



INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

RELATÓRIO DO TRABALHO PRÁTICO

Estrutura de Monitorização da Situação de Alerta e de Contingência

JOÃO AZEVEDO

ALUNO Nº 18845

CARLOS SANTOS

ALUNO Nº 19432

Trabalho realizado sob a orientação de:
Luís Ferreira

Integração de Sistemas de Informação

Licenciatura em Engenharia de Sistemas Informáticos

Barcelos, novembro de 2021

Índice

1	INTRODUÇÃO	1
1.1	Contextualização	2
1.2	Motivação e Objetivos	3
1.3	Estrutura do Documento	3
2	METODOLOGIA	5
2.1	Descrição do Problema	5
2.2	Estrutura de Dados	6
2.2.1	Base de Dados	6
2.2.2	Serviços SOAP	9
2.2.2.1	Gestão de Infetados/Isolados	9
2.2.2.2	Importar ficheiros XML/JSON	11
2.2.3	Cliente	13
2.2.3.1	Gestão de Infetados/Isolados	13
2.2.3.2	Importar Ficheiros Xml/Json	14
2.2.3.3	Dashboard com estatísticas das encomendas	16
2.2.3.4	Dashboard números atuais da pandemia	18
2.2.3.5	Requisição de Produtos	20
2.2.4	Rest Web Api	23
3	CONCLUSÃO	27
	BIBLIOGRAFIA	29
	ANEXOS	31

1 Introdução

1.1 Contextualização

Este documento foi realizado no contexto da unidade curricular de Integração de Sistema de Informação, do curso de Engenharia de Sistemas Informáticos, da Escola Superior de Tecnologia do Instituto Politécnico Cávado do Ave.

Neste Trabalho pretende-se criar soluções para problemas reais, soluções capazes de resolver ou auxiliar o Ser Humano no seu quotidiano.

O Tema deve ter abrangido um conjunto de requisitos capazes de ajudar na gestão de equipas multidisciplinares na visita a lares de idosos.

1.2 Motivação e Objetivos

Após recolha da informação necessária para a realização do projeto, definiram-se os objetivos da solução a desenvolver. Deve-se ter em consideração tópicos bastantes relevantes como:

- ✚ Implementação de uma aplicação cliente que permita demonstrar todos os recursos dos serviços implementados;
- ✚ Identificar casos positivos por covid e as pessoas com quem esteve em contacto;
- ✚ Importar ficheiros Xml ou Json e trabalhar o seu conteúdo;
- ✚ Apresentar produtos mais encomendados pelas equipas;
- ✚ Apresentar as dez equipas mais dispendiosas;
- ✚ Apresentar quantidade de visitas diárias efetuadas e respetivas irregularidades;
- ✚ Apresentar número médio de infetados por covid nos últimos seis meses;
- ✚ Criar um serviço de segurança onde os utilizadores terão de se autenticar de forma segura;
- ✚ Apresentar os números atuais da pandemia;

1.3 Estrutura do Documento

Este documento é constituído por quatro capítulos:

No **Capítulo 1.Introdução**, onde é feita uma breve introdução, motivação, objetivos, e a estrutura do documento.

No **Capítulo 2.Metodologia**, onde é descrito a forma como é resolvido o projeto e especificado as classes e serviços envolvidos.

No **Capítulo 3.Conclusão**, tem a finalidade de retirar conclusões do que foi realizado no projeto e também as aprendizagens deste mesmo.

2 Metodologia

2.1 Descrição do Problema

O projeto consiste na criação de um sistema que permita gerir equipas no auxílio a lares de idosos.

Para isto, é necessário que o sistema permita registar infetados, isolados requerer produtos como luvas e batas, importar dados com informações e gerir todos estes recursos da melhor forma.

Este projeto foi concebido para auxiliar os Municípios a gerir os seus lares de forma eficiente.

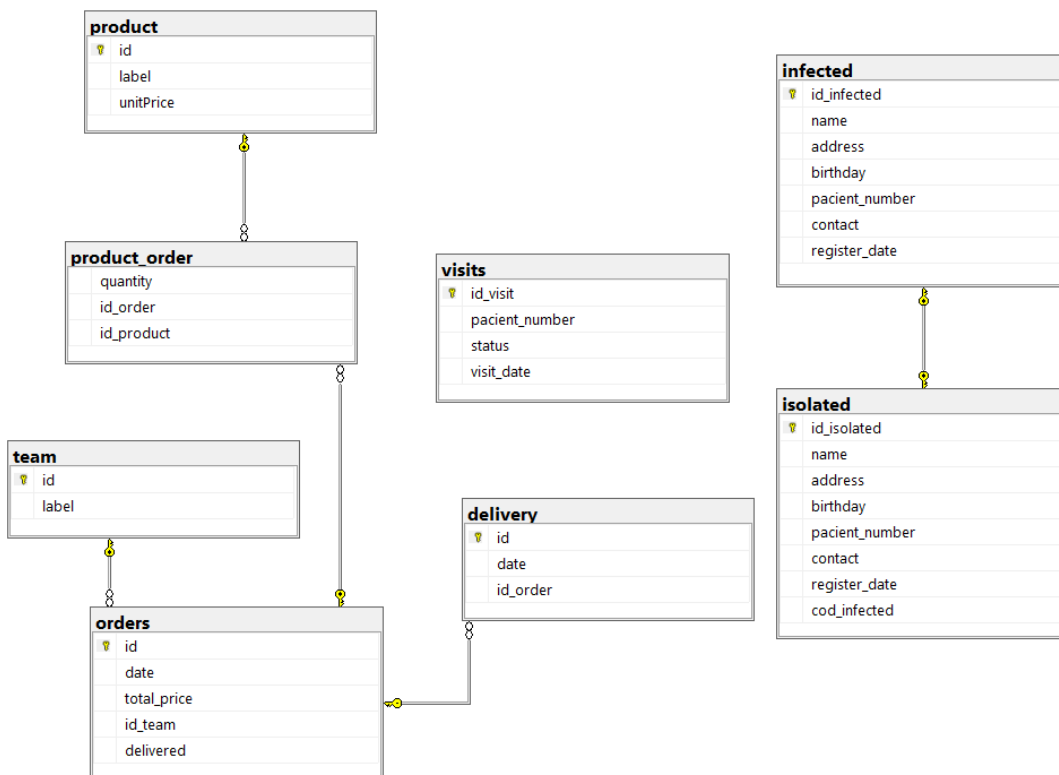
2.2 Estrutura de Dados

2.2.1 Base de Dados

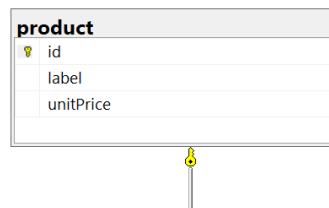
A solução pensada foi a utilização de uma base de dados SQL Server para guardar toda a informação relativa ao sistema.

Para isso criaram-se as seguintes tabelas:

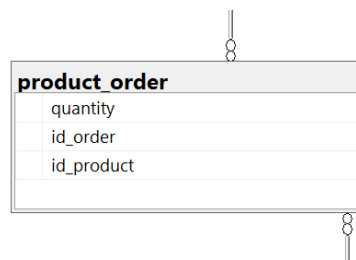
- Product
- Product_Order
- Team
- Orders
- Delivery
- Infected
- Isolated
- Visits



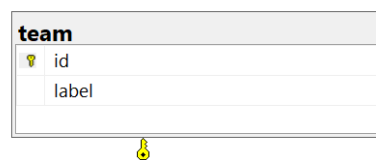
A tabela Product irá armazenar para cada produto de uma determinada encomenda a quantidade de mercadoria que uma equipa pretende, esta tabela possui a quantidade de um determinado produto, a que encomenda se refere e a que produto.



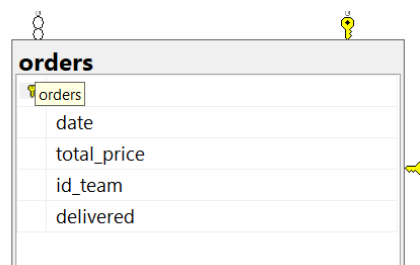
A tabela Product_Order irá armazenar todos os produtos que as equipas possam adquirir ao efetuar a encomenda, esta tabela possui o nome do produto e o seu respetivo preço.



A tabela Team irá armazenar todas as equipas existentes, esta tabela possui o nome da equipa.



A tabela Orders irá armazenar todas as encomendas efetuadas pelas equipas, esta tabela a data em que foi feita a encomenda, o preço total a que equipa corresponde à encomendas e o seu estado de entrega.



A tabela Delivery irá armazenar o estado das encomendas, esta tabela possui a data em que a entrega foi concluída e a que encomenda corresponde.

delivery	
id	
date	
id_order	


A tabela Infected irá armazenar todos os infetados, esta tabela possui o nome da pessoa infetada, morada, data de nascimento, número de saúde, contacto e data em foi registada a infeção por covid.

infected	
id_infected	
name	
address	
birthday	
pacient_number	
contact	
register_date	

A tabela Isolated irá armazenar todas pessoas que estão em isolamento profilático, esta tabela possui o nome da pessoa, morada, data de nascimento, número de saúde, contacto e data em que começou o isolamento.

isolated	
id_isolated	
name	
address	
birthday	
pacient_number	
contact	
register_date	
cod_infected	

A tabela Visits irá armazenar todas as visitas efetuadas e o estado da visita, esta tabela possui o número de saúde do paciente em isolamento, estado da visita e a data em ocorreu a visita.

visits	
	id_visit
	patient_number
	status
	visit_date

2.2.2 Serviços SOAP

2.2.2.1 Gestão de Infetados/Isolados

Com a Base de Dados criada o segundo passo foi implementar os serviços, era pedido que fosse implementado um serviço SOAP para que caso uma equipa deteta-se um caso positivo d covid, este fosse identificado e as pessoas com quem o infetado estivesse em contacto fossem para isolamento profilático.

```
[OperationContract]
/// Registo de um Infetado
1 reference
void RegisterInfected(Infected inf);

[OperationContract]
// Registo Isolados
1 reference
void RegisterIsolated(Isolated iso);
```

```
[DataContract]
// Classe que gere Infetados
4 references
public class Infected : IInfected
{
    [Atributos]
    [Construtor]
    [Propriedades]
}

[DataContract]
// Classe que gere Isolados
4 references
public class Isolated : IIsolated
{
    [Atributos]
    [Construtor]
    [Propriedades]
}
```

Foram criadas as Classes que gerem infetados e Isolados e respetivas interfaces. As funções RegisterInfected e RegisterIsolated vão permitir ao cliente efetuar o registo dos mesmos.

```

/// <summary> Registo de Infetados
1 reference
public void RegisterInfected(Infected inf)
{
    try
    {
        // Registrar na base de dados
        DataSet ds = new DataSet();

        //1º Connection String no Web Config

        string cs = ConfigurationManager.ConnectionStrings["EMSACConnectionString"].ConnectionString;

        //2º OpenConnection
        SqlConnection con = new SqlConnection(cs);

        // 3 Query
        string q = "insert into dbo.infected (name, address, birthday, pacient_number, contact, register_date)" +
            "values(@name, @address, @birthday, @pacient_number, @contact, @register_date)";

        //4º Execute
        SqlDataAdapter da = new SqlDataAdapter(q, con);

        //Instancia parâmetros
        da.SelectCommand.Parameters.Add("@name", SqlDbType.VarChar);
        da.SelectCommand.Parameters["@name"].Value = inf.Name;

        da.SelectCommand.Parameters.Add("@address", SqlDbType.VarChar);
        da.SelectCommand.Parameters["@address"].Value = inf.Address;

        da.SelectCommand.Parameters.Add("@birthday", SqlDbType.DateTime2);
        da.SelectCommand.Parameters["@birthday"].Value = inf.Birthday;

        da.SelectCommand.Parameters.Add("@pacient_number", SqlDbType.VarChar);
        da.SelectCommand.Parameters["@pacient_number"].Value = inf.Pacient_number;

        da.SelectCommand.Parameters.Add("@contact", SqlDbType.VarChar);
        da.SelectCommand.Parameters["@contact"].Value = inf.Contact;

        da.SelectCommand.Parameters.Add("@register_date", SqlDbType.DateTime2);
        da.SelectCommand.Parameters["@register_date"].Value = inf.Register_date;

        da.Fill(ds, "Infetados");
    }
}

```

```

/// <summary> Registo de Isolados
1 reference
public void RegisterIsolated(Isolated iso)
{
    try
    {
        // Registrar na base de dados
        DataSet ds = new DataSet();

        //1º Connection String no Web Config

        string cs = ConfigurationManager.ConnectionStrings["EMSACConnectionString"].ConnectionString;

        //2º OpenConnection
        SqlConnection con = new SqlConnection(cs);

        // 3 Query
        string q = "insert into dbo.isolated (cod_infected, name, address, birthday, pacient_number, contact, register_date)" +
            "values(@codigo, @name, @address, @birthday, @pacient_number, @contact, @register_date)";

        //4º Execute
        SqlDataAdapter da = new SqlDataAdapter(q, con);

        //Instancia parâmetros
        da.SelectCommand.Parameters.Add("@codigo", SqlDbType.VarChar);
        da.SelectCommand.Parameters["@codigo"].Value = iso.Name;

        da.SelectCommand.Parameters.Add("@name", SqlDbType.VarChar);
        da.SelectCommand.Parameters["@name"].Value = iso.Name;

        da.SelectCommand.Parameters.Add("@address", SqlDbType.VarChar);
        da.SelectCommand.Parameters["@address"].Value = iso.Address;

        da.SelectCommand.Parameters.Add("@birthday", SqlDbType.DateTime);
        da.SelectCommand.Parameters["@birthday"].Value = iso.Birthday;

        da.SelectCommand.Parameters.Add("@pacient_number", SqlDbType.VarChar);
        da.SelectCommand.Parameters["@pacient_number"].Value = iso.Pacient_number;

        da.SelectCommand.Parameters.Add("@contact", SqlDbType.VarChar);
        da.SelectCommand.Parameters["@contact"].Value = iso.Contact;

        da.SelectCommand.Parameters.Add("@register_date", SqlDbType.DateTime);
        da.SelectCommand.Parameters["@register_date"].Value = iso.Register_date;

        da.Fill(ds, "Isolados");
    }
}

```

As funções acima tanto para o registo de um infetado como para o registo de um isolado permitem ao receber como parâmetro um infetado/isolado abrir uma conexão na base de dados e com uma query já previamente criada adicionar os dados.

2.2.2.2 Importar ficheiros XML/JSON

Outro aspeto que era necessário ser implementado era um sistema capaz de importar ficheiros Xml/Json para a base de dados.

Foi criado uma função que recebia como parâmetro o ficheiro no formato de string e a sua extensão após a análise da sua extensão era enviada para uma função que trabalhava com o tipo de formato correspondente (Xml ou Json).

```
/// <summary> Importar ficheiros Json/XML
1 reference
public void Relatorioidigital(string file, string extension)
{
    const string json = ".json";
    const string xml = ".xml";
    // Verificar se a extensao e Json/XML
    if (String.Compare(extension, xml) == 0) RegisterXml(file);
    else if (String.Compare(extension, json) == 0) RegisterJson(file);
}
```

Caso a extensão fosse Xml a função RegisterXml executava um conjunto de parâmetros entre os quais conexão a base de dados e inserção dos dados na mesma.

```

/// <summary> Registo apenas de Ficheiro Xml
/// reference
private bool RegisterXml(string file)
{
    try
    {
        // Variaveis
        string mydate = string.Empty;
        string pacient = string.Empty;
        int status = -1;
        string hour = string.Empty;

        // Instanciar um Documento Xml
        XmlDocument docXml = new XmlDocument();
        // Carregar a string para o documento Xml
        docXml.LoadXml(file);

        // Retorno das tags (name , adress, ... de cada isolado)
        foreach (XmlNode xmlNode in docXml.DocumentElement.ChildNodes)
        {
            // Retorna o valor para cada variavel
            pacient = xmlNode.ChildNodes[0].InnerText;
            status = int32.Parse(xmlNode.ChildNodes[1].InnerText);
            mydate = xmlNode.ChildNodes[2].InnerText;
            hour = xmlNode.ChildNodes[3].InnerText;

            // Registrar na base de dados
            DataSet ds = new DataSet();

            //1º ConnectionString no Web Config
            string cs = ConfigurationManager.ConnectionStrings["EMSSACConnectionString"].ConnectionString;

            //2º OpenConnection
            SqlConnection con = new SqlConnection(cs);

            // 3 Query
            string q = "insert into dbo.visits (patient_number, status, visit_date)" +
                "values(@patient_number, @status, @visit_date)";

            //4º Execute
            SqlDataAdapter da = new SqlDataAdapter(q, con);

            //Instancia parâmetros
            da.SelectCommand.Parameters.Add("@patient_number", SqlDbType.VarChar);
            da.SelectCommand.Parameters["@patient_number"].Value = pacient;

            da.SelectCommand.Parameters.Add("@status", SqlDbType.VarChar);
            da.SelectCommand.Parameters["@status"].Value = status;

            da.SelectCommand.Parameters.Add("@visit_date", SqlDbType.DateTime);
            da.SelectCommand.Parameters["@visit_date"].Value = mydate + " " + hour;

            da.Fill(ds, "VisitsXml");
        }
    }
}

```

Caso a extensão fosseJson a função RegisterJson executava um conjunto de parâmetros entre os quais conexão a base de dados e inserção dos dados na mesma.

```

/// <summary> Registo apenas de ficheiro Json
/// reference
private bool RegisterJson(string file)
{
    try
    {
        // Criação de strigs vazias
        string hour = string.Empty;
        string myDate = string.Empty;

        // Instanciar um Documento Json
        JsonDocument jsonDoc = JsonDocument.Parse(file);

        JsonElement root = jsonDoc.RootElement;
        JsonElement isolationsJsonElement = root.GetProperty("isolations");

        // Percorrer todos os objeto no ficheiro Json
        foreach (JsonElement iso in isolationsJsonElement.EnumerateArray())
        {
            // Retorna o valor para cada variavel
            JsonElement patientEl = iso.GetProperty("patient_number");
            String pacient = patientEl.GetString();
            JsonElement statusEl = iso.GetProperty("status");
            Int32 status = statusEl.GetInt32();
            JsonElement dateEl = iso.GetProperty("visit_date");
            DateTime date = dateEl.GetDateTime();

            // Junção dos campo Dia + Mes + Ano
            myDate = date.Day + "/" + date.Month + "/" + date.Year;
            // Junção dos campo Hora + Minuto + Segundo
            hour = date.Hour + ":" + date.Minute + ":" + date.Second;

            // Adicionar a BD
            DataSet ds = new DataSet();

            //1º ConnectionString no Web Config
            string cs = ConfigurationManager.ConnectionStrings["EMSSACConnectionString"].ConnectionString;

            //2º OpenConnection
            SqlConnection con = new SqlConnection(cs);

            // 3 Query
            string q = "insert into dbo.visits (patient_number, status, visit_date)" +
                "values(@patient_number, @status, @visit_date)";

            //4º Execute
            SqlDataAdapter da = new SqlDataAdapter(q, con);

            //Instancia parâmetros
            da.SelectCommand.Parameters.Add("@patient_number", SqlDbType.VarChar);
            da.SelectCommand.Parameters["@patient_number"].Value = pacient;

            da.SelectCommand.Parameters.Add("@status", SqlDbType.VarChar);
            da.SelectCommand.Parameters["@status"].Value = status;

            da.SelectCommand.Parameters.Add("@visit_date", SqlDbType.DateTime);
            da.SelectCommand.Parameters["@visit_date"].Value = date;

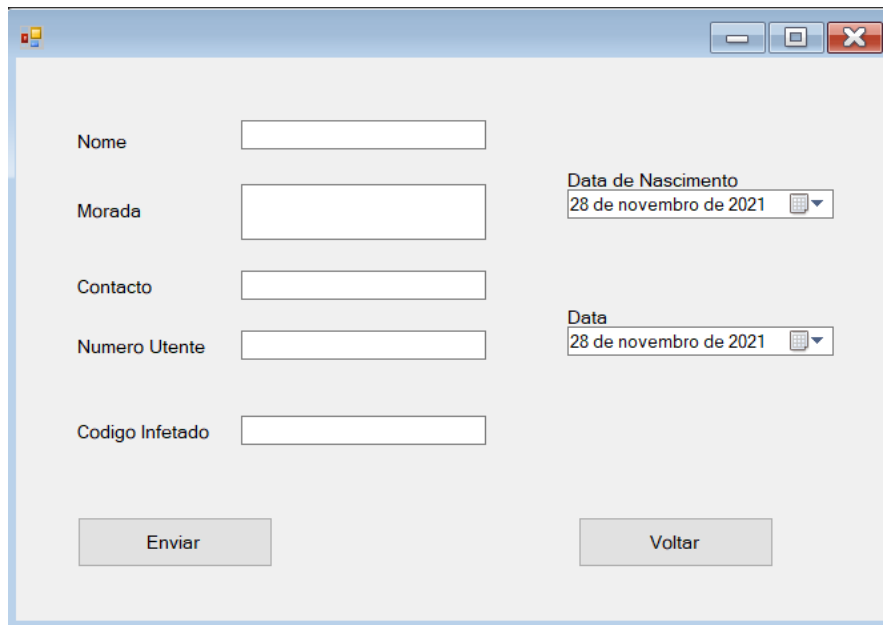
            da.Fill(ds, "VisitsJson");
        }
    }
}

```


2.2.3 Cliente

2.2.3.1 Gestão de Infetados/Isolados

No lado do cliente foi desenvolvido um form com o aspeto seguinte:



```
// Instanciar o serviço
EMSAC.EmsacServiceClient co = new EMSAC.EmsacServiceClient();
// Verificar se estamos a enviar uma Pessoa Infetada ou uma Pessoa Isolada
if (CodigoInfetado.Text.ToString().Length == 0)
{
    // Criar um Infetado
    EMSAC.Infected inf = new EMSAC.Infected();
    inf.Name = Nome.Text;
    inf.Contact = Contacto.Text;
    inf.Address = Morada.Text;
    inf.Pacient_number = NumeroUtente.Text;
    inf.Birthday = DataNascimento.Value;
    inf.Register_date = Data.Value;

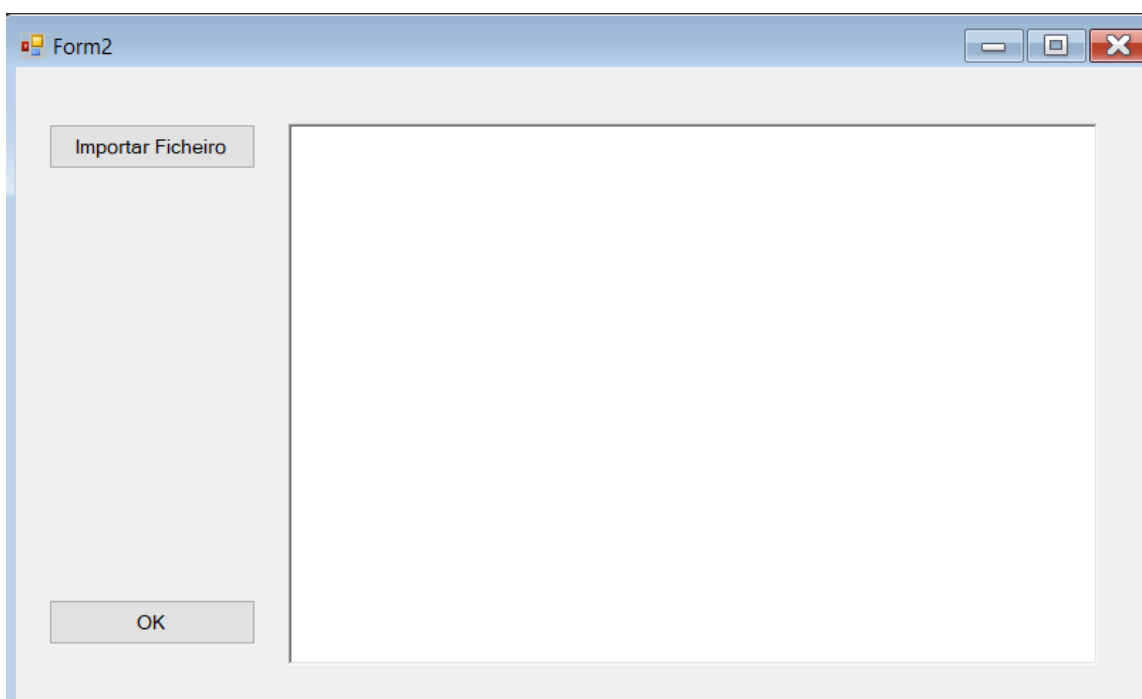
    // Registrar um Infetado
    co.RegisterInfected(inf);
}
else
{
    // Criar um Isolado
    EMSAC.Isolated iso = new EMSAC.Isolated();
    iso.Name = Nome.Text;
    iso.Contact = Contacto.Text;
    iso.Address = Morada.Text;
    iso.Pacient_number = NumeroUtente.Text;
    iso.Birthday = DataNascimento.Value;
    iso.Register_date = Data.Value;
    iso.Cod_infected = CodigoInfetado.Text;

    // Registrar um Isolado
    co.RegisterIsolated(iso);
}
```

Após o cliente inserir os dados relativos a um infetado/isolado, esses mesmos dados serão convertidos numa classe do tipo Infetado/Isolado correspondente. Após essa conversão é enviado para o serviço SOAP e feito o tratamento enunciado em cima.

2.2.3.2 Importar Ficheiros Xml/Json

No lado do cliente foi desenvolvido um form com o aspeto seguinte:



O botão de Importar Ficheiro abre o Explorador de Ficheiros e permite seleccionar o ficheiro que pretendemos enviar. Para além disso faz um conjunto de instruções de verificação como averiguar se o cliente seleccionou algum ficheiro, se o ficheiro existe e se eventualmente superou estas verificações é possível ver o conteúdo do ficheiro.

```

try
{
    // Caso tenha selecionado um ficheiro
    if (file.ShowDialog() == DialogResult.OK)
    {
        // Caminho absoluto do ficheiro
        path = file.FileName;

        // Verificar se existe o ficheiro
        if (File.Exists(path) == true)
        {
            // Ler o conteudo do Ficheiro
            string text = File.ReadAllText(path);
            // Mostra no Form o conteudo do ficheiro
            richTextBox1.Text = text.ToString();
        }
        else
        {
            throw new Exception("Erro: Ficheiro nao possui conteudo!");
        }
    }
}

```

O botão ok permite enviar para o serviço o conteúdo do ficheiro selecionado

```

try
{
    // Mostra a Form1 de novo e envia a informacao relativa a GNR ou PSP
    EMSAC.EmsacServiceClient co = new EMSAC.EmsacServiceClient();

    // Ler o conteudo do ficheiro
    string text = File.ReadAllText(path);

    // Verificar a extensao do ficheiro
    string extension = Path.GetExtension(path);

    // Comparar a extensao
    if (String.Compare(extension, xml) == 0)
    {
        // Caso seja um ficheiro xml
        co.Relatoriodigital(text.ToString(), xml);
    }
    else if (String.Compare(extension, json) == 0)
    {
        // Caso seja um ficheiro json
        co.Relatoriodigital(text.ToString(), json);
    }

    // Fechar o Form2
    this.Close();
}

```

2.2.3.3 Dashboard com estatísticas das encomendas

No lado do cliente foi desenvolvido um form com o aspeto seguinte:

The screenshot shows a Windows application window titled "Form4". Inside the window, the title "Estatísticas" is centered at the top. Below the title, the dashboard is organized into two columns. The left column contains two table placeholders: "Produtos Mais Vendidos" and "Equipas Com Mais Despesas". The right column contains two numerical displays: "Número médio de Visitas Diária nos últimos 30 dias" with the value "0", and "Número Médio de Infetados por Covid nos últimos 6 meses" with the value "0". At the bottom center of the window is a button labeled "Voltar".

Neste form é apresentado em formato de tabelas os produtos mais vendidos e as equipas com mais despesas. Por outro lado, em termos numéricos o número médio de visitas diárias nos últimos 30 dias e o número médio de infetados por covid nos últimos 6 meses.

```

InitializeComponent();
try
{
    // Criação de uma string
    StringBuilder url1 = new StringBuilder();
    // Conteúdo da string
    url1.Append("https://emsacwebapi.azurewebsites.net/orders/get_most_sold_products");

    // Consumo da Api
    HttpWebRequest request = WebRequest.Create(url1.ToString()) as HttpWebRequest;
    using (HttpWebResponse response = request.GetResponse() as HttpWebResponse)
    {
        // Verifica de o serviço esta ON
        if (response.StatusCode != HttpStatusCode.OK)
        {
            // Mensagem de Erro
            string message = String.Format("Get falhou!!");
            throw new ApplicationException(message);
        }

        // Converter Json numa Classe
        DataContractJsonSerializer jsonSerializer = new DataContractJsonSerializer(typeof(List<ProductSelled>));
        object objResponse = jsonSerializer.ReadObject(response.GetResponseStream());
        List<ProductSelled> jsonResponse = (List<ProductSelled>)objResponse;
        i = 0;
        foreach (ProductSelled p in jsonResponse)
        {
            if (i == 5) break;
            var index = dataGridView1.Rows.Add();
            dataGridView1.Rows[index].Cells["label"].Value = p.label;
            dataGridView1.Rows[index].Cells["quantity"].Value = p.quantity;
            i++;
        }
    }
}
catch (ApplicationException exception)
{
    MessageBox.Show(exception.Message);
}
}

```

```

try
{
    // Criação de uma string
    StringBuilder url3 = new StringBuilder();
    // Conteúdo da string
    url3.Append("https://emsacwebapi.azurewebsites.net/orders/get_average_infected");

    // Consumo da Api
    HttpWebRequest request3 = WebRequest.Create(url3.ToString()) as HttpWebRequest;
    using (HttpWebResponse response3 = request3.GetResponse() as HttpWebResponse)
    {
        // Verifica de o serviço esta ON
        if (response3.StatusCode != HttpStatusCode.OK)
        {
            // Mensagem de Erro
            string message = String.Format("Get falhou!!");
            throw new ApplicationException(message);
        }

        // Converter Json numa Classe
        DataContractJsonSerializer jsonSerializer = new DataContractJsonSerializer(typeof(double));
        object objResponse2 = jsonSerializer.ReadObject(response3.GetResponseStream());
        double jsonResponse2 = Math.Round((double)objResponse2, 3);
        label_medioinfetados.Text = jsonResponse2.ToString();
    }
}
catch (ApplicationException exception)
{
    MessageBox.Show(exception.Message);
}
}

```

```

try
{
    EMSAC.EmSACServiceClient c1 = new EMSAC.EmSACServiceClient();
    label_visitasdiarias.Text = c1.GetLastVisits().Visits_count.ToString();
    label_irregularidades.Text = c1.GetLastVisits().Irregularities_percent.ToString();
}
catch (ApplicationException exception)
{
    MessageBox.Show(exception.Message);
}

try
{
    // Criação de uma string
    StringBuilder url2 = new StringBuilder();
    // Conteúdo da string
    url2.Append("https://emsacwebapi.azurewebsites.net/orders/get_most_expensive_teams");

    // Consumo da Api
    HttpRequest request2 = WebRequest.Create(url2.ToString()) as HttpRequest;
    using (HttpWebResponse response2 = request2.GetResponse() as HttpWebResponse)
    {
        // Verifica de o serviço está ON
        if (response2.StatusCode != HttpStatusCode.OK)
        {
            // Mensagem de Erro
            string message = String.Format("Get falhou!!");
            throw new ApplicationException(message);
        }

        // Converter Json numa Classe
        DataContractJsonSerializer jsonSerializer = new DataContractJsonSerializer(typeof(List<TeamCost>));
        object objResponse2 = jsonSerializer.ReadObject(response2.GetResponseStream());
        List<TeamCost> jsonResponse2 = (List<TeamCost>)objResponse2;

        i = 0;
        foreach (TeamCost tc in jsonResponse2)
        {
            if (i == 10) break;
            var index = dataGridView2.Rows.Add();
            dataGridView2.Rows[index].Cells["team"].Value = tc.label;
            dataGridView2.Rows[index].Cells["price"].Value = tc.cost;
            i++;
        }
    }
}
catch (ApplicationException exception)
{
    MessageBox.Show(exception.Message);
}
}

```

2.2.3.4 Dashboard números atuais da pandemia

Outro requisito que o sistema deveria ter era apresentação de uma dashboard com os números atuais da pandemia.

No lado do cliente foi desenvolvido um form com o aspeto seguinte:

Situação Pandémica Portugal	
Casos Ativos	Casos Confirmados
0	0
Óbitos	Internados
0	0
Recuperados	Internados em UCI
0	0

Neste form é apresentado em termos numéricos a situação pandémica em Portugal apresenta-se no form os casos ativos, casos confirmados, óbitos, internados, recuperados e internados em UCI.

```

InitializeComponent();

// Criação de uma string
StringBuilder urlStatus = new StringBuilder();
// Conteúdo da string
urlStatus.Append("https://covid19-api.vost.pt/Requests/get_status");

// Consumo da Api
HttpWebRequest requestStatus = WebRequest.Create(urlStatus.ToString()) as HttpWebRequest;
using (HttpWebResponse responseStatus = requestStatus.GetResponse() as HttpWebResponse)
{
    // Verifica se o serviço está ON
    if (responseStatus.StatusCode != HttpStatusCode.OK)
    {
        // Mensagem de Erro
        string message = String.Format("GET falhou!!");
        throw new ApplicationException(message);
    }

    // Converter objeto num json
    DataContractJsonSerializer jsonSerializer = new DataContractJsonSerializer(typeof(Root));
    object objResponse = jsonSerializer.ReadObject(responseStatus.GetResponseStream());
    Root jsonResponse = (Root)objResponse;

    // Verifica se o serviço está ON
    if (String.Compare(jsonResponse.status, "Server is OK") != 0)
    {
        // Mensagem de Erro
        string message = String.Format("API falhou!!");
        throw new ApplicationException(message);
    }
}

// Criação de uma string
StringBuilder url = new StringBuilder();
url.Append("https://covid19-api.vost.pt/Requests/get_last_update");

// Consumo da Api
HttpWebRequest request = WebRequest.Create(url.ToString()) as HttpWebRequest;
using (HttpWebResponse response = request.GetResponse() as HttpWebResponse)
{
    // Verifica se o serviço está ON
    if (response.StatusCode != HttpStatusCode.OK)
    {
        // Mensagem de Erro
        string message = String.Format("Get falhou!!");
        throw new ApplicationException(message);
    }

    // Converter Json numa Classe
    DataContractJsonSerializer jsonSerializer = new DataContractJsonSerializer(typeof(Root));
    object objResponse = jsonSerializer.ReadObject(response.GetResponseStream());
    Root jsonResponse = (Root)objResponse;
    label_ativos.Text = jsonResponse.ativos.ToString();
    label_confirmados.Text = jsonResponse.confirmados.ToString();
    label_obitos.Text = jsonResponse.obitos.ToString();
    label_internados.Text = jsonResponse.internados.ToString();
    label_recuperados.Text = jsonResponse.recuperados.ToString();
    label_uci.Text = jsonResponse.internados_uci.ToString();
}

```

A seguinte função no cliente tem como objetivo buscar dados do url já conhecido (<https://covid19-api.vost.pt/>) obter os dados pretendidos e mostra-los para a dashboard.

2.2.3.5 Requisição de Produtos

No lado do cliente foi desenvolvido um form com o aspeto seguinte:

Id	Nome	Preço/Unidade
*		

Com a dataGridView é possível mostrar os produtos existentes em stock e sendo assim o cliente pode escolher de entre os vários existentes a sua quantidade associada.

A label ID da Equipa tem como função perceber que equipa se encontra a fazer a encomenda.

Quanto aos botões, no que toca ao de Adicionar Produto este tem como função criar uma lista de produtos do cliente.

```
try
{
    // Rerir a visibilidade da label
    label13.Hide();
    // Rerir a visibilidade do Input
    idequipa.Hide();

    // Criar um ProductOrder
    ProductOrder p1 = new ProductOrder(Int32.Parse(idproduto.Text), Int32.Parse(quantidade.Text));
    // Adicionar a lista o produto criado
    ProductsOrder.Add_ProductOrder(p1);
}
catch (Exception)
{
    throw;
}
```

O botão de Enviar Encomenda envia todos os produtos pedidos pelo cliente, ou seja, vai identificado para o serviço rest a equipa que fez a encomenda, a quantidade de cada produto escolhido e a data em que se efetuou a encomenda.


```

// Criar uma Lista de ProductOrder
List<ProductOrder> lst_nova = new List<ProductOrder>();
// O conteúdo da lista de ProductOrder é o retorno da função
lst_nova = ProductsOrder.Show_ProductOrder();

// Criar uma Encomenda
Order encomenda = new Order(DateTime.Today, Int32.Parse(idequipa.Text), lst_nova);

//// Criação de uma string
StringBuilder url = new StringBuilder();
//// Conteúdo da string
url.Append("https://emsacwebapi.azurewebsites.net/orders/create_new_order");

// // Consumo da Api
HttpWebRequest req = WebRequest.Create(url.ToString()) as HttpWebRequest;
req.Headers.Add("Authorization", "Bearer " + Program.token);
req.Method = "POST";
req.ContentType = "application/json; charset=utf-8";
req.Timeout = 30000;

string jsonString = JsonSerializer.Serialize(encomenda);
req.ContentLength = jsonString.Length;
var sw = new StreamWriter(req.GetRequestStream());
sw.Write(jsonString);
sw.Close();

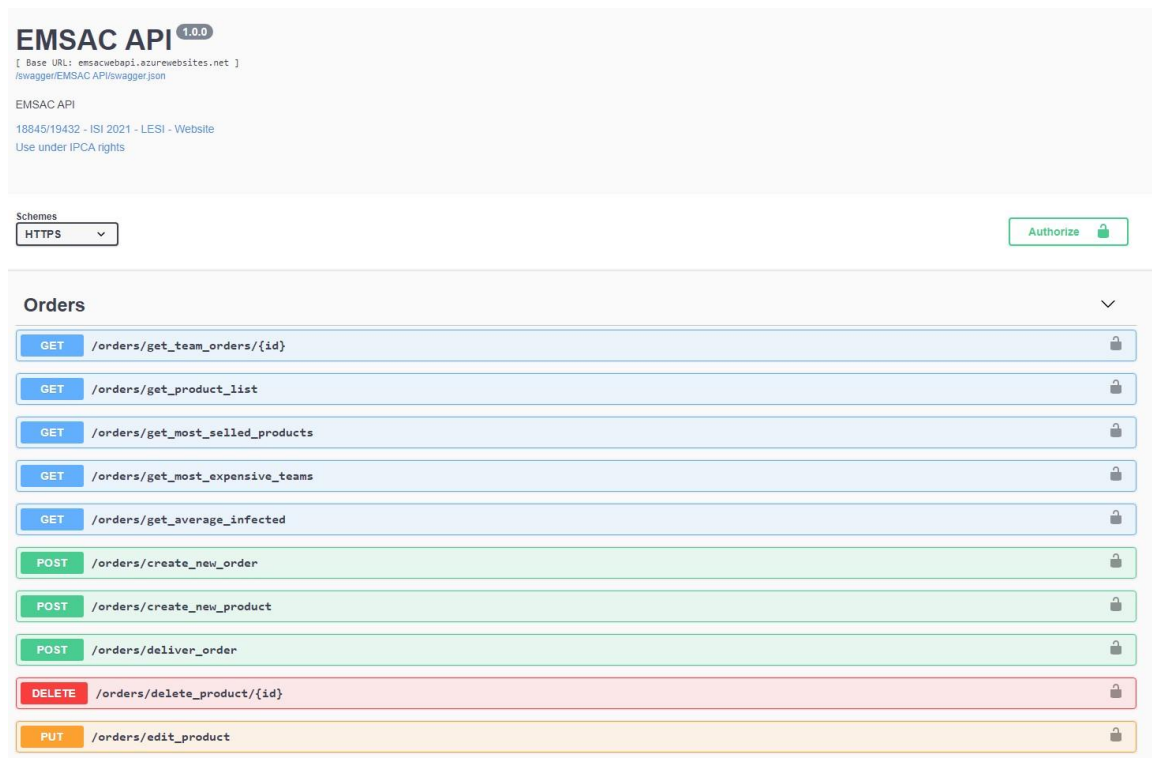
using (HttpWebResponse response = req.GetResponse() as HttpWebResponse)
{
    // Verifica se o serviço está ON
    if (response.StatusCode != HttpStatusCode.OK)
    {
        // Mensagem de Erro
        string message = String.Format("Post falhou!!");
        throw new ApplicationException(message);
    }
}

```

Botões como criar Produto e Consultar Requisições tem como função adicionar um novo produto aos já existentes na Base de Dados que possam ser encomendados em futuras encomendas e consulta de todas as encomendas feitas por uma determinada equipa podendo assim determinar quais já foram entregues ou que estão pendentes.

2.2.4 Rest Web Api

Foi criada uma RestFull Web API com operações CRUD disponíveis para serem utilizadas no âmbito de uma minicentral de compras para que equipas de combate à COVID-19 possam fazer requisições de material auxiliar.



Os métodos criados permitem obter estatísticas acerca de requisições efetuadas, realizar requisições, registar entregas e realizar a manutenção de produtos.

A documentação da API foi disponibilizada através da Extensão Swagger que permite listar os métodos disponíveis e realizar testes com estes. Esta documentação está disponível em <https://emsacwebapi.azurewebsites.net/swagger/>.

Para utilizar os métodos de POST, PUT ou DELETE é necessária autenticação através de OAuth.

```
[Route("orders")]
[ApiController]
1 reference
public class SecurityController : ControllerBase
{
    //classe que gera o JWT
    private readonly IJWTAuthManager jwtAuthManager;

    0 references
    public SecurityController(IJWTAuthManager jwtAuthManager)
    {
        this.jwtAuthManager = jwtAuthManager;
    }

    /// <summary>
    /// Método para Autenticação...não protegido!
    /// </summary>
    /// <param name="loginDetalhes"></param>
    /// <returns></returns>
    [AllowAnonymous]
    [HttpPost("login")]
    0 references
    public AuthResponse Login(AuthRequest loginDetalhes)
    {
        AuthResponse token = jwtAuthManager.Authenticate(loginDetalhes);

        if (token.Token == null)
        {
            token.Token = Unauthorized().ToString();
        }

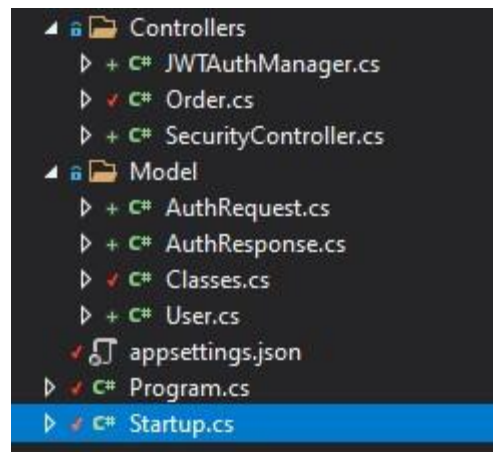
        return token;
    }
}
```

O método POST “login” é permitido em modo “anónimo”, ou seja, sem autenticação, isto para poder realizar a autenticação.

Este método, caso o utilizador e a palavra-passe constem na base de dados, retorna um token com uma validade 40 minutos, que permite ao utilizador enviar a cada método protegido no header do pedido este token como forma de autenticação.

```
Curlcurl -X 'POST' \  
    'https://emsacwebapi.azurewebsites.net/orders/login' \  
    -H 'accept: text/plain' \  
    -H 'Content-Type: application/json' \  
    -d '{  
      "username": "admin"  
      "password": "admin"  
    }'  
  
Request URL  
https://emsacwebapi.azurewebsites.net/orders/login  
  
Server response  
  
CodeDetails  
  
200Response body  
{  
  "name": "admin",  
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpvcS9yZjImlxdWVmbFtZSI6ImNmblu1wibmEiJXoiOjMMPzAeOTU4LjCjElHDQzpqDqczNtgtslndkClNTYzODMtMTkxOH0.L2NelcdPl_MPMAG6DCrRaepubn0vMDICg-B367KPsT",  
  "expiration": "2021-11-30T20:32:38.7233212Z"  
}  
  
Response headers  
content-encoding: gzip  
content-length: 377  
content-type: application/json; charset=utf-8  
date: Tue, 30 Nov 2021 19:52:38 GMT  
server: Microsoft-IIS/10.0  
vary: Accept-Encoding  
x-powered-by: ASP.NET
```

A Web API utilizada funciona sob MVC.



Os métodos passam inicialmente pelo Controlador onde têm as suas configurações, “[HttpGet]” que define a operação CRUD a realizar, definição do tipo de autenticação necessária para aceder ao método e a rota do método em questão. Estas funções chamam as funções do Modelo.

```
[ApiController]
[Route("orders")]
[Authorize]
1 reference
public class OrdersController : Controller
{
    static Orders o;

    0 references
    public OrdersController()
    {
        if (o == null) o = new Orders();
    }

    [HttpGet]
    [AllowAnonymous] // Permite ser chamado sem autenticação
    [Route("get_team_orders/{id}")]
    //orders/get_team_orders/1
    0 references
    public IEnumerable<Order> GetTeamOrders(int id)
    {
        return o.GetTeamOrders(id);
    }

    [HttpGet]
    [AllowAnonymous] // Permite ser chamado sem autenticação
    [Route("get_product_list")]
    //orders/get_product_list
    0 references
    public IEnumerable<Product> GetProductList()
    {
        return o.GetProductList();
    }

    [HttpGet]
    [AllowAnonymous] // Permite ser chamado sem autenticação
    [Route("get_most_sold_products")]
    //orders/get_most_sold_products
    0 references
    public IEnumerable<ProductSelled> GetProductsMostSelled()
    {
        return o.GetProductsMostSelled();
    }
}
```

As funções do Modelo manuseiam os dados, neste caso, sobre base de dados.

```
/// <summary>
/// Obter lista de produtos disponíveis
/// </summary>
/// <returns>Lista de Produtos</returns>
1 reference
public List<Product> GetProductList()
{
    List<Product> list = new();
    try
    {
        //18 ConnectionString
        string cs = "Server=tcp:emsac-isi.database.windows.net,1433;Database=EMSAC;User ID=a19432@emsac-isi;Password=ipca123!;Trusted_Connection=False;Encrypt=True;";

        //28 Conexao a BD
        SqlConnection con = new(cs);

        //38 Query
        string q = "SELECT * FROM product;";

        //48 Cria comando para permitir executar
        SqlCommand co = new(q, con);

        //Executa e lê dados
        con.Open();
        using (SqlDataReader read = co.ExecuteReader())
        {
            while (read.Read())
            {
                list.Add(new Product(int32.Parse(read["id"].ToString()), read["label"].ToString(), float.Parse(read["unitPrice"].ToString())));
            }
            con.Close();
        }
        return list;
    }
    catch (Exception e)
    {
        Trace.WriteLine(e.Message);
        return list;
    }
}
```

3 Conclusão

Na nossa opinião, foi muito interessante o desenvolvimento deste projeto, deu para potencializar a experiência do desenvolvimento de Software, assimilar o conteúdo da unidade curricular, melhorar as capacidades de programação em C#, consolidar conceitos e analisar problemas reais.

Sentimos que agora estamos mais preparados para futuros projetos que nos sejam apresentados, uma vez que este acabou por ser bastante exigente fazendo com que nos tivesse de nos aplicar e melhorar as nossas capacidades.

Com este trabalho adquirimos inúmeras valias que me serão úteis para futuros projetos.

Em suma, abordamos neste trabalho todos os assuntos lecionados nas aulas e graças a isso conseguimos cumprir os objetivos propostos.

Bibliografia

Anexos