pelo botão de área pessoal e logout respetivamente quando é feito o login.

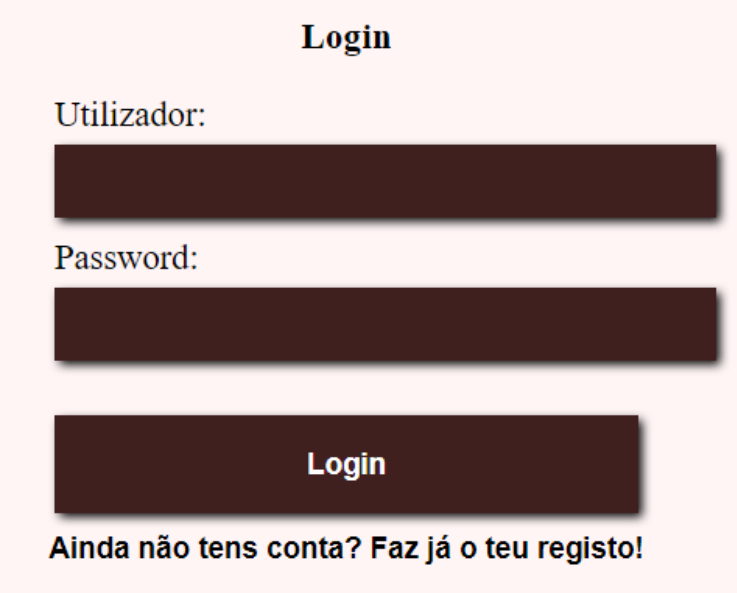
Para as informações do website a página apresenta a localização através de um mapa, para os horários uma tabela com os dias da semana e também os preços de cada aula.

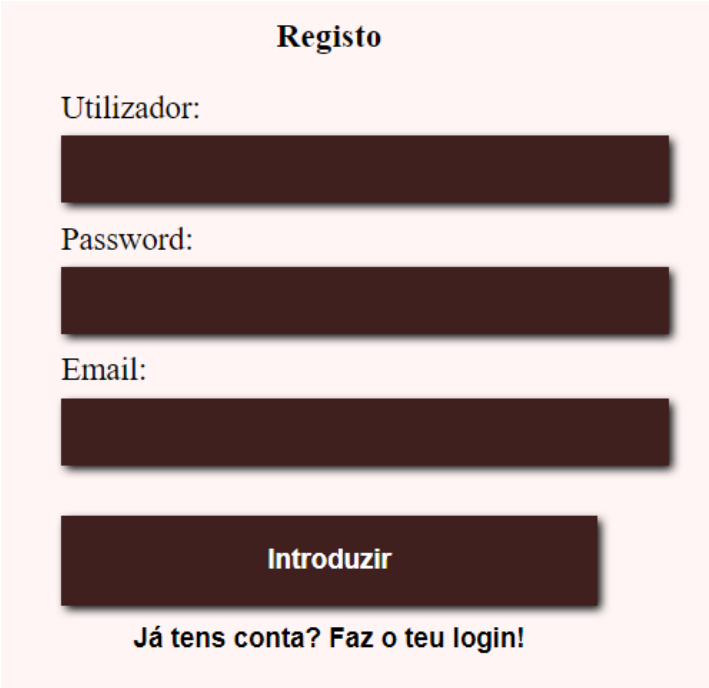


### 4.2 Página Login e Registo

A página de login permite entrar na conta de utilizador através do nome de utilizador e password.

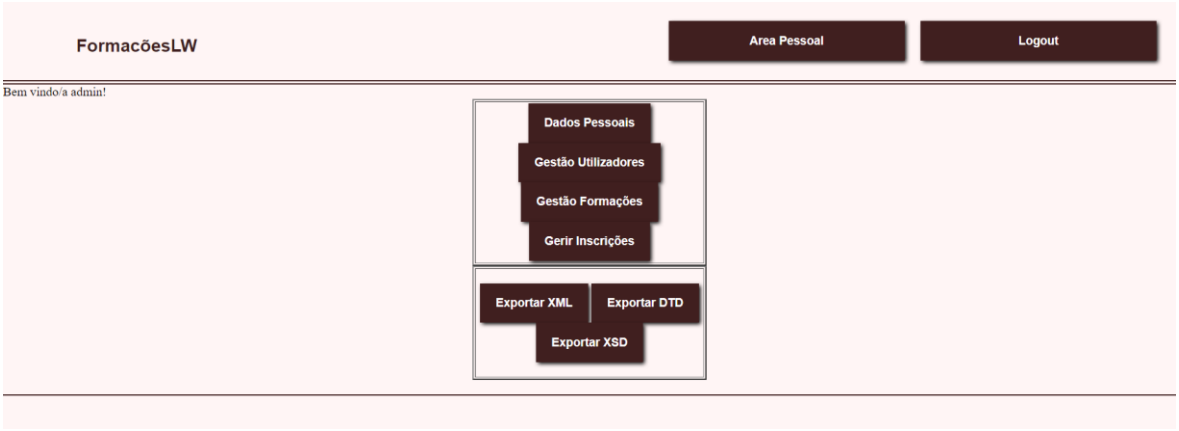
A página de registo permite criar uma conta de aluno (não validado), com os campos nome utilizador, password, número de telemóvel e email. Ambas as páginas têm uma maneira de trocar entre elas caso o utilizador já tenha (ou não) uma conta.



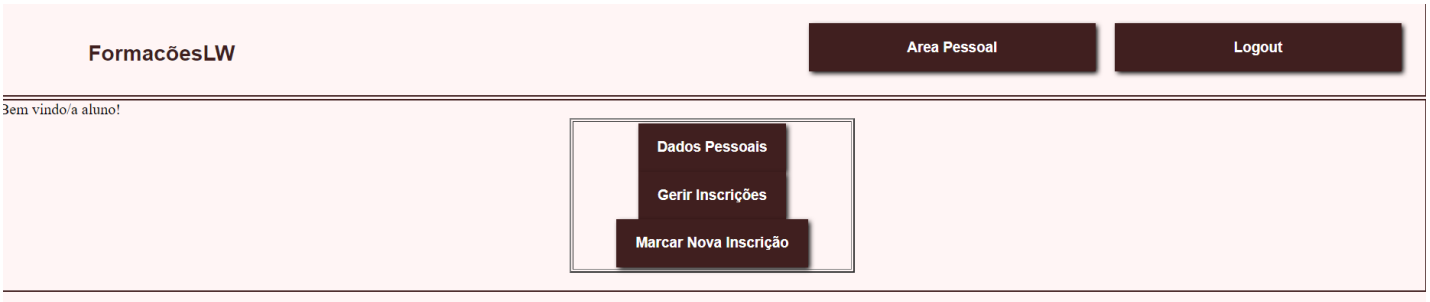



### 4.3 Página de gestão

A página de gestão permite o acesso às páginas de dados pessoais (todos os utilizadores), gestão de utilizadores (caso seja o admin), gestão de inscrições e gestão de formações (caso seja o docente ou admin) e marcação de reservas (apenas o aluno). Os admins podem ainda fazer a exportação da base de dados em XML, DTD e XSD. Ao clicar no botão, é criado o ficheiro na mesma diretoria.

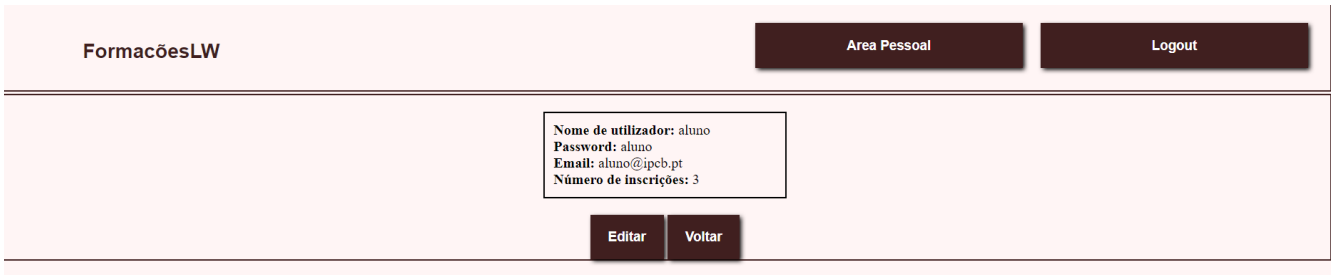






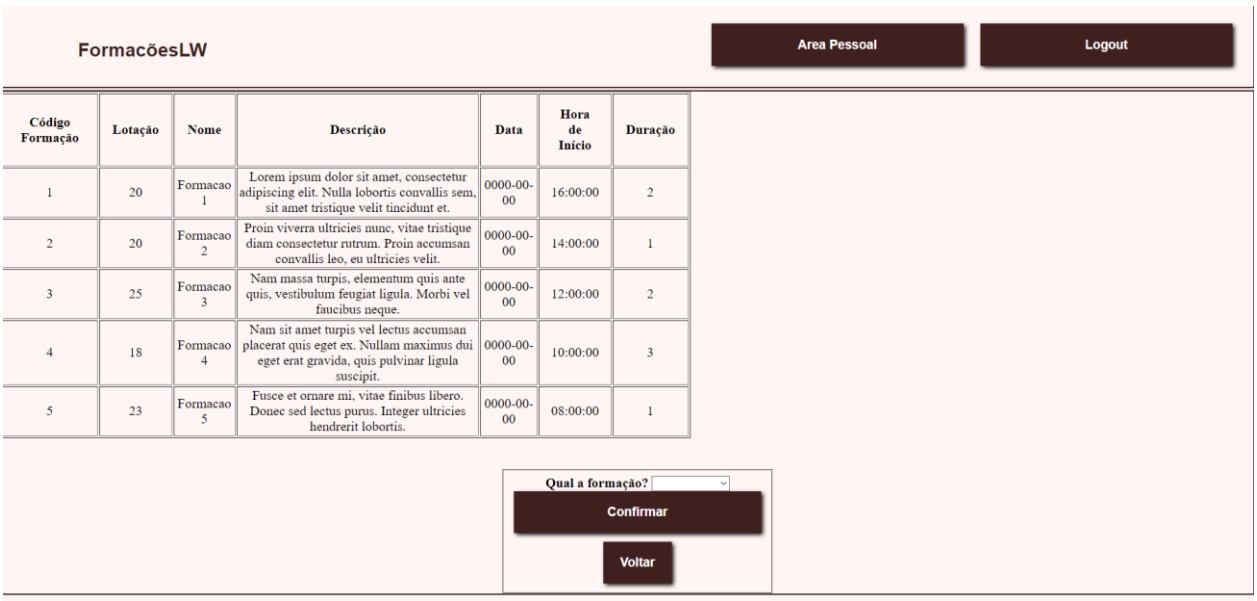
## 4.4 Dados pessoais

A página de dados pessoais contém a informação do utilizador, mais especificamente qual o seu nome de utilizador, password, número de telemóvel, email e ainda o número de inscrições feitas pelo cliente atual. O utilizador tem a opção de editar as suas informações básicas.



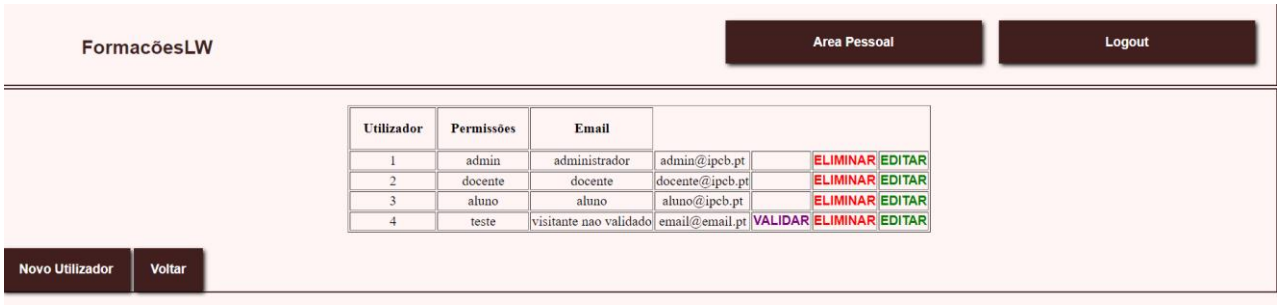

## 4.5 Página para fazer inscrições

Nesta página o aluno tem uma tabela com as informações das formações disponíveis, podendo fazer a sua inscrição.



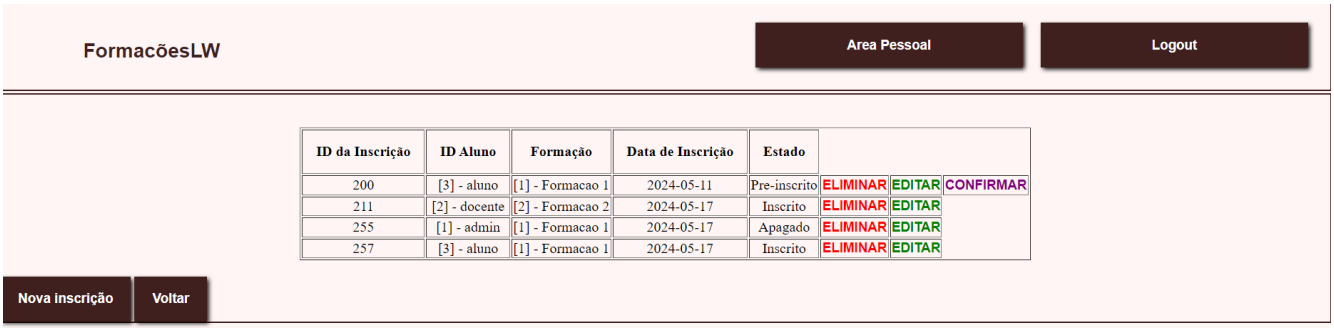
## 4.6 Página de gestão de utilizadores

Está página é utilizada exclusivamente pelo admin de maneira a conseguir gerir todos os utilizadores do website. Tem a capacidade de validar os alunos não validados e de eliminar e editar cada utilizador.



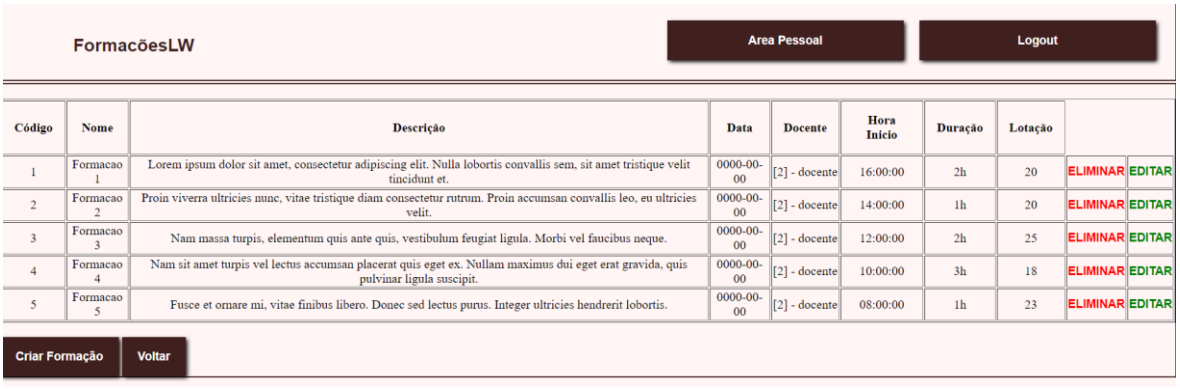

## 4.7 Página de gestão de inscrições

A gestão de inscrições é feita por todos os utilizadores. Os clientes e docentes têm acesso às suas inscrições, os admins e docentes têm acesso a todas as inscrições. Tanto os docentes como os admins têm a capacidade de confirmar, eliminar e editar inscrições, podendo ainda criar uma nova inscrição, enquanto os alunos apenas têm a capacidade de editar e eliminar.



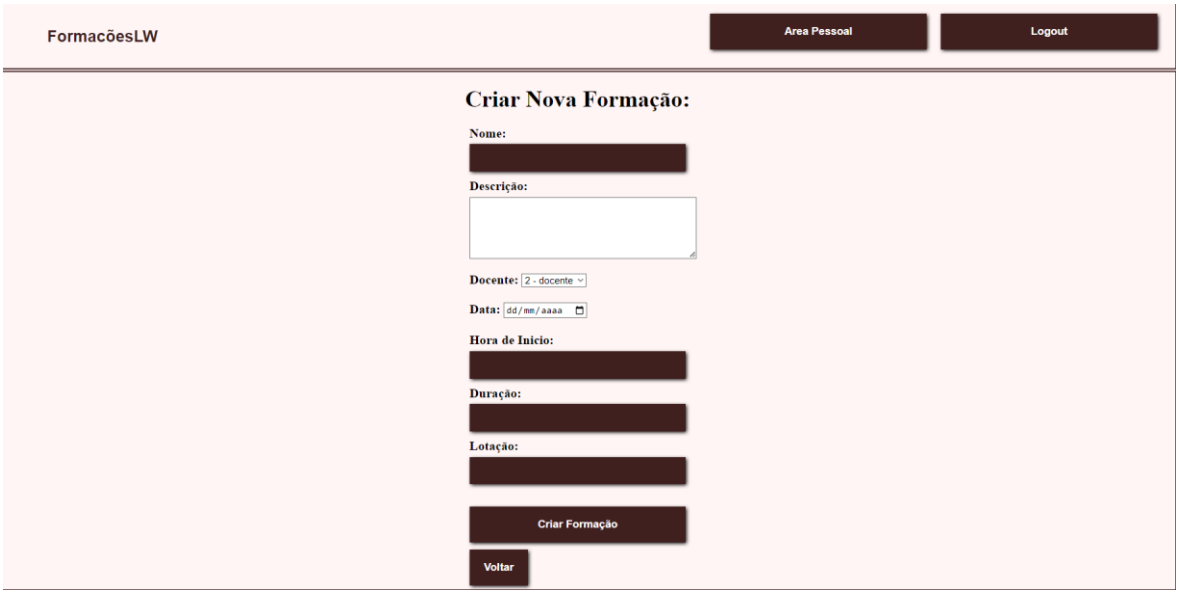
## 4.8 Página de gestão de formações

A gestão de formações é feita pelo docente e admin. Tanto os docentes como os admins têm a capacidade de eliminar e editar as formações, podendo também criar formações novas.




## 4.9 Página de criação de formações

Para criar uma formação o admin/docente pode customizar o seu nome, descrição, docente, data, hora de início, duração e lotação.



# 6. Exportação da base de dados

Para fazer as exportações das bases de dados nos vários formatos foram criados os devidos ficheiros para a sua criação. Estes ficheiros PHP, através da base de dados estruturam devidamente os dados de maneira a criarem o tipo de ficheiro pretendido. Apenas o administrador pode fazer esta exportação através da página de gestão.

**DTD**

1. Define um elemento tabela_<nome_da_tabela> que irá incluir vários elementos <tabela>.
2. Executa DESCRIBE <nome_da_tabela> para obter os nomes das colunas da tabela.
3. Define um elemento <tabela> com os devidos campos internos.
4. Define cada coluna como um elemento que contém #PCDATA.

```php
if ((!isset($_SESSION['autenticado']) || $_SESSION['autenticado'] !== true || $_SESSION['tipoUtilizador'] != "1")) {
    echo "Não estás autenticado! (Sem privilégios para aceder a esta página.)";
    header(header: "refresh:1; url=pgHomepage.php");
} else {

    $query = "SHOW TABLES";
    $result = mysqli_query(mysql: $conn, query: $query);

    $dtdContent = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";
    $dtdContent .= "<!DOCTYPE formacoesLW [\n";

    $tables = [];
    while ($row = mysqli_fetch_array(result: $result)) {
        $tables[] = $row[0];
    }

    $dtdContent .= "\t<!ELEMENT formacoesLW (" . implode(separator: ", tabela_", array: $tables) . ")>\n\n";

    foreach ($tables as $table) {
        $dtdContent .= "\t<!ELEMENT tabela_$table ($table*)>\n";
        $dtdContent .= "\t<!ELEMENT $table (";

        $query = "DESCRIBE $table";
        $columns_result = mysqli_query(mysql: $conn, query: $query);
        $columns = [];
        while ($col = mysqli_fetch_assoc(result: $columns_result)) {
            $columns[] = $col['Field'];
        }

        $dtdContent .= implode(separator: ", ", array: $columns) . ")>\n";

        foreach ($columns as $column) {
            $dtdContent .= "\t<!ELEMENT $column (#PCDATA)>\n";
        }

        $dtdContent .= "\n";
    }

    $dtdContent .= "]>";

    // Guardar o DTD num ficheiro
    $filename = "exportarDTD.dtd";
    file_put_contents(filename: $filename, data: $dtdContent);
```

Obtendo como resultando o seguinte ficheiro DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE formacoesLW [
    <!ELEMENT formacoesLW (estadoinscricao, tabela_formacao, tabela_inscricao, tabela_tipoutilizador, tabela_utilizador)>

    <!ELEMENT tabela_estadoinscricao (estadoinscricao*)>
    <!ELEMENT estadoinscricao (estado, descricao)>
    <!ELEMENT estado (#PCDATA)>
    <!ELEMENT descricao (#PCDATA)>

    <!ELEMENT tabela_formacao (formacao*)>
    <!ELEMENT formacao (codigoFormacao, lotacao, nome, descricao, data, horaInicio, duracao, docenteID)>
    <!ELEMENT codigoFormacao (#PCDATA)>
    <!ELEMENT lotacao (#PCDATA)>
    <!ELEMENT nome (#PCDATA)>
    <!ELEMENT descricao (#PCDATA)>
    <!ELEMENT data (#PCDATA)>
    <!ELEMENT horaInicio (#PCDATA)>
    <!ELEMENT duracao (#PCDATA)>
    <!ELEMENT docenteID (#PCDATA)>

    <!ELEMENT tabela_inscricao (inscricao*)>
    <!ELEMENT inscricao (idInscricao, idAluno, codigoFormacao, dataInscricao, estado)>
    <!ELEMENT idInscricao (#PCDATA)>
    <!ELEMENT idAluno (#PCDATA)>
    <!ELEMENT codigoFormacao (#PCDATA)>
    <!ELEMENT dataInscricao (#PCDATA)>
    <!ELEMENT estado (#PCDATA)>

    <!ELEMENT tabela_tipoutilizador (tipoutilizador*)>
    <!ELEMENT tipoutilizador (id, descricao)>
    <!ELEMENT id (#PCDATA)>
    <!ELEMENT descricao (#PCDATA)>

    <!ELEMENT tabela_utilizador (utilizador*)>
    <!ELEMENT utilizador (id, nomeUtilizador, password, tipoUtilizador, email)>
    <!ELEMENT id (#PCDATA)>
    <!ELEMENT nomeUtilizador (#PCDATA)>
    <!ELEMENT password (#PCDATA)>
    <!ELEMENT tipoUtilizador (#PCDATA)>
    <!ELEMENT email (#PCDATA)>

]>
```

**XML**

```php
// Obter utilizadores
$utilizadores = $dom->createElement(localName: 'tabela_utilizador');
$root->appendChild(node: $utilizadores);

$sql = "SELECT * FROM utilizador";
$utilizadoresRes = mysqli_query(mysql: $conn, query: $sql);

while ($utilizador = mysqli_fetch_assoc(result: $utilizadoresRes)) {
    $util = $dom->createElement(localName: 'utilizador');
    foreach ($utilizador as $key => $value) {
        $temp = $dom->createElement(localName: $key, value: $value);
        $util->appendChild(node: $temp);
    }
    $utilizadores->appendChild(node: $util);
}

// Obter tipos de utilizador
$tiposutilizadores = $dom->createElement(localName: 'tabela_tipoutilizador');
$root->appendChild(node: $tiposutilizadores);

$sql = "SELECT * FROM tipoutilizador";
$tiposRes = mysqli_query(mysql: $conn, query: $sql);

while ($tipoutilizador = mysqli_fetch_assoc(result: $tiposRes)) {
    $t = $dom->createElement(localName: 'tipoUtilizador');
    foreach ($tipoutilizador as $key => $value) {
        $temp = $dom->createElement(localName: $key, value: $value);
        $t->appendChild(node: $temp);
    }
    $tiposutilizadores->appendChild(node: $t);
}
```

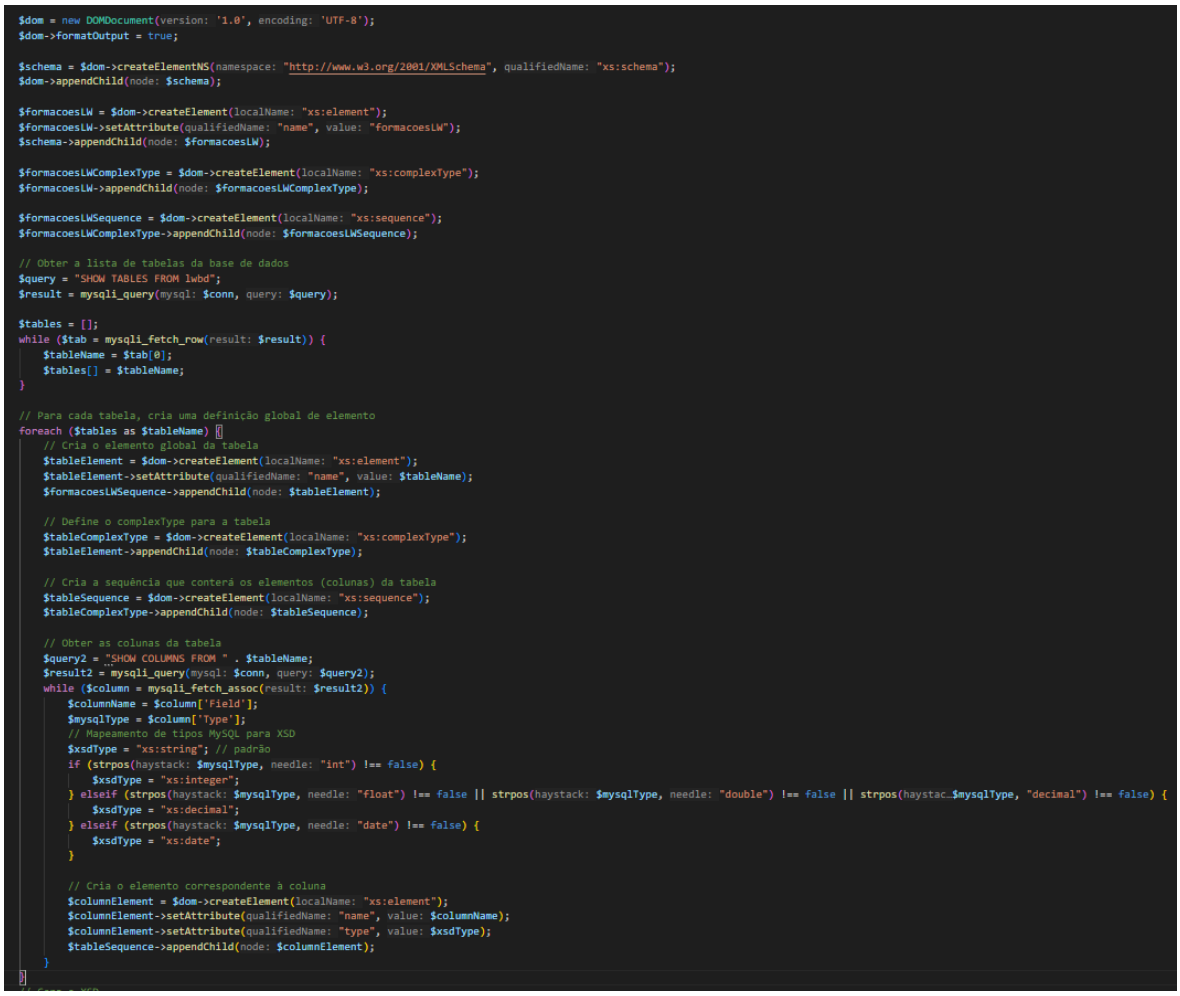
1. Cria o elemento <tabela_utilizador > e adiciona ao elemento raiz.
2. Executa a consulta SELECT * FROM utilizador para obter todos os utilizadores da base de dados.
3. Para cada registo:
    - Cria um elemento <utilizador>.
    - Para cada coluna ($key), cria um elemento XML <key> com o valor correspondente ($value).
    - Adiciona esse elemento dentro de <utilizador>.
    - Adiciona <utilizador> dentro de <tabela_utilizador>.

Este padrão é repetido para todas as outras tabelas.

Resulta da exportação em XML para a tabela de utilizadores:

```xml
<formacoesLW>
  <tabela_utilizador>
    <utilizador>
      <id>1</id>
      <nomeUtilizador>admin</nomeUtilizador>
      <password>admin</password>
      <tipoUtilizador>1</tipoUtilizador>
      <email>admin@ipcb.pt</email>
    </utilizador>
    <utilizador>
      <id>2</id>
      <nomeUtilizador>docente</nomeUtilizador>
      <password>docente</password>
      <tipoUtilizador>2</tipoUtilizador>
      <email>docente@ipcb.pt</email>
    </utilizador>
    <utilizador>
      <id>3</id>
      <nomeUtilizador>aluno</nomeUtilizador>
      <password>aluno</password>
      <tipoUtilizador>3</tipoUtilizador>
      <email>aluno@ipcb.pt</email>
    </utilizador>
```

## XSD



```php
$dom = new DOMDocument(version: '1.0', encoding: 'UTF-8');
$dom->formatOutput = true;

$schema = $dom->createElementNS(namespace: "http://www.w3.org/2001/XMLSchema", qualifiedName: "xs:schema");
$dom->appendChild(node: $schema);

$formacoesLW = $dom->createElement(localName: "xs:element");
$formacoesLW->setAttribute(qualifiedName: "name", value: "formacoesLW");
$schema->appendChild(node: $formacoesLW);

$formacoesLWComplexType = $dom->createElement(localName: "xs:complexType");
$formacoesLW->appendChild(node: $formacoesLWComplexType);

$formacoesLWSequence = $dom->createElement(localName: "xs:sequence");
$formacoesLWComplexType->appendChild(node: $formacoesLWSequence);

// Obter a lista de tabelas da base de dados
$query = "SHOW TABLES FROM lwbd";
$result = mysqli_query(mysql: $conn, query: $query);

$tables = [];
while ($tab = mysqli_fetch_row(result: $result)) {
    $tableName = $tab[0];
    $tables[] = $tableName;
}

// Para cada tabela, cria uma definição global de elemento
foreach ($tables as $tableName) {
    // Cria o elemento global da tabela
    $tableElement = $dom->createElement(localName: "xs:element");
    $tableElement->setAttribute(qualifiedName: "name", value: $tableName);
    $formacoesLWSequence->appendChild(node: $tableElement);

    // Define o complexType para a tabela
    $tableComplexType = $dom->createElement(localName: "xs:complexType");
    $tableElement->appendChild(node: $tableComplexType);

    // Cria a sequência que conterá os elementos (colunas) da tabela
    $tableSequence = $dom->createElement(localName: "xs:sequence");
    $tableComplexType->appendChild(node: $tableSequence);

    // Obter as colunas da tabela
    $query2 = "SHOW COLUMNS FROM " . $tableName;
    $result2 = mysqli_query(mysql: $conn, query: $query2);
    while ($column = mysqli_fetch_assoc(result: $result2)) {
        $columnName = $column['Field'];
        $mysqlType = $column['Type'];
        // Mapeamento de tipos MySQL para XSD
        $xsdType = "xs:string"; // padrão
        if (strpos(haystack: $mysqlType, needle: "int") !== false) {
            $xsdType = "xs:integer";
        } elseif (strpos(haystack: $mysqlType, needle: "float") !== false || strpos(haystack: $mysqlType, needle: "double") !== false || strpos(haystac…$mysqlType, "decimal") !== false) {
            $xsdType = "xs:decimal";
        } elseif (strpos(haystack: $mysqlType, needle: "date") !== false) {
            $xsdType = "xs:date";
        }

        // Cria o elemento correspondente à coluna
        $columnElement = $dom->createElement(localName: "xs:element");
        $columnElement->setAttribute(qualifiedName: "name", value: $columnName);
        $columnElement->setAttribute(qualifiedName: "type", value: $xsdType);
        $tableSequence->appendChild(node: $columnElement);
    }
}
```

É criado um elemento principal chamado <formacoesLW>, que agrupará todas as tabelas da base de dados.

Define-se um tipo complexo (xs:complexType) com uma sequência (xs:sequence), que permite que várias tabelas sejam adicionadas.

Executa-se a query SHOW TABLES FROM lwbd para listar todas as tabelas da base de dados lwbd.

São armazenados os nomes das tabelas no array $tables.

Para cada tabela:

1. Cria um elemento global (xs:element) com o nome da tabela.
2. Define um complexType (xs:complexType), pois cada tabela contém múltiplos campos.
3. Adiciona uma sequência (xs:sequence) para listar as colunas dessa tabela.

Para cada tabela, executa SHOW COLUMNS FROM nome_da_tabela para obter a estrutura de colunas.

Converte os tipos de dados do MySQL para tipos equivalentes no XSD:

- int → xs:integer
- float, double, decimal → xs:decimal
- date → xs:date
- Por padrão, tudo o resto é xs:string.

Para cada coluna:

- É criado um elemento XSD (xs:element) com o nome da coluna.
- É definido o seu tipo de dado (xs:integer, xs:string, etc.).
- É adicionada a coluna à sequência da tabela.

Resultado do ficheiro XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="formacoesLW">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="estadoinscricao">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="estado" type="xs:integer"/>
              <xs:element name="descricao" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="formacao">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="codigoFormacao" type="xs:integer"/>
              <xs:element name="lotacao" type="xs:integer"/>
              <xs:element name="nome" type="xs:string"/>
              <xs:element name="descricao" type="xs:string"/>
              <xs:element name="data" type="xs:date"/>
              <xs:element name="horaInicio" type="xs:string"/>
              <xs:element name="duracao" type="xs:integer"/>
              <xs:element name="docenteID" type="xs:integer"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="inscricao">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="idInscricao" type="xs:integer"/>
              <xs:element name="idAluno" type="xs:integer"/>
              <xs:element name="codigoFormacao" type="xs:integer"/>
              <xs:element name="dataInscricao" type="xs:date"/>
              <xs:element name="estado" type="xs:integer"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
```

# 6. Visualização em XML

Para fazer a visualização dos ficheiros HTML em XML, são criadas uma página XML e uma página XSLT. O ficheiro XML armazena os dados de maneira estruturada e o XSLT transforma esse ficheiro XML numa tabela com dados organizados, permitindo a sua visualização.

Ficheiro XML para a visualização dos utilizadores:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<utilizadores>
  <utilizador>
    <id>1</id>
    <nomeUtilizador>admin</nomeUtilizador>
    <tipoUtilizador>administrador</tipoUtilizador>
    <email>admin@ipcb.pt</email>
    <eliminar link="eliminaUtilizador.php?nomeUtilizador=admin&amp;tipoUtilizador=administrador">ELIMINAR</eliminar>
    <editar link="pgEditaUtilizador.php?id=1">EDITAR</editar>
  </utilizador>
  <utilizador>
    <id>2</id>
    <nomeUtilizador>docente</nomeUtilizador>
    <tipoUtilizador>docente</tipoUtilizador>
    <email>docente@ipcb.pt</email>
    <eliminar link="eliminaUtilizador.php?nomeUtilizador=docente&amp;tipoUtilizador=docente">ELIMINAR</eliminar>
    <editar link="pgEditaUtilizador.php?id=2">EDITAR</editar>
  </utilizador>
  <utilizador>
    <id>3</id>
    <nomeUtilizador>aluno</nomeUtilizador>
    <tipoUtilizador>aluno</tipoUtilizador>
    <email>aluno@ipcb.pt</email>
    <eliminar link="eliminaUtilizador.php?nomeUtilizador=aluno&amp;tipoUtilizador=aluno">ELIMINAR</eliminar>
    <editar link="pgEditaUtilizador.php?id=3">EDITAR</editar>
  </utilizador>
```

Ficheiro XSLT para a visualização dos utilizadores:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
    <body>
    <table border="1">
      <tr style="color: white" bgcolor="#401f1f">
        <th>ID</th>
        <th>Nome</th>
        <th>Tipo</th>
        <th>Email</th>
      </tr>
      <xsl:for-each select="utilizadores/utilizador">
      <tr>
        <td><xsl:value-of select="id"/></td>
        <td><xsl:value-of select="nomeUtilizador"/></td>
        <td><xsl:value-of select="tipoUtilizador"/></td>
        <td><xsl:value-of select="email"/></td>
        <td>
          <a style="color: purple">
            <xsl:attribute name="href">
              <xsl:value-of select="validar/@link" />
            </xsl:attribute>
            <xsl:value-of select="validar"/>
          </a>
        </td>
        <td><a style="color: red">
            <xsl:attribute name="href">
              <xsl:value-of select="eliminar/@link" />
            </xsl:attribute>
            <xsl:value-of select="eliminar"/>
          </a>
        </td>
        <td><a style="color: green">
            <xsl:attribute name="href">
              <xsl:value-of select="editar/@link" />
            </xsl:attribute>
            <xsl:value-of select="editar"/>
          </a>
        </td>
      </tr>
      </xsl:for-each>
    </table>
    </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

# 7. Conclusão

Neste trabalho, foi desenvolvida uma aplicação em PHP, HTML, MySQL, DTD, XML e XSD que permitisse a gestão de um website para gestão de cursos de formação. A aplicação abrange diferentes perfis de utilizadores: visitantes, alunos, docentes e administrador.

Os visitantes podem explorar as formações disponíveis, enquanto os alunos têm acesso a recursos adicionais, como acesso aos dados pessoas e marcação de inscrições com acesso à gestão das mesmas. Os docentes contam com ferramentas de gestão de inscrições e formações facilitando a organização do website. O administrador tem controlo total sobre a aplicação, fazendo uso de todas as funcionalidades incluindo a gestão de utilizadores, podendo também fazer a exportação da base de dados para os formatos XML, DTD e XSD.

A implementação desta aplicação visa melhorar a experiência do utilizador, oferecendo recursos personalizados para cada perfil. Além disso, procura melhorar a gestão interna do website, otimizando processos e facilitando a interação com os alunos.

Ao combinar tecnologias como PHP, HTML e MySQL, DTD, XML e XSD o trabalho resultou numa solução abrangente e eficiente para a gestão do website.