

Lisboa, 30 de maio de 2023

Ricardo da Cruz Coimbra

200059, TGPSI20/02

Técnico de Gestão e Programação de Sistemas Informáticos

Relatório

Loja de Roupa - RickyShop

2023

Índice

[Lista de Programas iv](#_Toc137046566)

[Fundação Técnica-Científica v](#_Toc137046567)

[IDE (Integrated Development Environment) v](#_Toc137046568)

[Visual Studio v](#_Toc137046569)

[C# vi](#_Toc137046570)

[C# Código vii](#_Toc137046571)

[Variáveis vii](#_Toc137046572)

[Declaração de variáveis: viii](#_Toc137046573)

[Operações ix](#_Toc137046574)

[Funções x](#_Toc137046575)

[Classes xi](#_Toc137046576)

[Ciclo xii](#_Toc137046577)

[Regex xiv](#_Toc137046578)

[Base de Dados xv](#_Toc137046579)

[Tabelas xv](#_Toc137046580)

[Propriedades dos Campos xvi](#_Toc137046581)

[Relações Entre Tabelas xvii](#_Toc137046582)

[SQL xviii](#_Toc137046583)

[SQL Comandos xix](#_Toc137046584)

[SELECT xix](#_Toc137046585)

[INSERT xix](#_Toc137046586)

[UPDATE xix](#_Toc137046587)

[DELETE xix](#_Toc137046588)

[INNER JOIN xix](#_Toc137046589)

[STORED PROCEDURE xix](#_Toc137046590)

[HTML xx](#_Toc137046591)

[HTML Código xxi](#_Toc137046592)

[<h> xxii](#_Toc137046593)

[<tittle> xxii](#_Toc137046594)

[href xxii](#_Toc137046595)

[<table> xxiii](#_Toc137046596)

[CSS xxiv](#_Toc137046597)

[JavaScript xxv](#_Toc137046598)

[Bootstrap xxvi](#_Toc137046599)

[MVC xxvii](#_Toc137046600)

[Model (Modelo) xxvii](#_Toc137046601)

[View (Visualização) xxvii](#_Toc137046602)

[Controller (Controlador) xxvii](#_Toc137046603)

[jQuery xxix](#_Toc137046604)

[GitHub xxx](#_Toc137046605)

[Estágio xxxi](#_Toc137046606)

[Empresa de estágio xxxi](#_Toc137046607)

[Atividades durante o estágio xxxi](#_Toc137046608)

[Análise critica xxxii](#_Toc137046609)

[Introdução 1](#_Toc137046610)

[Projeto 2](#_Toc137046611)

**Lista de Figuras**

[Figura 1.1 - Logo Visual Studio 5](file:///C:\Users\pitoc\Documents\GitHub\RickyShop\Relatorio.docx#_Toc137044952)

[Figura 1.2 - Logo C# 7](#_Toc137044953)

[Figura 2.1 – Tabela Exemplo Cliente 15](#_Toc137044954)

[Figura 3.1 - Base de Dados 17](#_Toc137044955)

[Figura 4.1 - Logo SQL Server Management Studio 18](#_Toc137044956)

[Figura 5.1 – Logo HTML 20](#_Toc137044957)

[Figura 6.1 - Exemplo <h1> 22](#_Toc137044958)

[Figura 7.1 - Exemplo <h6> 22](#_Toc137044959)

[Figura 8.1 - Logo CSS 24](#_Toc137044960)

[Figura 9.1 - Logo JavaScript 25](#_Toc137044961)

[Figura 10.1 – Logo Bootstrap 26](#_Toc137044962)

[Figura 11.1 - MVC 28](#_Toc137044963)

[Figura 12.1 - Logo jQuery 29](#_Toc137044964)

[Figura 13.1 - Logo GitHub 30](#_Toc137044965)

[Figura 14.1 – Logo Centrality 31](#_Toc137044966)

Lista de Programas

# Fundação Técnica-Científica

## IDE (Integrated Development Environment)

Um IDE (Integrated Development Environment), ou Ambiente de Desenvolvimento Integrado, é uma ferramenta de software que fornece um conjunto completo de recursos para desenvolver, depurar e testar aplicações de software. O IDE é projetado para oferecer um ambiente de trabalho conveniente e eficiente para os programadores, agrupando várias funcionalidades em uma única interface.

## Visual Studio

O Visual Studio é um ambiente de desenvolvimento integrado (IDE) criado pela Microsoft, que oferece um conjunto de ferramentas e recursos para desenvolvedores de software criarem, testarem e implantarem aplicações para várias plataformas, incluindo desktop, web, dispositivos móveis e nuvem.

O Visual Studio é composto por vários componentes, como o editor de código, o compilador, o gerenciador de projetos, o designer de interface do utilizador e entre outros. Esses componentes trabalham juntos para fornecer uma experiência de desenvolvimento integrada e produtiva, permitindo que os desenvolvedores escrevam código com eficiência.

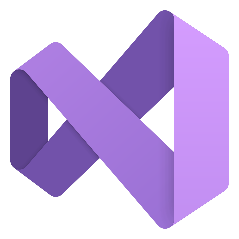
O Visual Studio é amplamente utilizado por desenvolvedores de software em todo o mundo para criar aplicações desktop, jogos, aplicações para dispositivos móveis, aplicações web, serviços em nuvem e muito mais. Oferece suporte a uma ampla variedade de linguagens de programação, incluindo C++, C#, Visual Basic, JavaScript, Python e muitas outras.

Figura .1 - Logo Visual Studio

## C#

C# é uma linguagem de programação moderna e orientada a objetos desenvolvida pela Microsoft. Faz parte da plataforma .NET, que é um conjunto de tecnologias usadas para a criação de aplicações para Windows, Web, dispositivos móveis e outras plataformas.

C# foi projetada para ser uma linguagem versátil e fácil de usar. Possui uma sintaxe semelhante a outras linguagens populares, como C, C++ e Java, o que facilita a transição para C#.

Uma das principais características do C# é o suporte à orientação a objetos. Isso significa que a linguagem se baseia no conceito de classes e objetos. Os desenvolvedores podem criar classes para representar entidades e objetos da vida real, e então criar instâncias dessas classes para manipular e interagir com os dados.

C# também possui uma ampla biblioteca padrão, conhecida como Biblioteca de Classes do .NET (ou .NET Framework), que fornece uma extensa gama de classes e métodos que facilitam o desenvolvimento de diferentes tipos de aplicações. Essa biblioteca oferece recursos para manipulação de ficheiros, acesso a base de dados, criação de interfaces gráficas, comunicação em rede e entre outras.

Além disso, C# possui suporte integrado para programação assíncrona, o que permite escrever código que pode executar tarefas em segundo plano, sem bloquear a execução principal do programa. Isso é especialmente útil para criar aplicações responsivos que lidam com operações intensivas em recursos ou operações de entrada e saída.

Embora inicialmente tenha sido desenvolvida para a plataforma Windows, a Microsoft introduziu o .NET Core, uma versão multiplataforma do .NET Framework. Com o .NET Core, é possível criar aplicações C# que podem ser executados em sistemas operacionais como Windows, Linux e macOS.

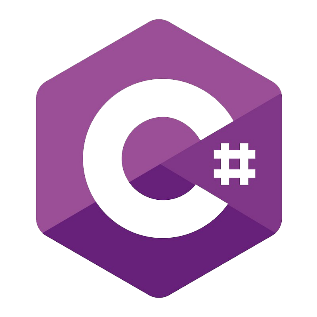


Figura 1. - Logo C#

## C# Código

O código em C# é escrito em arquivos com a extensão ".cs". Esses arquivos contêm instruções que dizem ao computador o que fazer. O código em C# é composto por várias partes, incluindo declarações, expressões, estruturas de controle e chamadas de função.

### Variáveis

Em C#, as variáveis são elementos fundamentais usados para armazenar e manipular dados dentro de um programa. São usadas para armazenar valores temporariamente na memória durante a execução do código.

Em C#, existem vários tipos de variáveis que podem ser usados para armazenar diferentes tipos de dados.

* **Tipos Numéricos:**
  + ‘*int’*: Representa números inteiros.
  + ‘*float’*: Representa números com casas decimais com precisão simples.
  + ‘*Double’*: Representa números com casas decimais com maior precisão.
  + ‘*Decimal’*: Representa números com casas decimais com precisão decimal maior.

Esses tipos de variáveis são usados para armazenar valores numéricos inteiros ou com casas decimais, e são úteis em várias situações, como contar objetos, armazenar IDs, guardar medidas, valores monetários ou cálculos científicos.

* **Tipo Booleano:**
  + ‘*bool*’: Representa valores booleanos verdadeiro (true) ou falso (false).

As variáveis booleanas são usadas para armazenar estados lógicos, como habilitar ou desabilitar uma funcionalidade, tomar decisões ou avaliar condições.

* **Tipo de Caracter**:
  + ‘*char*’: Representa um único caracter, como 'a', '1' ou '@'.
  + *‘string’*: Representa uma sequência de caracteres.

As variáveis de caracter são usadas para armazenar caracteres individuais, ou palavras e textos.

Declaração de variáveis: Para criar uma variável em C#, precisamos declará-la, especificando o seu tipo e o nome. Estes são os exemplos para declarar as variáveis dos tipos referidos em cima.

int idade = 25;

float altura = 1.75f;

double saldoBancario = 2500.50;

decimal precoProduto = 19.99m;

bool estaChovendo = true;

bool temAcesso = false;

char letra = 'A';

string mensagem = "Olá, mundo!";

### Operações

Em C#, você pode fazer várias operações, como operações aritméticas, operações comparação entre outras.

**Operações Aritméticas**

int num1 = 10;

int num2 = 5;

int soma = num1 + num2;

int subtracao = num1 - num2;

int multiplicacao = num1 \* num2;

double divisao = num1 / (double)num2;

**Operações de Comparação**

int idade = 25;

if (idade >= 18 && idade <= 65) {

Console.WriteLine("Você é um adulto.");

}

else if (idade < 18) {

Console.WriteLine("Você é menor de idade.");

}

else

{

Console.WriteLine("Você é um idoso.");

}

### Funções

Em C#, uma função é uma secção de código que executa uma tarefa específica e pode ser chamada em outras partes do programa. As funções são úteis porque permitem que você divida um programa em partes menores e reutilize o código em diferentes partes do programa.

// Definição de função para calcular o quadrado de um número

int CalcularQuadrado(int numero) {

int quadrado = numero \* numero;

return quadrado;

}

Neste exemplo, a função “CalcularQuadrado” recebe um número inteiro como parâmetro e retorna o quadrado desse número. O int atrás do “CalcularQuadrado” indica que a função retorna um valor inteiro. O corpo da função contém a lógica para calcular o quadrado do número e guardá-lo na variável quadrado, que é retornada.

### Classes

Em C#, uma classe é uma estrutura que define um objeto personalizado que contém dados. As classes são a base da programação orientada a objetos em C# e são usadas para modelar objetos.

class Turma

{

// Propriedades da classe

public string Curso;

public int Ano;

public int qtd\_Alunos;

// Método da classe

public void MostrarInformacoes()

{

Console.WriteLine("Curso: " + Curso);

Console.WriteLine("Ano: " + Ano);

Console.WriteLine("Quantidade Alunos: " + qtd\_Alunos);

}

}

Neste exemplo, a classe “Turma” define um objeto personalizado que possui propriedades Curso, Ano e qtd\_Alunos. Essas propriedades são públicas, o que significa que podem ser acedidas fora da classe. A classe também possui um método chamado "MostrarInformacoes” que imprime informações sobre a turma no console.

Para usar essa classe em outras partes do programa, pode criar uma instância da classe, preencher as propriedades e chamar o método.

Turma minhaTurma = new Turma();

minhaTurma.Curso = "TGPSI";

minhaTurma.Ano = "2020";

minhaTurma.qtd\_Alunos = 21;

minhaTurma.MostrarInformacoes();

Neste exemplo, uma instância da classe “Turma” é criada. As propriedades da instância são definidas para representar uma Turma de TGPSI de 2020 com 21 alunos. O método “MostrarInformacoes” é então chamado para exibir as informações da turma no console.

### Ciclo

Em C#, os ciclos, também conhecidos como loops, são utilizados para executar um bloco de código repetidamente com base em uma condição ou um número pré-determinado de vezes. Eles permitem automatizar tarefas repetitivas e iterar sobre coleções de dados.

Existem quatro tipos principais de ciclos em C#: for, while, do-while e foreach.

**For**: O ciclo for é usado quando se sabe o número de iterações a serem realizadas. Ele consiste em três partes: a inicialização, a condição de continuação e a atualização.

for (int i = 1; i <= 5; i++)

{

Console.WriteLine(i);

}

Nesse exemplo, a variável i é inicializada com o valor 1. A condição de continuação é i <= 5, o que significa que o ciclo continuará enquanto i for menor ou igual a 5. A cada ação, o valor de i é incrementado em 1. O resultado será a impressão na consola dos números de 1 a 5.

**While**: O ciclo while é usado quando a condição de repetição não é conhecida antecipadamente e precisa ser verificada antes de cada ação. Ele continua repetindo até que a condição seja falsa ou seja quando for verdade o ciclo termina. Aqui está um exemplo de um ciclo while que exibe os números de 1 a 5:

int i = 1;

while (i <= 5)

{

Console.WriteLine(i);

i++;

}

**do-while:** O ciclo do-while é semelhante ao ciclo while, mas a condição de repetição é verificada após a ação. Isso garante que o bloco seja executado pelo menos uma vez, mesmo se a condição inicialmente for falsa. Aqui está um exemplo de um ciclo do-while que lê números do usuário até que seja digitado um número negativo:

int numero;

do

{

Console.WriteLine("Digite um número positivo: ");

numero = int.Parse(Console.ReadLine());

Console.WriteLine("Número digitado: " + numero);

} while (numero >= 0);

Nesse exemplo, o bloco de código dentro do ciclo é executado pelo menos uma vez. Ele solicita ao usuário que digite um número positivo e exibe o número digitado. O ciclo continua enquanto o número digitado for maior ou igual a zero. Quando o usuário digitar um número negativo, o ciclo será interrompido.

**Foreach**: O foreach é uma estrutura de controlo em C# que permite iterar facilmente sobre os elementos de uma coleção. Ele simplifica o processo de percorrer os dados de uma coleção, fornecendo um ciclo de iteração mais legível e conveniente.

foreach (tipo item in colecao)

{

// bloco de código a ser executado para cada item

}

Aqui está uma explicação detalhada do **foreach** com exemplos:

* **tipo** é o tipo de dados dos elementos da coleção.
* i**tem** é a variável que receberá cada elemento da coleção durante a iteração.
* **colecao** é a coleção de elementos a serem iterados.

int[] numeros = { 1, 2, 3, 4, 5 };

foreach (int numero in numeros)

{

Console.WriteLine(numero);

}

Nesse exemplo, o ***foreach*** percorre cada elemento do *array* números e atribui o valor atual à variável “numero”. A cada iteração, o bloco de código dentro do ***foreach*** é executado, no qual estamos apenas imprimindo o valor de “numero” no console. Também é importante saber que a procura executada pelo ***foreach*** não é uma procura sequencial, assim sendo neste exemplo ele pode mostrar primeiro o valor 5 e depois o valor 1.

Regex  
*Regex*, também conhecido como expressão regular, é uma sequência de caracteres que define um padrão de busca em um texto. É uma ferramenta poderosa para manipulação e processamento de *strings*, permitindo buscar, validar, substituir e extrair informações com base em padrões específicos.

Os padrões de *regex* são construídos com uma combinação de caracteres literais e metacaracteres. Os caracteres literais correspondem exatamente aos caracteres que aparecem na *string*, enquanto os metacaracteres possuem um significado especial e são utilizados para representar classes de caracteres, quantificadores, posições, entre outros.

Regex regex = new Regex(@"^\d{4}(-\d{3})?$");

if (regex.IsMatch(codPostal) == true)

{

//Código a executar depois da validação

}

Neste bloco de código está um exemplo do uso de uma ***Regex***, começamos por declarar uma variável do tipo ***Regex*** e de seguida atribuímos a sua validação, que neste caso é de um código-postal.

Logo a seguir, verificamos se os dados da variável **codPostal** cumprem os requisitos pedidos na *regex*.

## Base de Dados

A base de dados é um conjunto de informações organizadas e estruturadas de forma a permitir o armazenamento, a gestão e a recuperação de dados de forma eficiente. É constituída por um conjunto de tabelas, que representam entidades e relações entre elas, e que são interligadas por chaves primárias e estrangeiras. A base de dados é utilizada em diversos tipos de aplicações e empresas, como sistemas de gestão empresarial, sistemas de informação de saúde, sistemas de controlo de stock, sistemas de gerenciamento de bibliotecas, entre outros. Permite que as informações sejam armazenadas de forma organizada e que sejam recuperadas de maneira rápida e eficiente.

### Tabelas

Uma tabela numa base de dados é uma coleção organizada de dados em forma de linhas e colunas. As tabelas são usadas para armazenar informações sobre entidades específicas, como clientes, produtos, pedidos, entre outros. Cada linha numa tabela representa um registo de dados, e cada coluna representa um campo de dados específico.

Por exemplo, uma tabela de clientes pode ter as seguintes colunas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Primeiro Nome | Último Nome | Email | Contacto |
| 1 | Ricardo | Coimbra | [I200059@inete.net](mailto:I200059@inete.net) | 968574123 |
| 2 | Joaquim | Aníbal | [I200080@inete.net](mailto:I200080@inete.net) | 921456378 |

Figura 2.1 – Tabela Exemplo Cliente

Nesse exemplo, cada linha representa um registo de um cliente e cada coluna representa um campo de dados específico, como o **Primeiro Nome**, **Segundo Nome**, **Email** e **Contacto**. O **ID** é um campo de chave primária que garante a exclusividade de cada registo.

As tabelas na base de dados são usadas para armazenar e organizar grandes quantidades de informações de forma estruturada.

Propriedades dos Campos

Os campos numa tabela de base de dados podem ter diversas propriedades, dependendo do tipo de dado e do objetivo da tabela.

Estes são alguns exemplos de propriedades que podem ser aplicadas aos campos de uma tabela:

**Tipo de dado:** Cada campo numa tabela deve ter um tipo de dado associado, que determina que tipo de valor pode ser armazenado nesse campo. Por exemplo, um campo pode ser do tipo *varchar*, *int*, *date*, entre outros.

**Tamanho:** O tamanho de um campo determina o número máximo de caracteres ou valores que podem ser armazenados nele. Por exemplo, um campo de texto pode ter um tamanho máximo de 50 caracteres.

**Valor padrão:** O valor padrão de um campo é o valor que será usado caso nenhum outro valor seja especificado. Por exemplo, um campo de data pode ter um valor padrão da data de hoje.

**Chave primária:** Um campo pode ser definido como chave primária, que é um identificador único para cada registo na tabela. A chave primária é usada para garantir que não haja registo duplicados na tabela. Uma tabela tem de ter sempre uma chave primária, normalmente é um ID.

**Chave estrangeira:** Um campo pode ser definido como chave estrangeira, que é um campo que faz referência a uma chave primária em outra tabela. A chave estrangeira é usada para estabelecer relações entre tabelas.

**Chave candidata:** Uma chave candidata é um conjunto de um ou mais campos numa tabela que podem ser usados como chave primária. Uma chave candidata é chamada assim porque, embora possa ser usada como chave primária, não é a chave primária atual da tabela. Temos o exemplo na tabela acima em que o Email e o Contacto podiam ser chaves primárias.

### Relações Entre Tabelas

As tabelas em uma base de dados podem estar relacionadas de várias maneiras.

**Relação um para um:** Nessa relação, cada registo de uma tabela está relacionado a apenas um registo noutra tabela. Por exemplo, uma tabela de funcionários pode estar relacionada a uma tabela de cargos. Cada funcionário tem apenas um cargo, e cada cargo pertence a apenas um funcionário.

**Relação um para muitos:** Nessa relação, cada registo de uma tabela está relacionado a vários registos noutra tabela. Por exemplo, uma tabela de clientes pode estar relacionada a uma tabela de pedidos. Cada cliente pode fazer vários pedidos, mas cada pedido pertence apenas a um cliente.

**Relação muitos para muitos:** Nessa relação, vários registos em uma tabela estão relacionados a vários registos noutra tabela. Para estabelecer essa relação, é necessário criar uma tabela de relação, que relaciona os registos das outras duas tabelas. Por exemplo, uma tabela de produtos pode estar relacionada a uma tabela de pedidos. Como um pedido pode conter vários produtos, e um produto pode estar em vários pedidos, é necessário criar uma tabela de relação que relacione os produtos aos pedidos.

**Relação de chave estrangeira:** Uma tabela pode ter uma coluna que é uma chave estrangeira, que faz referência a uma coluna em outra tabela. Essa relação é usada para garantir a integridade dos dados, para que não haja registos em uma tabela que não correspondam a registos em outra tabela. Por exemplo, uma tabela de pedidos pode ter uma coluna **ID\_Cliente** que é uma chave estrangeira para a tabela de clientes.

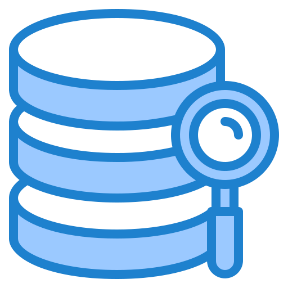


Figura 3.1 - Base de Dados

## SQL

SQL (*Structured Query Language*) é uma linguagem de programação utilizada para gerenciar e manipular base de dados relacionais. Ela foi desenvolvida na década de 1970 e é utilizada em diversos sistemas de gerenciamento de base de dados, como Oracle, MySQL, Microsoft SQL Server e PostgreSQL.

O SQL permite que os usuários criem, modifiquem e consultem dados armazenados em uma base de dados. Por meio de comandos específicos, é possível criar tabelas para armazenar dados, inserir, atualizar ou excluir registos nessas tabelas, bem como realizar consultas para obter informações específicas. Além disso, o SQL permite a definição de restrições de integridade para garantir a consistência dos dados.

O SQL é amplamente utilizado em aplicações empresariais para gerenciar dados, desde sistemas de gerenciamento de stock até sistemas de gerenciamento de recursos humanos. Além disso, é utilizado em sistemas de suporte à decisão, como *business intelligence*, onde os utilizadores podem criar consultas complexas para analisar grandes quantidades de dados.

O SQL é uma linguagem padrão, ou seja, a maioria dos comandos e sintaxes são os mesmos em diferentes sistemas de gerenciamento de base de dados. Isso torna a linguagem portátil e amplamente adotada em todo o mundo.



Figura 4.1 - Logo SQL Server Management Studio

## **SQL Comandos**

Existem vários tipos de comandos no SQL que são usados para aceder e  
manipular dados numa base de dados. Aqui estão alguns exemplos de tipos de  
comandos SQL:

SELECT**:** Usado para buscar dados de uma ou mais tabelas numa base de  
dados. Por exemplo:

SELECT \* FROM clientes

INSERT**:** Usado para inserir novos dados numa tabela. Por exemplo:

INSERT INTO clientes (nome, idade, cidade) VALUES ('João', 30, ‘Lisboa’)

UPDATE**:** Usado para atualizar dados existentes numa tabela. Por exemplo:

UPDATE clientes SET idade = 31 WHERE nome = 'João'

DELETE**:** Usado para remover dados de uma tabela. Por exemplo:

DELETE FROM clientes WHERE nome = 'João'

INNER JOIN**:** Usado para combinar dados de duas ou mais tabelas com base  
numa coluna em comum. Por exemplo:

SELECT clientes.nome, pedidos.numero FROM clientes JOIN pedidos ON ID\_Cliente = pedidos.ID\_Cliente

STORED PROCEDURE: Uma *Stored Procedure* é uma rotina armazenada em  
uma base de dados que pode ser chamada por aplicações para executar tarefas  
específicas. É um bloco de código SQL que é guardado na base de dados e pode  
ser executado posteriormente quando for necessário.

CREATE PROCEDURE SelectClientes  
AS  
SELECT \* FROM Clientes;  
GO

## HTML

HTML (*Hypertext Markup Language*) é uma linguagem de marcação utilizada

para criar conteúdo para a página web. Ela foi desenvolvida na década de 1990

e é a base para a criação de páginas web.

O HTML consiste em uma série de *tags* que indicam ao navegador como exibir

o conteúdo na página. Essas *tags* incluem elementos como “**<!DOCTYPE html>**”

“**<html>**”, “**<title>**”, “**<head>**”, “**<meta/>**” e “**<body>**”. O HTML permite

que os desenvolvedores criam páginas web estruturadas e organizadas, além

de fornecer recursos para tornar o conteúdo mais interativo e acessível.

Os desenvolvedores web usam o HTML para criar páginas web que podem ser

acedidas por meio de um navegador da web. O HTML é uma linguagem de

marcação de texto simples que é fácil de aprender e usar. Ela é usada em

conjunto com outras tecnologias web, como CSS (*Cascading Style Sheets*) e

JavaScript, para criar páginas web interativas e dinâmicas.

O HTML é uma tecnologia essencial para a criação de páginas web. Ele permite

que os desenvolvedores criem páginas estruturadas e bem organizadas que

possam ser exibidas em diferentes navegadores e dispositivos.



Figura 5.1 – Logo HTML

## **HTML Código**

O HTML é composto por uma série de elementos e *tags* que são usados para definir a estrutura e o conteúdo da página. Cada *tag* começa com “<” e termina com “>”. As *tags* podem ter atributos que definem propriedades adicionais da *tag*, como o ID ou a classe.

Os elementos HTML podem ser usados para exibir texto, imagens, vídeos, formulários e outros tipos de conteúdo. Eles também podem ser usados para criar links para outras páginas da web, criar tabelas e listas, e estilizar o conteúdo usando CSS (*Cascading Style Sheets*).

<!DOCTYPE html>

<html>

<head>

<title>Título da página</title>

</head>

<body>

<h1>Título principal</h1>

<p>Este é um parágrafo de texto.</p>

</body>

</html>

Neste exemplo, o código HTML define o título da página com a tag **<title>** e inclui um cabeçalho principal com a tag **<h1>** e um parágrafo de texto com a tag **<p>**. O código também inclui as tags **<head>** e **<body>**, que são seções importantes em todas as páginas HTML.

<h>**:** A tag <h> é usada em HTML para definir cabeçalhos ou títulos de uma página. Existem seis níveis de cabeçalhos, de **<h1>** a **<h6>**, cada um com um tamanho de fonte diferente.

<h1>Este é um cabeçalho de nível 1</h1>

Uma imagem com Tipo de letra, texto, captura de ecrã, Gráficos

Descrição gerada automaticamente

Figura 6.1 - Exemplo <h1>

<h6>Este é um cabeçalho de nível 6</h6>



Figura 7.1 - Exemplo <h6>

<tittle>**:** A *tag* **<title>** é uma *tag* HTML usada para definir o título de uma página da web. O título é exibido na barra de título do navegador e é uma das informações mais importantes para os motores de busca, pois ajuda a identificar e descrever o conteúdo da página.

<title> Título da Página </title>

href**:** O atributo **href** especifica o URL de destino para o link ou o caminho. Ele pode ser usado para criar links para outras páginas, imagens, arquivos PDF, vídeos ou para ficheiros que estão no projeto como o CSS.

<a [href="https://www.exemplo.com">Link](http://www.exemplo.com/) para site externo</a>

<link rel="stylesheet" type="text/css" href="style.css">

<table>**:** A *tag* **<table>** é usada para criar a tabela e a tag **<tr>** é usada para definir cada linha da tabela. A primeira linha da tabela é normalmente usada para definir os cabeçalhos da tabela, que são definidos usando a tag **<th>**. As células da tabela são definidas usando a tag **<td>**.

<table>

<tr>

<th>Nome</th>

<th>Idade</th>

<th>Curso</th>

</tr>

<tr>

<td>Ricardo</td>

<td>17</td>

<td>TGPSI</td>

</tr>

<tr>

<td>José</td>

<td>18</td>

<td>TSJ</td>

</tr>

</table>

## CSS

CSS (*Cascading Style Sheets*) é uma linguagem de marcação usada para definir a aparência e o layout de páginas da web. O CSS separa a apresentação visual do conteúdo da página, permitindo que os desenvolvedores criem páginas mais atraentes e acessíveis.

O CSS consiste em um conjunto de regras que definem como os elementos da página devem ser exibidos. Essas regras são escritas em um arquivo separado do HTML, que descreve a estrutura da página. O CSS é usado para definir o estilo dos elementos HTML, como cores, fontes, margens, espaçamentos, posicionamentos e animações.

As regras podem ser aplicadas diretamente aos elementos HTML, ou podem ser agrupadas em classes ou IDs para serem aplicadas a vários elementos de uma vez.

O CSS é usado para melhorar a experiência do utilizador em páginas web, tornando-as mais atraentes, legíveis e acessíveis. Ele permite que os desenvolvedores criem layouts flexíveis que se adaptam a diferentes dispositivos e tamanhos de ecrã. Também ajuda a melhorar a acessibilidade da página, permitindo que os utilizadores personalizem a aparência da página para atender às suas necessidades individuais.

Além disso, o CSS pode ser usado em conjunto com outras linguagens de programação, como **JavaScript**, para criar interatividade e animação em páginas da web.



Figura 8.1 - Logo CSS

## JavaScript

JavaScript é uma linguagem de programação de alto nível, interpretada e orientada a objetos, amplamente utilizada em desenvolvimento web para criar páginas dinâmicas e interativas.

JavaScript consiste em uma série de instruções que são executadas pelo navegador do utilizador. As instruções são escritas em um arquivo separado do HTML e CSS, e podem ser incluídas diretamente na página HTML ou em um arquivo externo. JavaScript é uma linguagem de programação de “alto nível”, o que significa que ela é mais fácil de ler e escrever do que outras linguagens de programação de “baixo nível”, como o C++.

JavaScript é usado para criar páginas da web mais interativas e dinâmicas. Ele permite que os desenvolvedores adicionem recursos como animações, janelas pop-up, menus de navegação e muito mais.

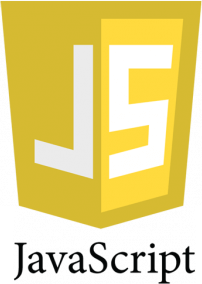


Figura 9.1 - Logo JavaScript

## Bootstrap

Bootstrap é um *framework* (conjunto de ferramentas e recursos) de *design*

flexível e de *front-end* para desenvolvimento de páginas da web.

O Bootstrap é um dos *frameworks* mais populares para desenvolvimento web, e é usado por milhões de desenvolvedores em todo o mundo.

O Bootstrap consiste em um conjunto de arquivos **CSS**, **JavaScript** e fontes, que podem ser incorporados em um projeto de página da web para criar uma estrutura de layout consistente e flexível. Ele oferece uma ampla variedade de componentes e estilos pré-construídos, como botões, menus de navegação, formulários, ícones e muito mais.

O Bootstrap é usado para acelerar o processo de desenvolvimento de páginas da web, permitindo que os desenvolvedores se concentrem mais no conteúdo e menos na aparência. Ele ajuda a garantir que um site seja consistente e responsivo em diferentes dispositivos e tamanhos de tela, melhorando a experiência do usuário.

Além disso, o Bootstrap é compatível com uma ampla variedade de navegadores e plataformas, o que o torna uma escolha popular para desenvolvimento web. No geral, o Bootstrap é uma ferramenta valiosa para qualquer desenvolvedora web que deseja criar páginas da web mais eficientes, flexível e visualmente atraentes.



Figura 10.1 – Logo Bootstrap

## MVC

MVC é uma sigla que significa Model-View-Controller, que é um padrão de arquitetura de software geralmente usado em desenvolvimento de aplicações web.

O MVC é composto por três componentes principais:

Model (Modelo) - Representa a camada de dados da aplicação, que lida com a manipulação e armazenamento dos dados. É responsável por validar e atualizar os dados da aplicação.

O ***Model*** é responsável por representar a camada de dados da aplicação e é usado pelo ***Controller*** para buscar e atualizar os dados da Base de dados. Quando o ***Model*** é atualizado, os dados apresentados na ***View*** são alterados.

View (Visualização) - Representa a camada de apresentação da aplicação, que lida com a exibição dos dados para o utilizador. A ***View*** recebe os dados do ***Model*** através do ***Controller***. O ***Controller*** solicita os dados relevantes ao ***Model*** e envia-os para a ***View*** para mostrar ao utilizador. A ***View*** não faz alterações nos dados, apenas mostra-os ao utilizador.

Controller (Controlador) - ***Controller*** é a camada que lida com as solicitações de entrada feitas pelos o utlizador, realiza as ações necessárias e devolve a resposta correspondente á ***View***. É responsável por intermediar a camada da ***View*** e a camada do ***Model***.

Dentro do ***Controller*** existe os ***ActionResult,*** que é utilizado no ***Controller*** para representar o resultado de uma ação (*action*) executada pelo utilizador.

O ActionResult é uma classe que encapsula as informações a serem retornadas pelo ***Controller*** para a ***View***. Ele pode conter diferentes tipos de dados, como objetos, coleções, redirecionamentos, erros, entre outros. O objetivo principal do ***ActionResult*** é fornecer uma forma genérica de representar o resultado de uma ação.

O ***Model*** lida com a lógica e os dados do sistema, a ***View*** lida com a apresentação visual dos dados e o ***Controller*** coordena as interações entre eles. Essa separação de responsabilidades é uma das principais vantagens da arquitetura MVC facilitando a organização, a proteção e a manutenção do projeto.

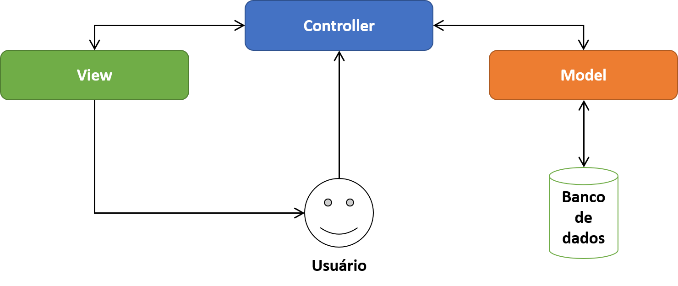


Figura 11.1 - MVC

## jQuery

*jQuery* é uma biblioteca de JavaScript que simplifica a manipulação de elementos HTML, o acesso ao *Document* *Object* *Model* (DOM) e a criação de animações e efeitos em páginas web. Com **jQuery**, é possível escrever menos código e fazer mais em menos tempo, além de ter suporte a diversos navegadores.

// Selecionar um elemento pelo ID

$('#myElement');

// Selecionar um elemento pela classe

$('.myClass');

// Selecionar vários elementos pela tag

$('div');

Este exemplo é uma parte de código JavaScript que usa a biblioteca ***jQuery*** e o plugin *DataTables* para configurar uma tabela interativa.

<script type="text/javascript">

$(document).ready(function() {

$('#example').DataTable();

});

</script>



Figura 12.1 - Logo jQuery

## GitHub

O GitHub é uma plataforma online usada para hospedar e gerenciar projetos. Ele permite que desenvolvedores e equipas trabalhem juntos, compartilhem e colaborem no desenvolvimento do código. É um ambiente centralizado onde o projeto é armazenado, gerenciado e modificado.

Um dos principais componentes do GitHub é o repositório (*repository*). Um repositório é um local onde o código de um projeto é armazenado. Ele contém todos os arquivos, pastas e histórico de alterações. Os repositórios podem ser públicos, o que significa que qualquer pessoa pode ver e contribuir para o projeto, ou privados, com acesso restrito aos utilizadores autorizados.



Figura 13.1 - Logo GitHub

# Estágio

## Empresa de estágio

Fundada em 2006, a Centrality é uma empresa especializada em implementações, concepção de soluções na área dos Sistemas de Informação e TV-IP.



Figura 14.1 – Logo Centrality

## Atividades durante o estágio

No início do estágio realizei uma funcionalidade pedida para acrescentar a empresa. Consistia num programa que fazia a comparação de dois ficheiros de Excel sobre o stock e dava como resultados os produtos novos, atualizados e os produtos descontinuados.

O utilizador começava por fazer upload de dois ficheiros de stock (1º Ficheiro, stock na loja, 2º ficheiro stock recebido pelo fornecedor) e depois submetia as percentagens (%) do IVA e das tabelas para venda ao publico. De seguida submetia a comparação e dava o resultado das diferenças entre ficheiros de stock.

Ao concluir a funcionalidade pedida com a ajuda dos colaboradores da empresa, fizemos o upload para uma VM (Máquina Virtual) para conseguir-se aceder sempre que quiser.

No meio do estágio, foi me dada a oportunidade de participar numa formação para toda a empresa sobre a Starti, que acabaria por ser um novo parceiro da Centrality. A Starti é uma empresa brasileira que é especializada em cybersegurança. Tem um próprio software de cybersegurança, no qual eu tive a oportunidade mexer um pouco e aprender mais sobre a importância de cybersegurança.

Na parte final do estágio, foi-me proposto participar num projeto em conjunto com mais estagiários para consertar portáteis que estavam avariados, e que acabariam por ser enviados para angola, como doação, para a Casa do Gaiato. Projeto que a Centrality participou para ajudar quem nunca teve a oportunidade de ter mexido num computador, e que com tão pouco fica contente.

## Análise critica

O estágio realizado na Centrality, teve a duração de cerca de três meses (75 dias uteis). Assim que entrei fui muito bem recebido por todos os colaboradores da empresa, deixando-me a vontade como se tivesse em casa. Sem formalidades, e sempre ao dispor caso eu tivesse alguma dificuldade ou dúvida, isto deixou-me descontraído sabendo que poderia ter o suporte de qualquer pessoa na empresa. Depois de uma primeira conversa com o Sr. Jorge Carvalho foi me proposto criar uma funcionalidade para ser implementada na empresa. Tratei o projeto com seriedade e responsabilidade, visto que era o único a trabalhar no projeto e era algo novo e importante para ser implementado na empresa. No meio do estágio tive uma formação sobre a Starti e achei bastante interessante e deu-me nova visão da importância da cybersegurança. Sobre o último projeto que me foi atribuído achei bastante interessante a iniciativa da Centrality para ajudar quem mais preciso, de esse modo, empenhei-me para conseguir o máximo de computadores possíveis para enviar para angola.

Para concluir, as expectativas sobre o estágio não foi a esperada. O principal problema foi o tipo de projetos e trabalhos que a empresa desenvolve nos quais eu não identifico. Esperava ter um estágio mais direcionado para a parte da programação mais avançada, tendo a possibilidade de aprender linguagens novas e desenvolver a minha capacidade programática, e não para a parte de suporte técnico e de gestão de redes. Quanto ao suporte dado pelo o acompanhante não foi o esperado devido a falta de tempo do mesmo, mas foi recompensado com o suporte do resto dos colaboradores da empresa que tiveram sempre aptos para me ajudar esclarecer as minhas dúvidas.

Apesar das diversidades ocorridas durante o estágio a experiência foi positiva, pois consegui com que melhorasse as minhas capacidades de trabalho autónomo, capacidade de desenvolver para além do pedido e adaptação a trabalhos novos. Fez também que conseguisse desenvolver as minhas *Soft Skills* e preparou-me para um futuro profissional onde desafios semelhantes acontecem, fazendo com que esteja preparado para lidar com os mesmos.

# Introdução

Este projeto foi realizado no âmbito da Prova de Aptidão Profissional, do Curso de Técnico