



RELATÓRIO DE MODULO 16

CURSO TÉCNICO PROFISSIONAL DE GESTÃO E PROGRAMAÇÃO DE SISTEMAS INFORMÁTICOS

Professores: José Alexandre, Luísa Alves, Joaquim Buinho

André Custódio, L1949

Marco Henriques, L1974

01/06/2021

O relatório encontra-se em condições para ser apresentado

Ciclo de Formação 2018/2021
Ano Letivo 2020/2021



Agradecimentos

O nosso grupo tem muito a agradecer aos professores José Alexandre, Joaquim Buinho e Luísa Alves por terem estado sempre ajudar ao longo do projeto.

Gostávamos de agradecer ao professor José Alexandre, por nos ter ajudado na conceção da rede informática do projeto assim como ajuda em vários outros problemas que iam aparecendo

Gostaríamos também de agradecer à professora Luísa Alves pelo gasto do seu tempo pessoal com vista a nos encaminhar no projeto e ajudar na realização do mesmo.



Índice

Agradecimentos	1
Índice	2
Índice de Imagens	4
Introdução	6
Capítulo I – Cronograma Inicial	7
Descrição do Cronograma	8
Capítulo II- Conceção do Projeto.....	9
Objetivos do projeto	9
Desenvolvimento da rede:	10
Equipamentos ativos:.....	10
Tipologia de rede:.....	10
Packet tracer (simulação da rede).....	12
Desenvolvimento do Website ASP.NET	13
Encriptação RSA	13
Registo do cliente	14
Login do Cliente.....	17
Ver/Editar perfil.....	19
Visualizar as encomendas	21
Navegação no Website.....	22
Carrinho.....	24
Query da Base de Dados	27
Desenvolvimento da aplicação do Admin	31
Tecnologias.....	31
Frameworks.....	32
.NET Framework.....	32
Vantagens de usar um Framework:	32
Linguagem de Programação.....	33
C#.....	33
Sql.....	33
IDE	33
Json.....	34



O Programa	34
Abreviaturas:	34
CamelCase	34
Linq	35
<i>Class</i>	35
Ligações a base de dados	36
Try, catch finally	36
Notificações.....	36
Stored Procedure	37
Auto-Run	38
Projeto.....	38
Capítulo IV – Cronograma Final e Justificação de desvios	46
Justificação	47
Conclusão	48
Bibliografia e Web Grafia:	49



Índice de Imagens

Figura 1 - Cronograma Inicial	7
Figura 2 - Topologia em estrela.....	10
Figura 3 - Tabela de IPs de Sta. Maria da Feira	11
Figura 4 - Tabela de IPs de Setúbal	11
Figura 5 - Rede da delegação de Sta. Maria da Feira	12
Figura 6 - Rede da delegação de Setúbal	12
Figura 7 - Encriptação da password	13
Figura 8 - Desencriptação da password	13
Figura 9 - Interface para o utilizador se registar	14
Figura 10 - Verificar se o username já existe	15
Figura 11 - Verificar se as senhas inseridas coincidem e envio do Username e Password.....	15
Figura 12 - Mensagem de passwords diferentes	15
Figura 13 - Código que mostra o cliente a ser inserido com sucesso	16
Figura 14 - Icon de login	17
Figura 15 - Modal do login	17
Figura 16 - Código do Login.....	18
Figura 17 - Mensagem de erro	18
Figura 18 - Mensagem Sucesso no login	18
Figura 19 - Icon para editar perfil.....	19
Figura 20 - Informações pessoais do cliente	19
Figura 21 - Código do select da base de dados dos campos pretendidos e afixação dos mesmos	19
Figura 22 - Menu para editar as informações pessoais	20
Figura 23 - Update das informações pessoais.....	20
Figura 24 - Icon visualizar encomenda	21
Figura 25 - GridView das encomendas feitas pelo utilizador.....	21
Figura 26 - Redes sociais da empresa e definições, encomendas e carinho de compras.....	22
Figura 27 - Produto Masculino	22
Figura 28 - Código do menu de pesquisa do tipo de produto.....	22
Figura 29 - Seleção do tipo de produto da DB e conversão da imagem do produto.....	23
Figura 30 - Display do item escolhido para comprar	24
Figura 31 - Código dos detalhes do produto	24
Figura 32 - Insert dos itens no carrinho de compras.....	25
Figura 33 - Carrinho de compras com itens inseridos.....	25
Figura 34 - Select dos dados dos produtos que estão no carrinho assim como o IdCliente.....	26
Figura 35 - Insert dos produtos na tbl_Encomendas	26
Figura 36 - Delete dos itens da listBox	27
Figura 37 - .Net Framework.....	31
Figura 38 - Linguagens de Programação	31
Figura 39 - Inicio do .Net Framework	32
Figura 40 - C#.....	33
Figura 41 - Sql	33
Figura 42 - Json.....	34



Figura 43 - camelCase	34
Figura 44 - <i>Linq</i>	35
Figura 45 - <i>Class</i> Cliente	35
Figura 46 - Enviar os dados para a <i>class</i>	35
Figura 47 - Ligação a base de dados.....	36
Figura 48 - comando para mostrar o balão das notificações.....	36
Figura 49 - Balão de Notificações.....	36
Figura 50 - C# mais Stored Procedure.....	37
Figura 51 - Stored Procedure Insert Encomendas.....	37
Figura 52 - Código do Auto Run	38
Figura 53 - Form quando iniciamos sessão	38
Figura 54 - Form Administrador	39
Figura 55 - Form dos Administradores.....	39
Figura 56 - Form Adicionar Produtos	39
Figura 57 - Formulário de Editar ou Apagar Produtos	40
Figura 58 - Inicio do <i>Form</i> Cliente e Funcionário	40
Figura 59 - Quando clicamos no Clientes ou Funcionários	41
Figura 60 - Modificar os Dados do Cliente e Funcionário	41
Figura 61 - Form de Adicionar Carrinho	42
Figura 62 - Form Consulta Carrinho	43
Figura 63 - Detalhes de Produtos.....	43
Figura 64 - Cronograma Final	46



Introdução

O projeto do módulo 16, consiste na criação de uma rede informática e um website para o cliente poder aceder aos produtos e poder encomendar, para o Sr. José RoupaJusta da Silva, proprietário da empresa VesteBem Lda.

Este relatório tem como objetivo mostrar o nosso percurso para desenvolver o projeto assim como demonstrar todas as aprendizagens adquiridas neste projeto.

Iremos demonstrar todas as atividades desenvolvidas, os apoios teóricos a que recorreremos e explicar como foi desenvolvido o nosso projeto, identificando todas as ferramentas utilizadas ao longo do seu desenvolvimento



Capítulo I – Cronograma Inicial

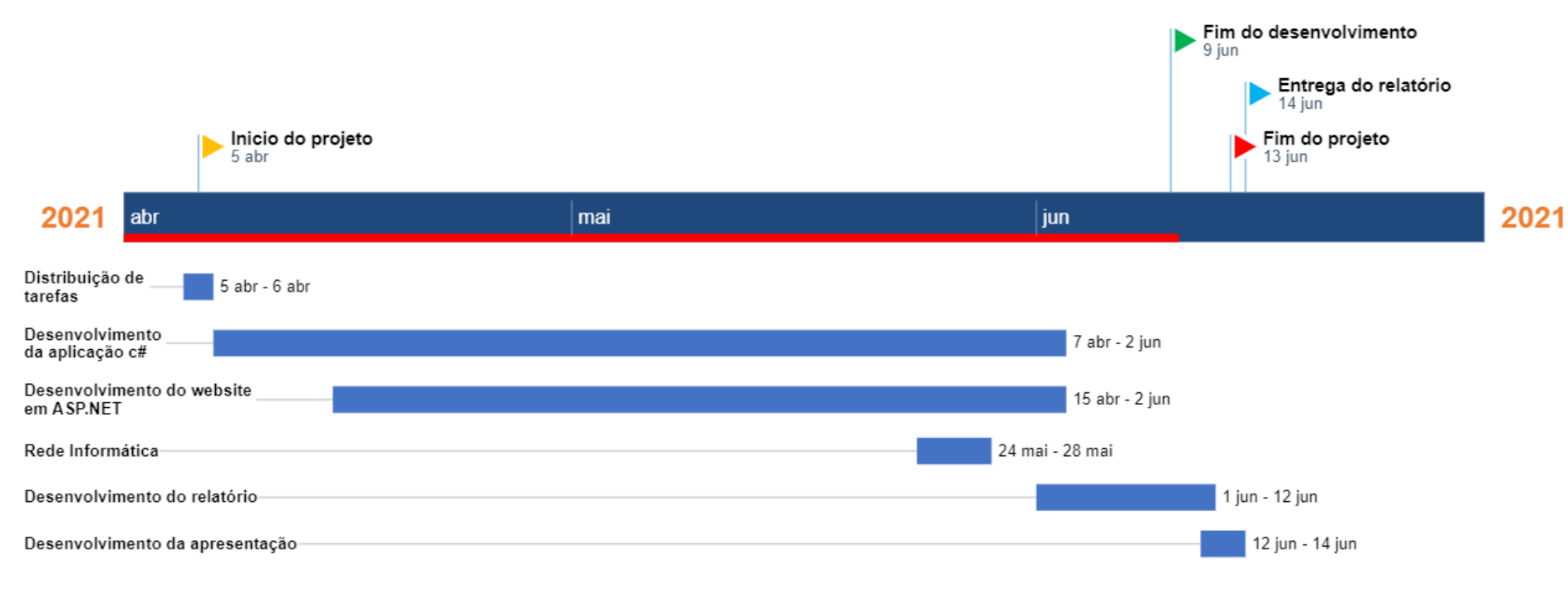


Figura 1 - Cronograma Inicial



Descrição do Cronograma

No cronograma mostrado acima mostra a nossa expectativa de realização de todas as etapas do projeto até ao dia de entrega do projeto, o projeto começou no dia 05/04/2021 e a data de conclusão irá ser dia 13/06/2021.

Se tudo correr como esperado, no dia 02/06/2021 iremos ter a aplicação e o website já prontos, e a rede informática do projeto já feita também no dia 28/05/2021 faltando terminar o relatório e a apresentação.



Capítulo II- Conceção do Projeto

Objetivos do projeto

Projeto: VesteBem_Admin e VesteBem_Site

Descrição do Projeto: O Website tem como objetivo o cliente poder conseguir consultar e encomendar os produtos da empresa Veste Bem. Para o cliente poder efetuar a encomenda terá de fazer login.

A aplicação do administrador, tem como função a manutenção de dados, inserir dados e apagar os mesmos. Para aceder ao aplicativo tem que fazer login primeiro no site e só assim consegue fazer login no aplicativo.

Objetivos:

Ser acessível para todos os clientes;

Fácil de utilizar;

Compatível com a plataforma *Windows*;

✓ Segurança;



Desenvolvimento da rede:

Equipamentos ativos:

- ✓ 1 servidor DHCP(genérico)
- ✓ 2 Routers 5505, um em Setúbal e outro em Sta. Maria da Feira;
- ✓ 4 Switches 2960-24TT em Setúbal
- ✓ 6 Switches 2960-24TT em Lisboa;

Tipologia de rede:

A tipologia de rede que o nosso grupo escolheu adotar foi a tipologia em estrela, nós escolhemos usar esta tipologia porque já a tínhamos usado num projeto anterior e tínhamos visto que era a mais segura e a mais utilizada.

A tipologia de rede em estrela faz com que toda a informação deva passar pelo switch, e graças a isso vai tornar a receção de informação mais eficiente, fazendo com que não haja o encaminhamento de informações para todos os utilizadores e sim só para o devido destinatário.

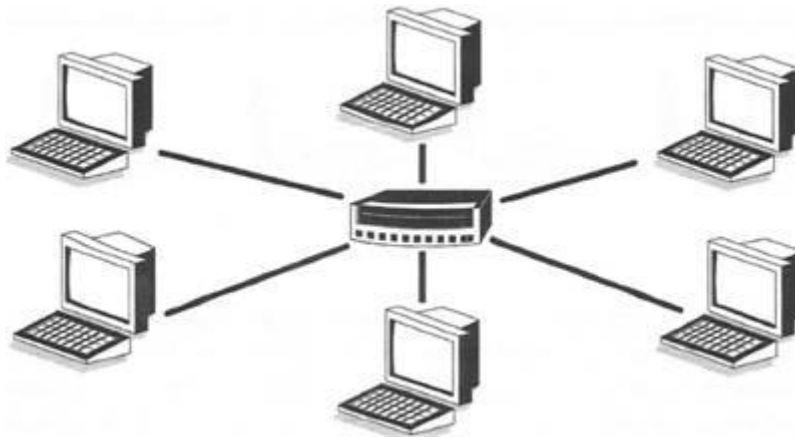


Figura 2 - Topologia em estrela

IPs (DHCP)

O nosso servidor de DHCP tem o Ip: 192.168.1.1 e fornece para Sta. Maria da Feira os IPs para as máquinas entre 192.168.2.1 e 192.168.2.254 enquanto em Setúbal disponibiliza entre 192.168.1.1 e 192.168.1.254. A Subnet mask utilizada nos dois departamentos é 255.255.255.0.



Sta Maria da Feira					
Tabela de Ip's					
Computador	Ip	Ip Inicial	Ip Final	Broadcast	Gateway
Vendas-1	192.168.2.25	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Vendas-2	192.168.2.13	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Vendas-3	192.168.2.26	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Vendas-4	192.168.2.12	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Armazem/Fábrica-1	192.168.2.16	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Armazem/Fábrica-2	192.168.2.23	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Armazem/Fábrica-3	192.168.2.22	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Armazem/Fábrica-4	192.168.2.14	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Armazem/Fábrica-5	192.168.2.21	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Informatica/RH-1	192.168.2.37	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Informatica/RH-2	192.168.2.38	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Informatica/RH-3	192.168.2.32	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Informatica/RH-4	192.168.2.9	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Informatica/RH-5	192.168.2.41	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Informatica/RH-6	192.168.2.5	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Informatica/RH-7	192.168.2.28	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Direção-1	192.168.2.17	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Direção-2	192.168.2.6	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Direção-3	192.168.2.33	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Direção-4	192.168.2.34	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Direção-5	192.168.2.11	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Direção-6	192.168.2.10	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
comercio-1	192.168.2.7	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
comercio-2	192.168.2.29	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
comercio-3	192.168.2.31	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
comercio-4	192.168.2.27	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
comercio-5	192.168.2.15	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
comercio-6	192.168.2.19	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
comercio-7	192.168.2.18	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
comercio-8	192.168.2.30	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Administração-1	192.168.2.20	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Administração-2	192.168.2.35	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Administração-3	192.168.2.8	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Administração-4	192.168.2.36	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Administração-5	192.168.2.39	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1
Administração-6	192.168.2.40	192.168.2.1	192.168.2.254	192.168.2.255	192.168.2.1

Figura 3 - Tabela de IPs de Sta. Maria da Feira

Setúbal					
Tabela de Ip's					
Computador	Ip	Ip Inicial	Ip final	Broadcast	Gateway
Informatica/Direcao-1	192.168.1.60	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Informatica/Direcao-2	192.168.1.59	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Informatica/Direcao-3	192.168.1.72	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Informatica/Direcao-4	192.168.1.61	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Comerciais-1	192.168.1.71	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Comerciais-2	192.168.1.70	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Comerciais-3	192.168.1.69	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Comerciais-4	192.168.1.68	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Comerciais-5	192.168.1.67	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Comerciais-6	192.168.1.66	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Comerciais-7	192.168.1.65	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Comerciais-8	192.168.1.64	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Administrativos-1	192.168.1.62	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Administrativos-2	192.168.1.58	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Administrativos-3	192.168.1.57	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Administrativos-4	192.168.1.56	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Administrativos-5	192.168.1.55	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Administrativos-6	192.168.1.53	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Armazenamento-1	192.168.1.50	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Armazenamento-2	192.168.1.51	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1
Armazenamento-3	192.168.1.52	192.168.1.1	192.168.1.254	192.168.1.255	192.168.1.1

Figura 4 - Tabela de IPs de Setúbal

Packet tracer (simulação da rede)

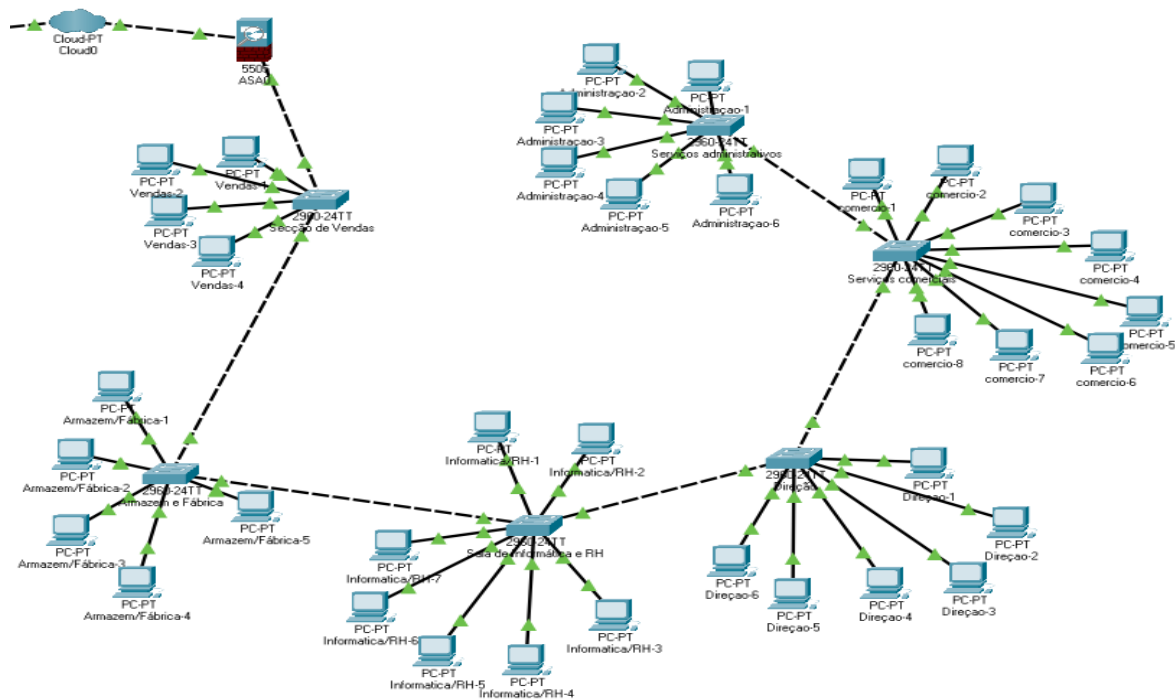


Figura 5 - Rede da delegação de Sta. Maria da Feira

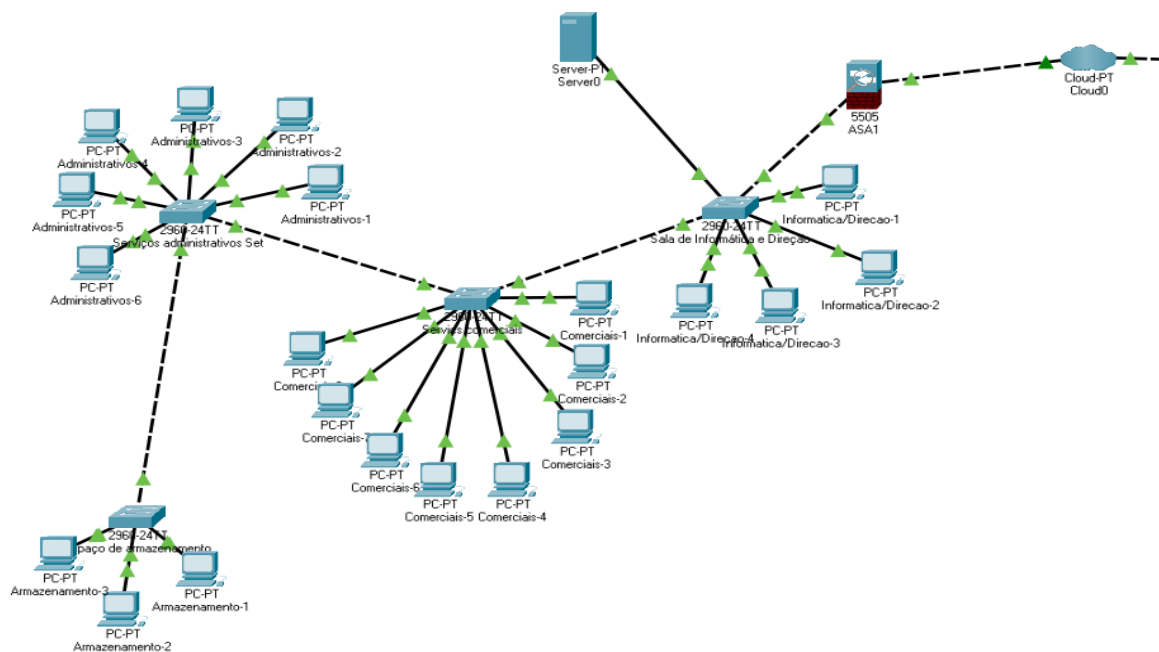


Figura 6 - Rede da delegação de Setúbal



Desenvolvimento do Website ASP.NET

Encriptação RSA

A encriptação RSA foi usada no projeto para encriptar as passwords do utilizador fazendo assim com que o registo e o login do utilizador seja mais seguro, o resultado gerado por cada encriptação de cada password vai ser sempre diferente mesmo que a senha seja igual, ou seja, se a password do utilizador “TesteA” for “123” e a password do utilizador “TesteB” for “123” a encriptação gerada para as suas passwords vai ser diferente mesmo sendo elas iguais.

```
public static string EncryptRSA(string strText)
{
    var publicKey = "<RSAKeyValue><Modulus>21wEnTU+mcD2w0Lfo1Gv4rtcSwsQJQTNa6gio05A";

    var testData = Encoding.UTF8.GetBytes(strText);

    using (var rsa = new RSACryptoServiceProvider(1024))
    {
        try
        {
            // client encrypting data with public key issued by server
            rsa.FromXmlString(publicKey.ToString());

            var encryptedData = rsa.Encrypt(testData, true);

            var base64Encrypted = Convert.ToBase64String(encryptedData);

            return base64Encrypted;
        }
        finally
        {
            rsa.PersistKeyInCsp = false;
        }
    }
}
```

Figura 7 - Encriptação da password

```
public static string DecryptRSA(string strText)
{
    var privateKey = "<RSAKeyValue><Modulus>21wEnTU+mcD2w0Lfo1Gv4rtcSwsQJQTNa6gio05AOk";

    var testData = Encoding.UTF8.GetBytes(strText);

    using (var rsa = new RSACryptoServiceProvider(1024))
    {
        try
        {
            var base64Encrypted = strText;

            // server decrypting data with private key
            rsa.FromXmlString(privateKey);

            var resultBytes = Convert.FromBase64String(base64Encrypted);
            var decryptedBytes = rsa.Decrypt(resultBytes, true);
            var decryptedData = Encoding.UTF8.GetString(decryptedBytes);
            return decryptedData.ToString();
        }
        finally
        {
            rsa.PersistKeyInCsp = false;
        }
    }
}
```

Figura 8 - Desencriptação da password



Registo do cliente

Para os cliente poderem comprar roupa no nosso website os clientes precisam se registar, sem se registarem os clientes conseguem ver os produtos disponiveis na loja através do menu de navegação, mas para poder fazer uma encomenda precisam fazer um registo e um login.

Na nossa página de registo o utilizador tem que nos fornecer todos o seus dados pessoais como o seu nome, data de nascimento, sexo, nif, morada, código postal, localidade, email, telefone, username e a sua password tendo um outro campo para confirmar a password inserida para evitar que o utilizador insira uma password com erros.

Criar Utilizador


Nome do cliente:	<input type="text"/>
Data de nascimento:	<input type="text" value="dd/mm/aaaa"/> 
Sexo:	<input type="radio"/> Masculino <input type="radio"/> Feminino <input type="radio"/> Indefinido
NIF:	<input type="text"/>
Morada:	<input type="text"/>
Código Postal:	<input type="text"/>
Localidade:	<input type="text"/>
Email	<input type="text"/>
Telefone:	<input type="text"/>
Username:	<input type="text"/>
Password	<input type="password"/>
Confirmar Password	<input type="password"/>
<input type="button" value="Registar"/>	<input type="button" value="Voltar"/>

Figura 9 - Interface para o utilizador se registar



```
comando4.CommandText = "Select Count(*) from tbl_login where Usern='" + txtUsername.Text + "'";  
int existe = (int)comando4.ExecuteScalar();  
if(existe > 0 )  
{  
    lblMensagem.Text = "O username já existe!!";  
}
```

Figura 10 - Verificar se o username já existe

Nota: Depois de inserir o username nós decidimos executar o código acima para verificar se o username já existia na base dados, onde se o valor do Count fosse “0” mostraria ao cliente a mensagem “ O username já existe!!”, e se o Count fosse “1” prosseguiria então o registo.

```
if (txtPassword.Text == txtPasswordConfirmar.Text)  
{  
    comando.CommandText = "Insert into Tbl_Login(Usern ,Passw, Funcionario) " + "VALUES('" + txtUsername.Text + "', '" + EncryptAdeDecrypt.EncryptRSA(txtPassword.Text) + "', '0')";  
    SqlDataReader dr;  
}
```

Figura 11 - Verificar se as senhas inseridas coincidem e envio do Username e Password

```
else  
    lblMensagem.Text = "As passwords não são iguais, verifique se estão as duas iguais!";
```

Figura 12 - Mensagem de passwords diferentes

Nota: Nas 2 imagens acima é-nos mostrado como verificamos se as duas passwords são iguais, assim como também o envio dos dados para a base dados.



```
try
{
    comando.ExecuteNonQuery();
    liga.Close();
    liga.Open();
    comando2.CommandText = "Select IdLogin, Passw from tbl_Login where Usern like '" + txtUsername.Text + "'";
    dr = comando2.ExecuteReader();

    if (dr.Read())
    {
        int id = Convert.ToInt32(dr["IdLogin"].ToString());
        if (txtPassword.Text == EncryptADEDecrypt.DecryptRSA(dr["Passw"].ToString()))
        {
            liga.Close();
            liga.Open();
            comando3.CommandText = "Insert into Tbl_Cliente(Nome ,Sexo , Nif,Morada ,CodPostal ,Localidade , DataNasc ,Email ,Telefone, Id_Login) "
            + "VALUES('" + txtNomeCliente.Text + "', '" + RadioButtonList1.SelectedValue + "', '" + txtNif.Text + "', "
            + "'" + txtMorada.Text + "', '" + txtCodPostal.Text + "', '" + txtLocalidade.Text + "', '" + txtDataNasc.Text + "', "
            + "'" + txtEmail.Text + "', '" + txtTelefone.Text + "', '" + id + "')";

            comando3.ExecuteNonQuery();
            lblMensagem.Text = "Registado com sucesso";
        }
    }
}
catch (Exception er)
{
    lblMensagem.Text = er.Message;
}
liga.Close();
}
```

Figura 13 - Código que mostra o cliente a ser inserido com sucesso

Nota: Nesta imagem é nos mostrado então depois das duas verificações já faladas, o envio dos dados do utilizador para a base de dados, assim como a encriptação do username, se algum erro inesperado acontecer enquanto o utilizador está a criar conta e que não tenham esses erros a ver com a password e o username, usámos então um `catch(Exception er)` o que esta função faz é escrever na “`lblMensagem.text`” o tipo de erro que ocorreu na criação do utilizador.



Login do Cliente

Para o login do utilizador nós decidimos utilizar um modal que fosse mais user friendly, o modal então pede o username, a sua password, tem um botão para cancelar a operação e tem também a opção para criar uma conta nova, onde depois de clicado vai dar ao registo de utilizador.

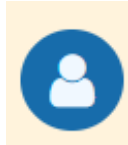


Figura 14 - Icon de login

Nota: Para o utilizador aceder ao modal do login precisa então de clicar no icon que é representado na imagem em cima com uma pessoa.

Figura 15 - Modal do login

Nota: Depois do utilizador clicar no icon vai dar pop-up deste modal onde o proprio poderá fazer o login inserindo os seus dados .



```
try
{
    liga.Open();
    comando.Connection = liga;
    comando.CommandText = "Select Passw, IdLogin from tbl_login where Usern like '" + uname.Value + "'";
    dr = comando.ExecuteReader();
    if (dr.Read())
    {
        int login = Convert.ToInt32(dr["IdLogin"]);
        string pass = dr[0].ToString();
        if (psw.Value == EncryptAeDecrypt.DecryptRSA(dr["Passw"].ToString()))
        {
            Session["Username"] = uname.Value;
            liga.Close();
            liga.Open();
            comando.CommandText = "Select IdCliente from Tbl_Cliente where Id_Login = " + login + ";";
            dr = comando.ExecuteReader();
            if (dr.Read())
            {
                Session["IdCliente"] = dr[0].ToString();
                CheckAdmin.SelectFuncionario(login, uname.Value, EncryptAeDecrypt.EncryptRSA(psw.Value));
                Response.Write("<script>alert('Sessão efetuada com sucesso')</script>");
            }
            entrou = true;
        }
        else
        {
            Response.Write("<script>alert('Pass ou Username/Email Inválidos!!!')</script>");
            entrou = false;
        }
    }
}
```

Figura 16 - Código do Login

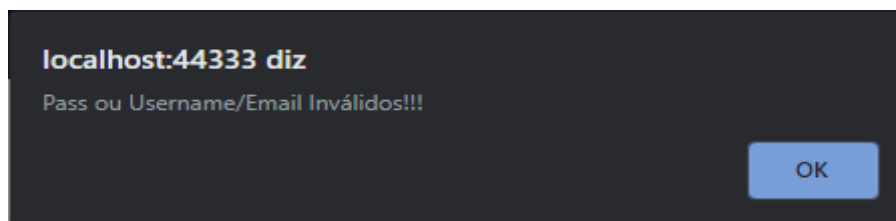


Figura 17 - Mensagem de erro

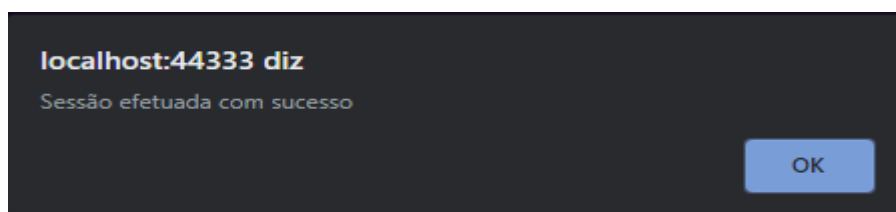


Figura 18 - Mensagem Sucesso no login

Nota: Quando o cliente clicar no botão login o programa retira o valor que está na textbox do username e verifica se esse username existe na base de dados, seguidamente se o username existir vai então descodificar a password e vai verificar se a password inserida coincide com a password que está guardada para aquele utilizador na base de dados, se a password não coincidir com a que está guardada na base de dados vai então aparecer uma mensagem de erro a dizer que a password está errada.



Ver/Editar perfil



Figura 19 - Icon para editar perfil

Informações pessoais

	Nome	Sexo	Nif	Morada	Código Postal	Localidade	Data de nascimento	Email	Telefone
Selecionar	Marco	M	192931232	Rua A	2600-090	Lisboa	09-07-00	marco.henriques.890@gmail.com	911233456

Figura 20 - Informações pessoais do cliente

Nota: Quando o cliente clicar no botão de editar perfil a gridview vai buscar a session[Idcliente] e vai afixar os seus dados pessoais na gridview.

```
protected void GridView1_SelectedIndexChanged1(object sender, EventArgs e)
{
    Panel1.Visible = true;
    BtnAlterações.Visible = true;

    comando.Connection = liga;
    liga.Open();

    try
    {
        comando.CommandText = "Select IdCliente, Nome, Sexo, Nif, Id_Login, Morada, CodPostal, Localidade, " +
            "DataNasc, Email, Telefone from tbl_Cliente where IdCliente = '" + Session["IdCliente"] + "'";
        dr = comando.ExecuteReader();

        if (dr.Read())
        {
            txtNome.Text = dr["Nome"].ToString();
            txtNif.Text = dr["Nif"].ToString();
            txtMorada.Text = dr["Morada"].ToString();
            txtCP.Text = dr["CodPostal"].ToString();
            txtLocalidade.Text = dr["Localidade"].ToString();
            txtDataNasc.Text = dr["DataNasc"].ToString();
            txtEmail.Text = dr["Email"].ToString();
            txtTelefone.Text = dr["Telefone"].ToString();
        }
    }
    catch { }
    liga.Close();
}
```

Figura 21 - Código do select da base de dados dos campos pretendidos e afixação dos mesmos



Informações pessoais

	Nome	Sexo	Nif	Morada	Código Postal	Localidade	Data de nascimento	Email	Telefone
Selecionar	Marco	M	192931232	Rua A	2600-090	Lisboa	09-07-00	marco.henriques.890@gmail.com	911233456

Nome

Marco

Morada

Rua A

Data de nascimento

dd-mm-aaaa

Sexo

☐ Masculino
☐ Feminino
☐ Indefinido

Código Postal

2600-090

Email

marco.henriques.890@gmail.co

Nif

192931232

Localidade

Lisboa

Telefone

911233456

Voltar

Guardar alterações

Figura 22 - Menu para editar as informações pessoais

Nota: Depois de clicar no selecionar vai aparecer este menu onde o cliente irá poder mudar todas as suas informações pessoais com a excessão da sua data de nascimento, o cliente também irá poder voltar para trás clicando no botão voltar, botão este que o encaminhará para a homepage do website.

```
protected void BtnAlterações_Click(object sender, EventArgs e)
{
    try
    {
        comando.Connection = liga;
        liga.Open();
        comando.CommandText = "Update tbl_Cliente set Nome='" + txtNome.Text + "', Sexo = '" + RadioButtonList1.SelectedValue + "', " +
            "Nif = '" + txtNif.Text + "', Morada = '" + txtMorada.Text + "', CodPostal = '" + txtCP.Text +
            "', Localidade = '" + txtLocalidade.Text + "', " +
            "Email = '" + txtEmail.Text + "', Telefone = '" + txtTelefone.Text + "' where IdCliente = '" + Session["IdCliente"] + "'";
        dr = comando.ExecuteReader();
        Response.Write("<script>alert('Alterações registadas')</script>");
    }
    catch (Exception er)
    {
        Response.Write("<script>alert('{er.Message}')</script>");
    }
    liga.Close();
}
```

Figura 23 - Update das informações pessoais

Nota: Com este comando vai ser dado o update das informações pessoais que o cliente escolheu alterar.

Visualizar as encomendas



Figura 24 - Icon visualizar encomenda

As suas encomendas

Nº da encomenda	Data da encomenda	Estado	Valor	Número de cliente
37	09-06-21	Pendente	46,00 €	2
38	09-06-21	Pendente	97,00 €	2
39	09-06-21	Pendente	396,00 €	2
40	09-06-21	Entregue	16,00 €	2
42	13-06-21	Pendente	138,00 €	2

Figura 25 - GridView das encomendas feitas pelo utilizador

Nota: Para que o utilizador veja as suas encomendas efetuada ele terá que clicar no icon da caixa, seguidamente irá então aparecer uma gridview com as informações das suas encomendas efetuadas.



Navegação no Website

Depois do cliente então fazer o login ele terá a possibilidade de navegar no website e de fazer compras, para fazer isso tem duas categorias que é homem e mulher, depois de carregar num dos sexos vai ser então redirecionado para a pagina de itens correspondente ao sexo escolhido pelo utilizador.

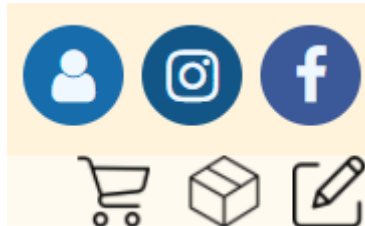


Figura 26 - Redes sociais da empresa e definições, encomendas e carinho de compras

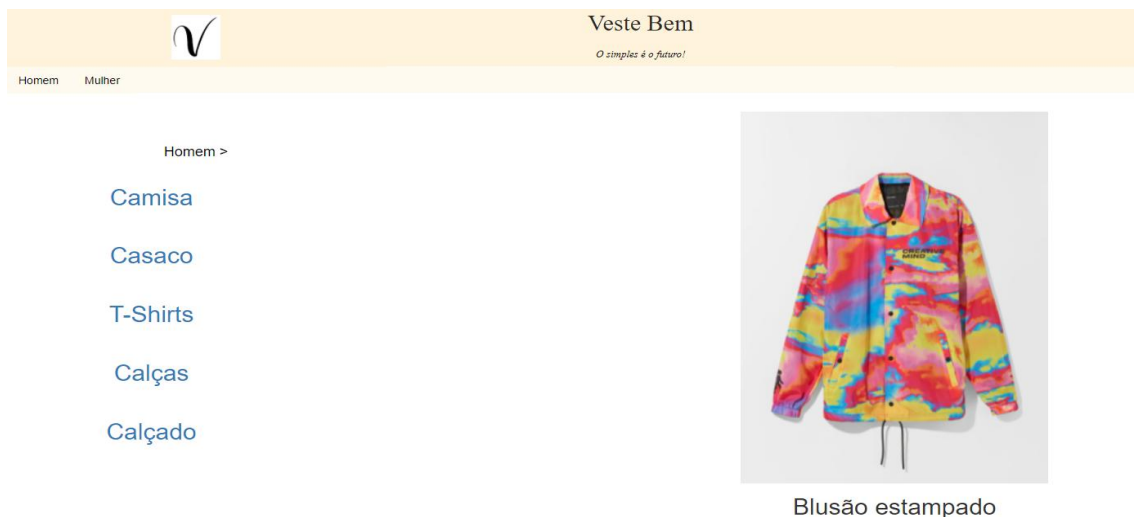


Figura 27 - Produto Masculino

Nota: Quanto o utilizador clicar na secção desejada, irá então ser mostrado a roupa que está nessa categoria, no exemplo da imagem em cima nos casacos.

```
<asp:Menu ID="Menu1" runat="server" StaticSubMenuIndent="16px" Height="394px" RenderingMode="Table" Width="297px">
  <Items>
    <asp:MenuItem Text="Camisa" Value="Camisa" NavigateUrl="~/FrmCHomem.aspx?CategoriaClasse=Camisa"></asp:MenuItem>
    <asp:MenuItem Text="Casaco" Value="Casaco" NavigateUrl="~/FrmCHomem.aspx?CategoriaClasse=Casaco"></asp:MenuItem>
    <asp:MenuItem Text="T-Shirts" Value="T-Shirts" NavigateUrl="~/FrmCHomem.aspx?CategoriaClasse=T-Shirts"></asp:MenuItem>
    <asp:MenuItem Text="Calças" Value="Calças" NavigateUrl="~/FrmCHomem.aspx?CategoriaClasse=Calças"></asp:MenuItem>
    <asp:MenuItem Text="Calçado" Value="Calçado" NavigateUrl="~/FrmCHomem.aspx?CategoriaClasse=Calçado"></asp:MenuItem>
  </Items>
  <StaticMenuStyle HorizontalPadding="60px" VerticalPadding="100px" />
</asp:Menu>
```

Figura 28 - Código do menu de pesquisa do tipo de produto



```
private void FillPage()
{
    string art = Request.QueryString["CategoriaClasse"].ToString();
    string sexo = "M";
    command.Connection = liga;
    liga.Open();
    command.CommandText = "Select IdProduto ,Icon, CategoriaClasse, Nome from Tbl_Produtos where CategoriaClasse like '" + art + "' and Sexo ='" + sexo + "'";
    dr = command.ExecuteReader();
    while (dr.Read())
    {
        Panel productPanel = new Panel();
        ImageButton images = new ImageButton();
        Label LBL = new Label();
        byte[] image = (byte[])dr["Icon"];
        string PROFILE_PIC = Convert.ToBase64String(image);
        images.ImageUrl = String.Format("data:image/jpeg;base64,{0}", PROFILE_PIC);
        images.Width = 376;
        images.Height = 500;
        //images.ImageUrl = "~/Image/" + ;
        images.CssClass = "productImage";
        images.PostBackUrl = "~/Detalhes.aspx?IdProduto=" + dr["IdProduto"];
        LBL.Text = dr["Nome"].ToString();
        // LBL.CssClass = "productName";
        productPanel.Controls.Add(images);
        productPanel.Controls.Add(new Literal { Text = "<br/>" });
        productPanel.Controls.Add(LBL);
        productPanel.Controls.Add(new Literal { Text = "<p>" });
        productPanel.Controls.Add(new Literal { Text = "<br/>" });
        productPanel.Controls.Add(new Literal { Text = "<br/>" });
        productPanel.Controls.Add(new Literal { Text = "<br/>" });

        Imagens.Controls.Add(productPanel);
    }
}
```

Figura 29 - Seleção do tipo de produto da DB e conversão da imagem do produto

Nota: Quando o utilizador escolher um produto que tem definido uma “CategoriaClasse” é enviado esse valor à base de dados e a base de dados retorna os valores que foram escolhidos pelo utilizador e então é mostrada a roupa pretendida na página (Página do sexo feminino feito da mesma maneira).



Carrinho

Quando um utilizador clicar num item ele será enviado para o form Detalhes que vai então possibilitar a compra do mesmo assim como examinar o item melhor.

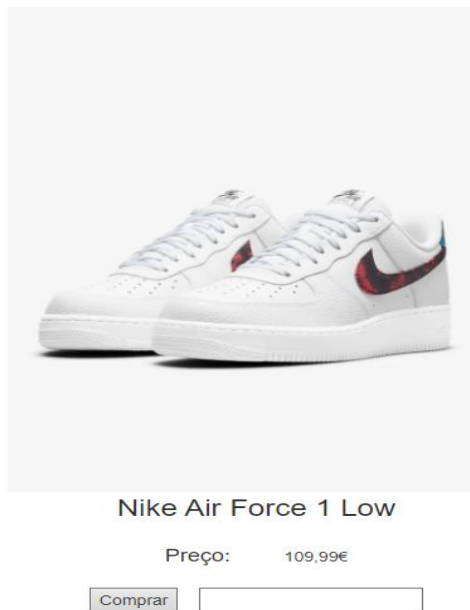


Figura 30 - Display do item escolhido para comprar

```
private void FillPage()
{
    if (!string.IsNullOrEmpty(Request.QueryString["IdProduto"]))
    {
        int id = Convert.ToInt32(Request.QueryString["IdProduto"]);
        comando.Connection = liga;
        liga.Open();
        comando.CommandText = "Select * from Tbl_Produtos where IdProduto='" + id + "'";
        dr = comando.ExecuteReader();
        if (dr.Read())
        {
            lblItem.Text = id.ToString();
            byte[] image = (byte[])dr["Icon"];
            string PROFILE_PIC = Convert.ToBase64String(image);
            imgProduct.ImageUrl = String.Format("data:image/jpg;base64,{0}", PROFILE_PIC);
            imgProduct.Width = 376;
            imgProduct.Height = 500;
            // <!-- src="roupa%20HM/cl2.jfif">
            lblTitle.Text = dr["Nome"].ToString();
            lblPreco.Text = dr["Valor"].ToString() + "€";
        }
        liga.Close();
    }
}
```

Figura 31 - Código dos detalhes do produto

Nota: Este é código por trás do display do item no form Detalhes que vai fazer com que sejam afixadas as informações do produto desejado



```
comando.CommandText = "Insert into tblCarrinho (Id_Cliente, Id_Produtos, QuantCar) " + "VALUES(" + Session["IdCliente"].ToString() + "," + lblItem.Text + "," + txtquant.Text + ")";  
  
try  
{  
    int estado = comando.ExecuteNonQuery();  
  
    if (estado == 1)  
        Response.Write($"<script>alert('Item adicionado ao carrinho!!!')</script>");  
}  
  
catch (Exception er)  
{  
    Response.Write($"<script>alert('{er.Message}')</script>");  
}  
  
liga.Close();
```

Figura 32 - Insert dos itens no carrinho de compras

Nota: Quando é cliado o botão comprar, este código é executado e insere os dados do cliente e do item inserido na tblCarrinho.

Veste Bem
O simples é o futuro!

Homem Mulher

O meu carrinho

Produto:

Preço

Camisa relaxed estampada - 2
Calças chinos slim fit - 1
T-shirt Wu-tang Clan - 1
Nike Air Force 1 Low - 1

186 €

Continuar Compras Finalizar encomenda

Figura 33 - Carrinho de compras com itens inseridos

Nota: Nesta página vão então aparecer todos os produtos inseridos na tblCarrinho com o idCliente do utilizador que está logado no website, nesta página tem também a opção de continuar as compras, apagaros itens do carrinho de compras e finalizar a encomenda, sendo a morada de entrega aquela que o cliente inseriu no registo.



```
comando.Connection = liga;
liga.Open();

comando.CommandText = "select tblCarrinho.Id_Produtos, Nome, Valor, QuantCar from tblCarrinho, tbl_Produtos " +
"where tblCarrinho.Id_Produtos = Tbl_Produtos.IdProduto and tblCarrinho.Id_Cliente = '" + Session["IdCliente"] + "'";
try
{
    dr = comando.ExecuteReader();
    while (dr.Read())
    {
        lstProduto.Items.Add(dr["Nome"].ToString() + " - " + dr["QuantCar"].ToString());
        total = total + (Convert.ToInt32(dr["Valor"]) * Convert.ToInt32(dr["QuantCar"]));
    }
    lblpreco.Text = total.ToString() + " € ";
}
catch (Exception er)
{
    Response.Write($"<script>alert('{er.Message}')</script>");
}
liga.Close();
```

Figura 34 - Select dos dados dos produtos que estão no carrinho assim como o IdCliente

Nota: É graças a este código que aparece os itens no carrinho pois ele dá um select aos itens que foram postos no carrinho por um determinado IdCliente e depois vai buscar à DB os valores da tblCarinho que correspondem a esse IdCliente e afixa essas valores na listbox.

```
comando.CommandText = "Insert into tbl_Encomendas (Id_Cliente, DataEncomenda, ValorEncomendas, EstadoEncomendas)" +
"Values ('" + Session["IdCliente"] + "', '" + data + "', '" + total + "', '" + situacao + "')";
try
{
    comando.ExecuteNonQuery();
    liga.Close();
    liga.Open();
    comando.CommandText = "Select Max(IdEncomendas) from Tbl_Encomendas";
    dr = comando.ExecuteReader();
    if (dr.Read())
    {
        numero = dr.GetValue(0).ToString();
        comando.CommandText = "Insert into tblDetalleEncomendas (Id_Encomendas, Id_Produtos, QuantEnc) Select " + "'" + numero + "', Id_Produtos, QuantCar from tblCarrinho " +
"where tblCarrinho.Id_Cliente = '" + Session["IdCliente"] + "'";

        liga.Close();
        liga.Open();
        comando.ExecuteNonQuery();
        liga.Close();
        liga.Open();
        comando.CommandText = "Delete from tblCarrinho where Id_Cliente = '" + Session["IdCliente"] + "'";
        comando.ExecuteNonQuery();
        Response.Write($"<script>alert('Encomenda finalizada com sucesso')</script>");
        liga.Close();
    }
    lstProduto.Items.Clear();
    lblpreco.Text = "";
}
catch (Exception er)
{
    Response.Write($"<script>alert('{er.Message}')</script>");
}
liga.Close();
```

Figura 35 - Insert dos produtos na tbl_Encomendas

Nota: Com este código são então enviados os valores da listbox para a tabela tbl_Encomendas, e são também inseridos os detalhes da encomenda estando assim então a encomenda concluída.



```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    comando.Connection = liga;
    liga.Open();
    comando.CommandText = "Delete from tblCarrinho where Id_Cliente = '" + Session["IdCliente"] + "'";
    comando.ExecuteNonQuery();
    liga.Close();

    lstProduto.Items.Clear();
    lblpreco.Text = "";
}
```

Figura 36 - Delete dos itens da listbox

Nota: Este código é usado para apagar os itens que estão no carrinho.

Query da Base de Dados

Create table tbl_login(

IdLogin int identity(1,1) primary key,

Uservn varchar(max) not null,

Passw varchar(max) not null,

Funcionario varchar(1) not null

);

Create table tblFuncao(

IdFuncao int identity(1,1) primary key,

Funcao varchar(100), --Nome da funcao

);

Create table tbl_Funcionario(

IdFuncionario int identity(1,1) primary key,

Id_Funcao int Foreign key references tblFuncao(idFuncao),

Nome varchar(100) not null,



```
Telemovel varchar(9) not null,  
Id_Login int Foreign key references tbl_login(idLogin)  
);
```

```
Create table tblEstado(  
IdEstado int identity(1,1) primary key,  
Estado varchar(100) not null, --Em loja, Nos ctts, ...  
);
```

```
Create table tbl_Cliente(  
IdCliente int identity(1,1) primary key,  
Nome varchar(100) not null,  
Sexo varchar(1) not null,  
Nif varchar(9) not null,  
Id_Login int foreign key references tbl_login(idLogin) not null,  
Morada varchar(250) not null,  
CodPostal varchar(8) not null,  
Localidade varchar(100) not null,  
DataNasc date not null,  
Email varchar(300) not null,  
Telefone varchar(9) not null,  
Icon Image  
);
```

```
Create table tbl_Produtos(  
IdProduto int identity(1,1) primary key,
```



Nome varchar(100) not null,
Valor Decimal(7,2) not null,
NomedaEmpresa varchar(100) not null,
CategoriaClasse varchar(100),
CategoriaSubClasse varchar(100),
Sexo varchar(1) not null,
Icon Image not null
);

Create table tbl_Encomendas(
IdEncomendas int identity(1,1) primary key,
ValorEncomendas Decimal(7,2) not null,
EstadoEncomendas int foreign key references tblEstado(IdEstado),
DataEncomenda date not null,
Id_Cliente int foreign key references tbl_Cliente(idCliente)
);

Create table tblDetalheEncomendas(
Id_Encomendas int foreign key references tbl_Encomendas(IdEncomendas),
Id_Produtos int foreign key references tbl_Produtos(IdProduto),
QuantEnc int not null,
Primary key(Id_Encomendas, Id_Produtos)
);

Create table tblCarrinho(
IdCarrinho int identity(1,1) primary key,



```
Id_Cliente int foreign key references tbl_Cliente(idCliente),  
Id_Produtos int foreign key references tbl_Produtos(IdProduto),  
QuantCar int not null,  
);
```



Desenvolvimento da aplicação do Admin

Este aplicativo tem como objetivo o funcionário, fazer atualização dos dados, eliminar os mesmos ou inserir. O mesmo tem um aspeto visual simples e fácil de ser utilizado

Tecnologias

O nosso projeto do Admin utilizamos uma frameworks (**.Net Framework**) e duas linguagem de programação (**C# e Sql**).



Figura 37 - .Net Framework



Figura 38 - Linguagens de Programação



Frameworks

Um framework consiste numa abstração que une códigos entre vários projetos de software, fornecendo uma funcionalidade sem pormenores. Trata-se de um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação.

Ao contrário das bibliotecas é o framework quem dita o controlo da aplicação.



Figura 39 - Início do .Net Framework

.NET Framework

.NET Framework como o nome diz é um *Framework open-source*, desenvolvida pela *Microsoft*. Qualquer código .NET pode ser executada em todos os dispositivos que tenham um *Framework* da plataforma. A sua versão mais atual e estável é o 4.8 lançada em 04/18/2019.

Vantagens de usar um Framework:

Uma das grandes vantagens de um *framework* é a padronização (auxiliar na maximização da compatibilidade, reprodutibilidade, segurança e qualidade) do desenvolvimento. Por termos um conjunto já definido de classes e/ou funções, somos “forçados” a trabalhar conforme a ferramenta de escolha.



Linguagem de Programação

C#

Foi criada pela *Microsoft* em 2001. É simples, moderna, orientada por objetos (conceito de “objetos”, que podem conter dados na forma de campos) e flexível para empreendedores modernos, pois é um recurso para criar *software* que não só funcionará hoje, mas também será aplicável futuramente.



Figura 40 - C#

Sql

Em 12 de junho de 1988, a *Microsoft* uniu-se com o *Ashton-Tate* e *Sybase* para criar um software que iria guardar os dados todos digitalmente, pois antes disso os dados eram todos guardados em papel.

Edgar Frank Codd desenvolveu o modelo relacional, que consiste na maneira como os dados são guardados e que teriam ligação entre elas próprias.



Figura 41 - Sql

Mais tarde a *Sybase* e a *Microsoft* separaram-se e cada uma criou o seu negocio e os seus modelos O *Sql* tenta fazer a redundância de dados (evitar erros, poupar espaço de memória, evitar perda de tempo e facilitar na pesquisa de dados) e é case sensitivity, aceita tanto maiúsculas tanto minúsculas no código.

IDE

Integrated Development Environment ou Ambiente de Desenvolvimento Integrado é uma software que auxilia os desenvolvedores a desenvolverem o programa. Normalmente os IDEs facilitam a técnica de RAD (Rapid Application Development, Desenvolvimento Rápido de Aplicações) a criar um debug muito rapidamente. Com o uso da IDE é mais fácil de encontrar o erro e de o corrigir, sem ter de ler o código todo, uma grande vantagem é que os IDEs são orientados a objetos, completa alguns linhas de código, tem texto de atalho (como o *cw*, que fica *Console.WriteLine*), podemos importar alguns dlls publicas para o nosso projeto, etc.

No nosso projeto usamos o *Microsoft SQL Management Studio* e *Visual Studio 2019*.



Json

Json ou *Java Script Object Notation* é uma extensão de ficheiro, neste caso *.json. Temos de o instalar no “newtonsoft.com/json”, para conseguirmos escrever e ler objetos. Para o conseguirmos usar no programa tem de ser convertido.

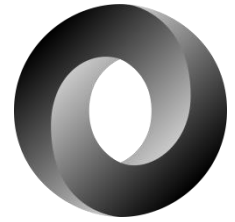


Figura 42 - Json

O Programa

Abreviaturas:

Para deixarmos a codificação mais fácil e sabermos qual é o objeto que estamos a chamar usamos abreviaturas; quase todas as abreviaturas são utilizadas por todos os programadores. Vou demonstrar os utilizados no programa:

Backgroundworker -> bgw...
Button -> btn...
ComboBox -> cmb...
Label -> lbl...
NotifyIcon -> icn...
OpenFileDialog -> ofd...
Panel -> pnl...
PictureBox -> pic_...
TextEdit -> txt...
NumericUpDown -> rdo...
DateTimePicker -> dtp...
FlowLayoutPanel -> flp...
Form -> frm...
Timer -> tmr...

Onde estão as reticências, colocamos o nome da utilização do objeto, por exemplo.:
btnGuardar, btnLogin, txtUserName .

CamelCase

Para identificarmos o objeto e a sua função usamos a abreviatura dos objetos mais a sua função. Existe várias técnicas, há mais comum de os desenvolvedores usarem é a *CamelCase*, a primeira palavra pode ser escrita em *UpperCamelCase* ou *lowerCamelCase*, mas o resto tem de ser escrito em maiúscula.

Optamos por usar o *lowerCamelCase*, por ser mais fácil e versátil.



Figura 43 - camelCase



Linq

Language Integrated Query ou consulta integrada à linguagem, é uma tecnologia, e um mecanismo para facilitar consulta dos dados organizados da lista.

O *LINQ* é mais comum de ser usado nas consultas de bancos de dados usando o *Entity Framework*.

```
lstDetalhesEncomendas.ToList().ForEach(item =>
{
    ds
});
```

Figura 44 - Linq

Class

Uma *Class* ajuda-nos a retornar valores e a enviar valores para aquela variável. Nas listas associamos as classes como o tipo de variável.

Por exemplo para enviarmos os valores para as variável da class temos de escrever o nome da classe, neste caso *Cliente*. Depois para chamar a class usamos o *Cli.Nome*.

```
Cliente Cli = new Cliente();
Cli.Nome = Txt.Text;
Cli.Morada = Txt.Text;
```

Figura 46 - Enviar os dados para a class

```
public class Cliente
{
    /// <summary>
    /// Id do Cliente
    /// </summary>
    9 references | André Custódio, 20 days ago | 2 authors, 2 changes
    public int Id_Cliente { get; set; }
    /// <summary>
    /// Nome do Cliente
    /// </summary>
    14 references | 11949, 38 days ago | 1 author, 2 changes
    public string Nome { get; set; }
    /// <summary>
    /// Sexo do Cliente
    /// </summary>
    7 references | 11949, 38 days ago | 1 author, 2 changes
    public string Sexo { get; set; }
    /// <summary>
    /// Nif do Cliente
    /// </summary>
    6 references | 11949, 38 days ago | 1 author, 2 changes
    public string Nif { get; set; }
    7 references | 11949, 38 days ago | 1 author, 1 change
    public int Id_Login { get; set; }
    /// <summary>
    /// Morada do Cliente
    /// </summary>
    7 references | 11949, 38 days ago | 1 author, 2 changes
    public string Morada { get; set; }
```

Figura 45 - Class Cliente



Ligações a base de dados

```
public static List<Funcao> SelectFuncao()
{
    List<Funcao> lst = new List<Funcao>();
    SqlConnection liga = new SqlConnection(@"Server=tcp:srv-epbjc.database.windows.net,1433;Initial Catalog=bd;Persist Security Info=False;User ID=epbjc;
    SqlCommand comando = new SqlCommand("Select IdFuncao, Funcao From tblFuncao", liga);
    try
    {
        comando.Connection = liga;
        liga.Open();
        using (SqlDataReader oReader = comando.ExecuteReader())
        {
            while (oReader.Read())
            {
                Funcao Fun = new Funcao();
                Fun.IdFuncao = int.Parse(oReader["IdFuncao"].ToString());
                Fun.Funcoes = oReader["Funcao"].ToString();
                lst.Add(Fun);
            }
        }
    }
    catch
    {
        return null;
    }
    finally
    {
        liga.Close();
    }
}
```

Figura 47 - Ligação a base de dados

Na imagem 47 temos o comando *Select IdFuncao e Funcao* para conseguir identificar a função do funcionário e o seu id. Com o *SqlCommand* faço o comando que quero que seja executado, o *SqlConnection* é o nome do servidor para tentar ligar ao mesmo. Ligo o *comando.connection* a ligação do sql e abro a ligação. Mando os dados para o *SqlDataReader* para certificar se existe dados e envia todos os dados do *DataReader* para uma lista criada. Após a sua finalização ele retorna os valores.

Try, catch finally

Esta ferramenta ajuda muito na execução do código. Tenta executar aqueles comandos, se aparecer algum erro, entra no *catch*. Após finalizar o *try* ou o *catch* ele vai para o *finally*, onde nós metemos a *liga.Close()*;

Notificações

O 1 é a duração que o balão das notificações aparece, o 2 é o título da mensagem, o 3 é o texto e o ponto 4 é o icon da notificação.

```
1 2 3 4
IconNotificação.ShowBalloonTip(25, "Error Login", "Login sem sucesso!\nCertifique se a password e username está correto!", ToolTipIcon.Error);
```

Figura 48 - comando para mostrar o balão das notificações

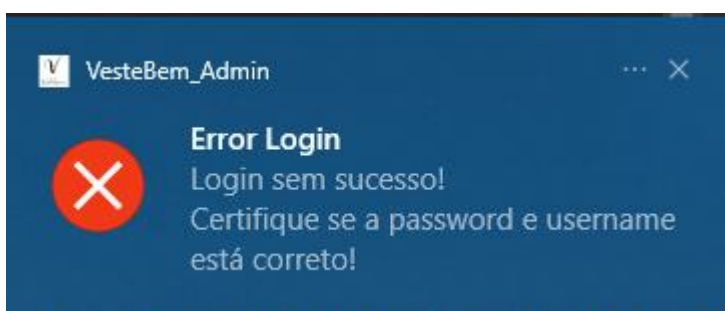


Figura 49 - Balão de Notificações



Stored Procedure

```
command.CommandText = "SpInsertEncomenda";  
command.CommandType = System.Data.CommandType.StoredProcedure;  
  
try  
{  
    command.Parameters.Add(new SqlParameter("ValorEncomendas", enc.ValorEncomendas));  
    command.Parameters.Add(new SqlParameter("EstadoEncomendas", IdEstado));  
    command.Parameters.Add(new SqlParameter("DataEncomenda", enc.DataEncomenda));  
    command.Parameters.Add(new SqlParameter("Id_Cliente", enc.Id_Cliente));  
  
    command.Connection = liga;  
  
    liga.Open();  
  
    command.ExecuteNonQuery();  
}
```

Figura 50 - C# mais Stored Procedure

A *Stored Procedure* ajuda muito na execução de código SQL. Em vez de escrever muitas vezes o mesmo código (“*SELECT * FROM tbl...*”), basta só chamar a *Stored Procedure* e enviar os seus campos para ela e após isso executar e os dados são enviados com sucesso.

Criamos variáveis temporárias que são associadas as do nosso projeto e podemos inserir na tabela.

```
1  /***** Object: StoredProcedure [dbo].[SpInsertEncomenda]    Script Date: 13/06/2021 00:21:11 *****/  
2  SET ANSI_NULLS ON  
3  GO  
4  SET QUOTED_IDENTIFIER ON  
5  GO  
6  ALTER PROCEDURE [dbo].[SpInsertEncomenda]  
7  (@ValorEncomendas decimal(7, 2), @EstadoEncomendas int, @DataEncomenda date, @Id_Cliente int)  
8  AS  
9  BEGIN  
10     INSERT INTO tbl_Encomendas(ValorEncomendas, EstadoEncomendas, DataEncomenda, Id_Cliente) values (@ValorEncomendas, @EstadoEncomendas,  
11         @DataEncomenda, @Id_Cliente)  
12     SET NOCOUNT ON  
13 End
```

Figura 51 - Stored Procedure Insert Encomendas



Auto-Run

```
CheckBox checkBox = sender as CheckBox;  
if (checkBox.Text == "Auto Run (Abrir com o SO, Sistema Operativo)")  
{  
    RegistryKey rkApp = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true);  
    if (chkAuto.Checked)  
    {  
        Application.EnableVisualStyles();  
        rkApp.SetValue("VesteBem Admin", Application.ExecutablePath.ToString());  
    }  
    else  
        rkApp.DeleteValue("VesteBem Admin", false);  
}
```

Figura 52 - Código do Auto Run

O *RegistryKey* é uma variável onde podemos enviar informação para a central do sistema operativo. Normalmente todos os processos do Windows estão guardados no *regedit*, onde podemos ver todos os eventos do Windows, até o do nosso projeto.

Projeto

Figura 53 - Form quando iniciamos sessão

Quando arrancamos com a aplicação deparamos com este layout. Para o *textbox* do *username* e a *textbox* da *password* estarem completos, precisamos de fazer login no site primeiro e só depois conseguimos fazer login no aplicativo.

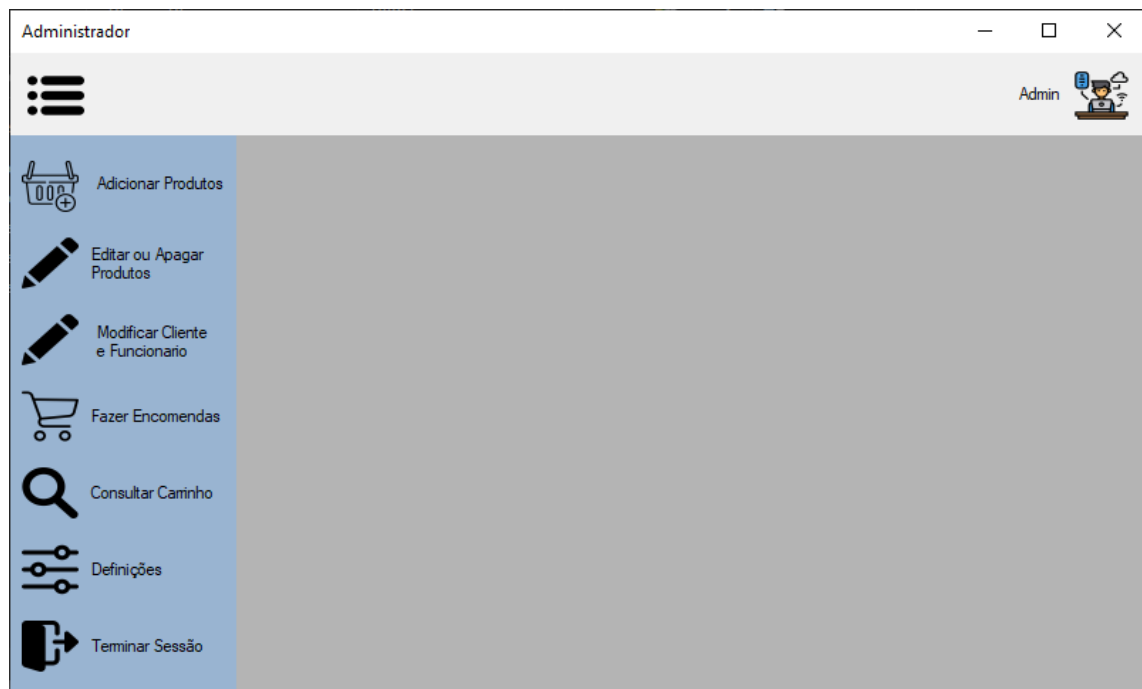


Figura 54 - Form Administrador

Após efetuar o login com sucesso poderá o funcionário aceder a este *form*, onde poderá adicionar produtos, Editar ou Apagar Produtos, Modificar Clientes e Funcionário, Fazer Encomendas, Consultar Carrinho e Terminar Sessão.

Se clicar no Adicionar Produtos, ele irá mostrar outro *form* para adicionar um novo Produto. Algumas *textbox*, só aceitam números e outras só aceitam texto. Um exemplo disso é a *textBoxValor* só números e os pontos.

Figura 56 - Form Adicionar Produtos



Após adicionar os dados todos, irá aparecer uma notificação do programa a dizer que os dados foram adicionados com sucesso. E fecha o *form* e o *formAdministrador* aparece novamente. Para consultar os produtos todos temos de clicar no Editar ou Apagar Produto, onde o Funcionário poderá apagar o produto ou editar o mesmo, que levará ao *form* anterior com os dados daquele produto. Poderá procurar todos os produtos ou pelo nome ou pela categoria.

Nome do Produto	Categoria	Editar	Apagar
Calças de ganga	Calças	Calças de Ganga	
T-shirt RickMorty	T-shirts	T-shirt Masculina	
T-shirt Wu-tang Clan	T-shirts	T-shirt com estampado	
T-shirt preta Nirvana	T-shirts	T-shirt com estampado nirvana	
T-shirt cinza Manhattan	T-shirts	T-shirt cinz	
T-shirt com estampado	T-shirts	T-shirt com estampado	
Calças chinos slim fit	Calças	Calças chinos	
Calças jogger com estampado	Calças	Calças com estampado	
Calças fluidas com abertura	Calças	Calças diferenciadas	
Skinny Jeans fit	Calças	Calças fit	

Figura 57 - Formulário de Editar ou Apagar Produtos

No *form* Cliente e Funcionário conseguimos consultar os funcionários e clientes. Para consultar todos os clientes, temos de clicar em cima do *toolStripMenuItem* onde diz Cliente, e aparece todos os clientes da Base de Dados. Poderá procurar todos os clientes com aquele nome, modificar os seus dados e eliminar. Mesma coisa para o funcionário, mas o funcionário poderá adicionar

Figura 58 - Início do *Form* Cliente e Funcionário



Figura 59 - Quando clicamos no Clientes ou Funcionários

Como no adicionar produtos, as *textbox* só aceitam certos caracteres, no *Nif* só aceitam números e no telemóvel. No nome só aceita texto como na localidade. Após as alterações feitas o funcionário clica no guardar e vai voltar para o outro *form* e este é atualizado automaticamente.

Modificar Cliente	Modificar Funcionario
Nome: <input type="text" value="André"/>	Função: <input type="text" value="Admin"/>
Sexo: <input type="text" value="Masculino"/>	Nome: <input type="text" value="Andre"/>
Nif: <input type="text" value="696969696"/>	Telemovel: <input type="text" value="999999999"/>
Morada: <input type="text" value="Rua amélia"/>	Username: <input type="text" value="Admin"/>
Codigo Postal: <input type="text" value="2900-090"/>	Password: <input type="text" value="Admin"/>
Localidade: <input type="text" value="Lisboa"/>	
Data de Nascimento: <input type="text" value="07 May 2021"/>	
Email: <input type="text" value="Rua amélia"/>	
Telefone: <input type="text" value="937723232"/>	
<input type="button" value="Guardar"/>	<input type="button" value="Guardar Alterações"/>
<input type="button" value="Cancelar"/>	<input type="button" value="Cancelar"/>

Figura 60 - Modificar os Dados do Cliente e Funcionário



Desenvolvi este *form*, o Adicionar Carrinho, caso o cliente vá a loja efetuar a encomenda de um produto.

O Funcionário seleciona um produto na *combobox* e depois escreve a quantidade e depois clica na *picturebox* com um sinal de mais e o produto será adicionado a lista de compras. Quando achar que acabou de efetuar a compra, clica no carrinho e os dados serão enviados para o lado esquerdo, depois tem de selecionar o cliente onde será efetuado a compra e mais ou menos a data de entrega. Quando tiver finalizado é só clicar efetuar Compra.

Figura 61 - Form de Adicionar Carrinho



Estados	Cliente ou IdEncomenda	Encomendas de	Encomendas até
Pendente		13 June 2021	14 June 2021

Estados	Cliente	Nº de Encomenda	Data Encomendada
Pendente	jesus nazareno	42	2021-06-13
Pendente	André	36	2021-06-13

Figura 62 - Form Consulta Carrinho

O funcionário consegue consultar todos os carrinhos disponíveis no *form* Consultar Carrinho. Conseguem pesquisar pelo nome de cliente, Id da Encomenda e selecionar as encomendas naquele período de tempo. Para ver o que os clientes compraram tem de clicar na lupa e irá aparecer todos os produtos do carrinho.


Cliente	Produto	Imagem
jesus nazareno	T-shirt RickMorty	
jesus nazareno	T-shirt com estampado	
jesus nazareno	Blusão bomber	
jesus nazareno	Camisa relaxed estampada	

Figura 63 - Detalhes de Produtos

Nas definições só tem uma *checkBox* onde ativa o programa com o arranque do windows e por final temos o terminar sessão onde e volta para o *form* de Login.



Capítulo IV – Recursos Utilizados

	<p>Visual Studio – Software utilizado para a programação da aplicação.</p>
	<p>Power Point & Office Timeline – Software utilizado na concessão da apresentação e no cronograma</p>
	<p>Word – Utilizado para realizar o relatório</p>
	<p>Opera– Navegador de internet utilizado para pesquisar informação e para esclarecer dúvidas.</p>



 <p>Cisco Packet Tracer</p>	<p>Cisco Packet Tracer – Utilizado para fazer Diagrama da Rede.</p>
 <p>Microsoft® SQL Server™</p>	<p>Microsoft SQL server management studio – Sistema gestor de base dados utilizado na concessão da nossa base de dados</p>
	<p>Git Hub – Plataforma onde se guarda os projetos graças ao <i>Git</i>, que envia tudo para o <i>GitHub</i>. Usamos o <i>GitHub</i> para enviarmos o projeto um para o outro.</p>



Capítulo IV – Cronograma Final e Justificação de desvios

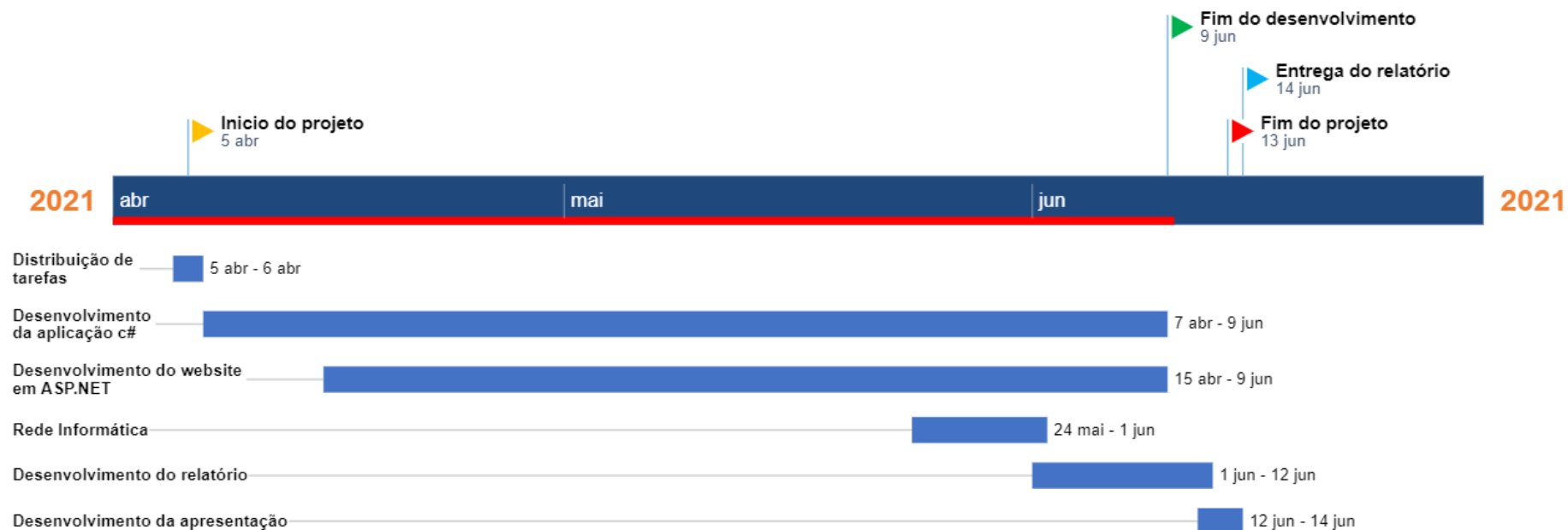


Figura 64 - Cronograma Final



Justificação

O cronograma inicial não foi cumprido por alguma falta de conhecimento e alguns contratempos, assim como a aprendizagem respetiva aos conhecimentos necessários tendo assim terminado a concessão da aplicação e do website no dia 09/06/2021 e a rede informática no dia 01/06/2021, enquanto que o fim do relatório e da apresentação se realizaram no prazo esperado para tal.



Conclusão

Após terminarmos este relatório concluímos que este projeto foi bastante longo, pois eram bastantes coisas a serem feitas durante a realização do mesmo.

É claro que tivemos dificuldade com o projeto pois haviam certas técnicas e conhecimentos que nunca tínhamos usado como a Encriptação da palavra-passe durante o registo do Cliente e a sua descriptação durante o Login, entre outros fatores.

Após terminarmos este projeto sentimo-nos mais aliviados pois numa certa altura estávamos bastantes cansados, não só com o projeto como também com outras disciplinas

É também com os erros que aprendemos e nos tornamos melhores pessoas e, no nosso caso, melhores programadores. Adquirimos bastantes conhecimentos e métodos de trabalho.



Bibliografia e Web Grafia:

GitHub:

- <https://github.com/>

Gmail:

- <https://www.google.com/intl/pt-PT/gmail/about/>

StackOverFlow:

- <https://stackoverflow.com/>

Youtube:

- <http://youtube.com>