Relatório do trabalho prático de Integração de Sistemas de Informação

Curso – Licenciatura em Engenharia de Sistemas Informáticos

Trabalho realizado por:

João Morais – 17214

Luís Esteves - 16960

Barcelos – 22 de dezembro de 2022

Índice de Figuras

Figura 1 - Diagrama Base de dados 6
Figura 2 - Camadas 7
Figura 3 – BLL 8
Figura 4 - Controller Books 8
Figura 5 - Camada DAL 9
Figura 6 - Connection 9
Figura 7 - BookOp 10
Figura 8 - Camada DTO 10
Figura 9 - book 11
Figura 10: Método que realiza a ligação a API externa 12
Figura 11 – SOAP 13
Figura 12 - ASMX 14
Figura 13 -WCF 14
Figura 14: Ligação feita ao Azure 15
Figura 15 - Swagger 16

Conteúdo

Introdução 4
Descrição do Problema 5
Desenvolvimento 6
Base de dados 6
Serviço Rest 7
Arquitetura 7
Camada BLL 8
Camada DAL 9
Camada DTO 10
API Externa 12
SOAP 13
Arquitetura 13
Funcionalidades 13
Cloud 15
Cliente 16
Conclusão 17

Introdução

Este trabalho prático foi realizado no âmbito da disciplina de Integração de Sistemas de Informação, inserida no plano de estudos do curso de Engenharia de Sistemas Informáticos, têm como objetivo explorar a implementação de solução REST e uma solução SOAP, que permite integrar informação, proveniente de diversas fontes numa só base de dados.

O trabalho irá debruçar-se sobre a entrega de livros. Os clientes da plataforma também vão ter que se registar e fazer login para poder realizar pedidos de determinados livros.

Descrição do Problema

Foi escolhido como tema, uma loja que permite a requisição de livros, tem como propósito criar um hábito de leitura á população a custo zero. O nosso objetivo é conseguir simular uma loja de livros.

Tendo em vista que a organização disponibiliza os livros de forma gratuita.

O utilizador para conseguir obter os nossos serviços tem se registar na aplicação, ver o catálogo de livros disponíveis e requisitar os mesmos.

Apos a receção dos livros por parte do utilizador este terá um tempo limite para devolve-lo.

Desenvolvimento

Base de dados

De forma a guardar os dados, fizemos uma base de dados que irá ser a base para a solução REST.

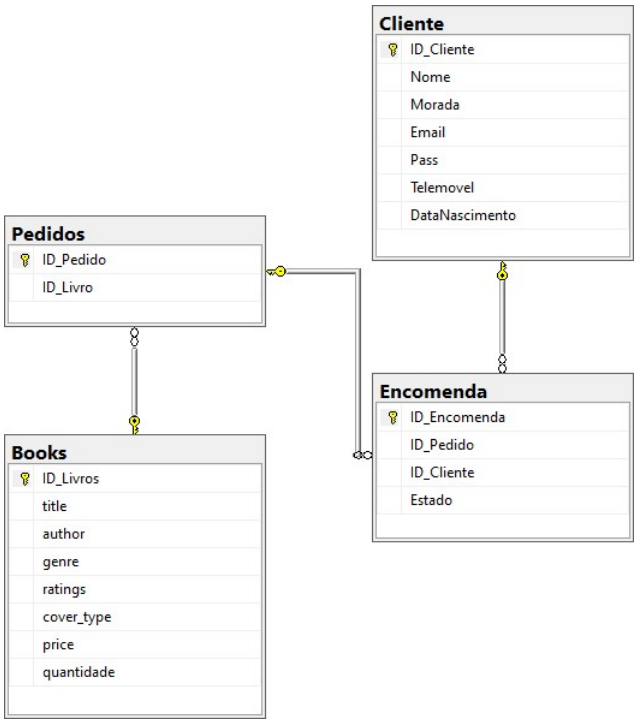


Figura 1 - Diagrama Base de dados

Serviço Rest

Arquitetura

Na estruturação do projeto em camadas, decidi-mos usar uma aplicação em 3 camadas:

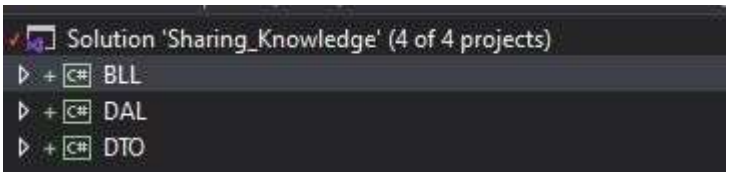


Figura 2 - Camadas

- Camada BLL (Business Logic Layer): serve para escrever, para determinar, os métodos que possibilitam o bom funcionamento das outras camadas.
- Camada DAL (Data Acess Layer): serve como a biblioteca que acede ao banco de dados e todas as operações da manipulação da mesma.
- Camada DTO (Data Transfer Object): serve para definir os diferentes tipos de objetos que vão ser utilizados ao longo do trabalho.

Camada BLL

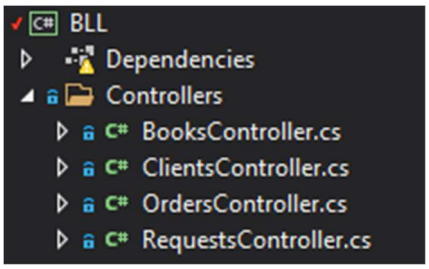


Figura 3 – BLL

Nesta camada inserimos os Controladores. Aqui irão ser aplicados os métodos criados na camada DAL e vão ser criados os métodos CRUD.

```
* Disciplina: Integração de Sistemas de Informação;
* Projecto II;
* Propósito do trabalho: Criar uma API REST Full de gerência de utilizadores e de entrega de livros;
*
*/
namespace BLL.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class BooksController : ControllerBase
    {
        private readonly MockBookRepo _repository = new MockBookRepo();
        //GET api/Books
        //Esta função realiza a operação GET, buscando todos os dados de todos os livros;
        [HttpGet]
        public ActionResult<IEnumerable<Book>> GetAllBooks()
        {
            var booksItems = _repository.GetAllBooks();
            if (booksItems != null) return Ok(booksItems);
            else
            {
                return NotFound();
            }
        }

        //GET api/Books/best-sellers/10
        //Método que permite assesar a uma outra API e receber os best-sellers ditos pela NYT com um determinado preço;
        [HttpGet("best-sellers/{price}")]
        public ActionResult GetBooks(int price)
        {
           var bestSellers = _repository.GetBestSellers(price);
           if (bestSellers == null) return NotFound();
           else
           {
               return Ok(bestSellers);
           }

        }

        //GET api/Books/best-sellers/title
        //Método que permite assesar a uma outra API e receber os best-sellers ditos pela NYT com um determinado autor;
        [HttpGet("best-sellers/byAuthor/{author}")]
        public ActionResult GetBooks2(string author)
        {
            var bestSellers = _repository.GetBestSellers2(author);
            if (bestSellers == null) return NotFound();
            else
            {
                return Ok(bestSellers);
            }

        }
```

Figura 4 - Controller Books

Na imagem 4, podemos observar um exemplo de alguns controladores. Neste caso, que manipulam os dados acerca de livros.

Camada DAL

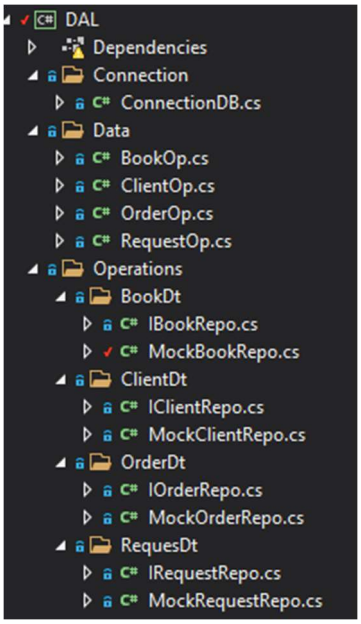
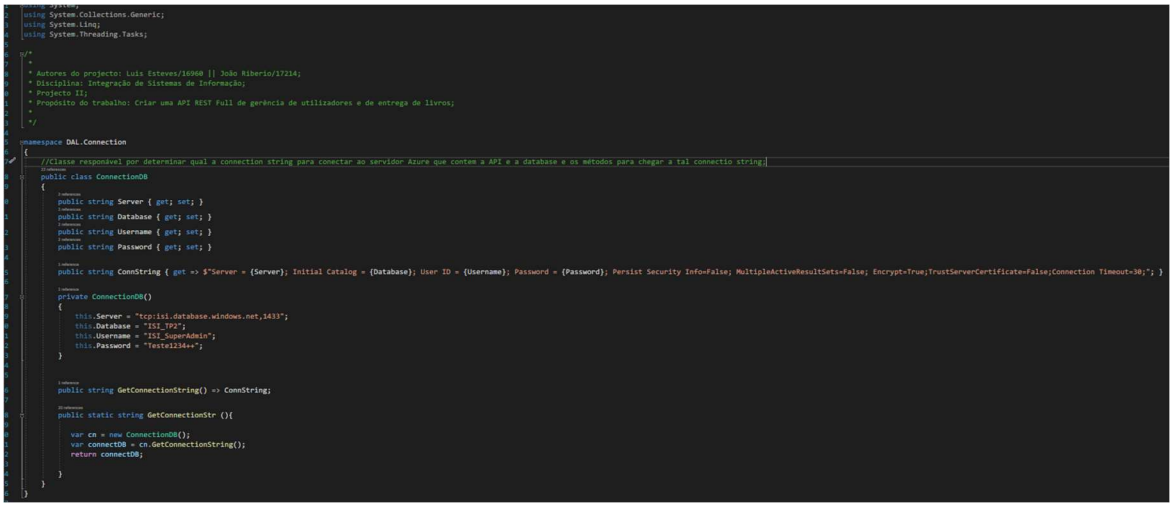


Figura 5 - Camada DAL

Nesta camada, realizamos todo o tipo de operações que manipulam com a base de dados. Dividimos em 3 pastas. A primeira, a Connection, contêm a connection string que liga a Azure. A Data é responsável por manipular diretamente a database e por fim a Operations permite utilizar os dados recebidos na pasta Data e manipulá-los caso seja necessário.



```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

/*
 * Autores do projecto: Luis Esteves/16960 || João Ribeiro/17214;
 * Disciplina: Integração de Sistemas de Informação;
 * Projecto II;
 * Propósito do trabalho: Criar uma API REST Full de gerência de utilizadores e de entrega de livros;
 *
 */

namespace DAL.Connection
{
    //Classe responsável por determinar qual a connection string para conectar ao servidor Azure que contem a API e a database e os métodos para chegar a tal connectio string;
    public class ConnectionDB
    {
        public string Server { get; set; }
        public string Database { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }

        public string ConnString { get => $"Server = {Server}; Initial Catalog = {Database}; User ID = {Username}; Password = {Password}; Persist Security Info=False; MultipleActiveResultSets=False; Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;"; }

        private ConnectionDB()
        {
            this.Server = "tcp:isi.database.windows.net,1433";
            this.Database = "ISI_TP2";
            this.Username = "ISI_SuperAdmin";
            this.Password = "Teste1234++";
        }


        public string GetConnectionString() => ConnString;

        public static string GetConnectionStr (){

           var cn = new ConnectionDB();
           var connectDB = cn.GetConnectionString();
           return connectDB;

        }
    }
}
```

Figura 6 - Connection

```
public static bool UpdateBook(Book book)
{
    int x = 0;
    string mainConnection = ConnectionDB.GetConnectionStr();
    using (SqlConnection sqlConnection = new SqlConnection(mainConnection))
    {
        sqlConnection.Open();
        using (SqlCommand cmd = new SqlCommand("UPDATE Books SET ID_Livros=@ID_Livros, Title=@Title, Author=@Author, Genre=@Genre, Ratings=@Ratings, Cover_Type=@Cover_Type, Price=@Price, Quantidade=@Quantidade WHERE ID_Livros=@ID_Livros", sqlConnection))
        {
            cmd.CommandType = CommandType.Text;
            cmd.Parameters.AddWithValue("@ID_Livros", book.ID_Livros);
            cmd.Parameters.AddWithValue("@Title", book.Title);
            cmd.Parameters.AddWithValue("@Author", book.Author);
            cmd.Parameters.AddWithValue("@Genre", book.Genre);
            cmd.Parameters.AddWithValue("@Ratings", book.Ratings);
            cmd.Parameters.AddWithValue("@Cover_Type", book.Cover_type);
            cmd.Parameters.AddWithValue("@Price", book.Price);
            cmd.Parameters.AddWithValue("@Quantidade", book.Quantidade);
            cmd.ExecuteNonQuery();
            x++;
        }
    }
    if (x == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}

public static bool DeleteBook(int id)
{
    List<Book> b = new List<Book>();
    int x = 0;
    string mainConnection = ConnectionDB.GetConnectionStr();
    using (SqlConnection conn = new SqlConnection(mainConnection))
    {
        conn.Open();
        using (SqlCommand cmd = new SqlCommand("Delete from dbo.Books where ID_Livros=@Id", conn))

        {

            cmd.Parameters.AddWithValue("@Id", id);
            cmd.CommandType = CommandType.Text;
            cmd.ExecuteNonQuery();
            x++;

        }
    }
```

Figura 7 - BookOp

Na imagem 7, podemos ver o exemplo de algumas funções que manipulam a database, que neste caso, manipulam dados acerca de Livros.

Camada DTO

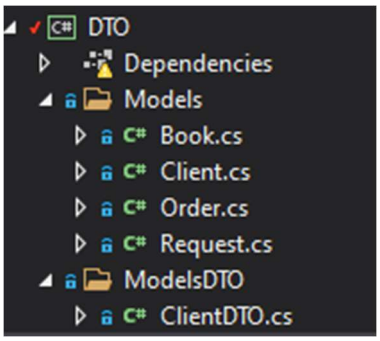
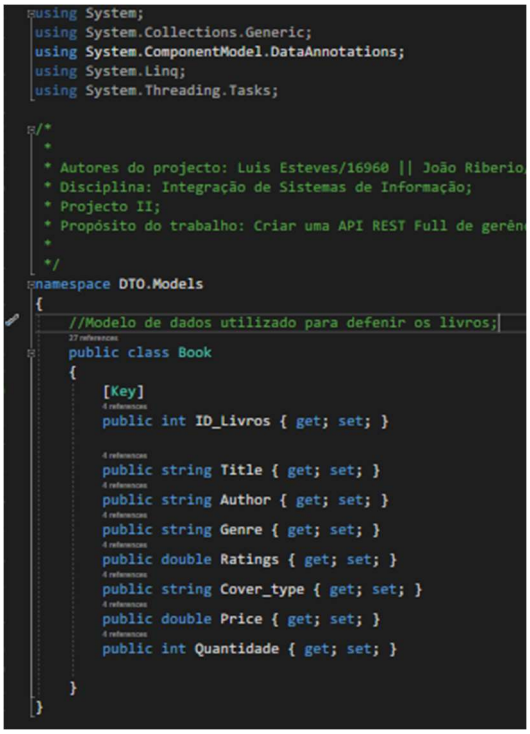


Figura 8 - Camada DTO

Nesta camada, por final, é onde vamos definir os tipos de objetos que vamos utilizar. Ou seja, transpor em objetos os tipos de dados que temos na database.

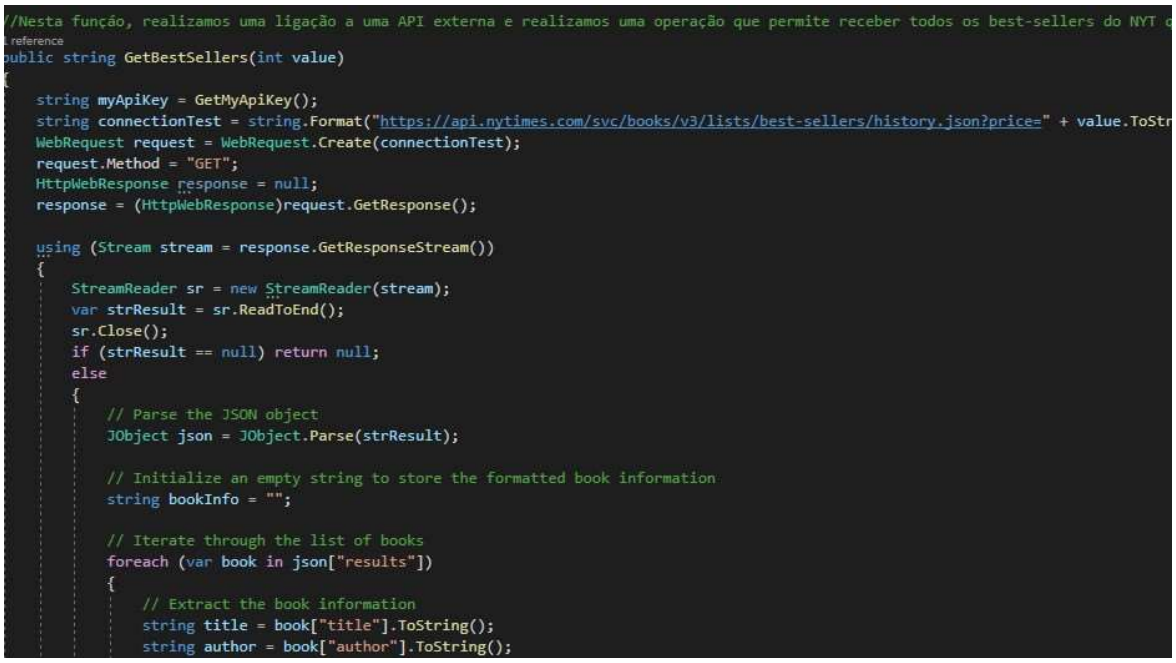


```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

/*
 *
 * Autores do projecto: Luis Esteves/16960 || João Riberio
 * Disciplina: Integração de Sistemas de Informação;
 * Projecto II;
 * Propósito do trabalho: Criar uma API REST Full de gerên
 *
 */
namespace DTO.Models
{
    //Modelo de dados utilizado para defenir os livros;
    27 references
    public class Book
    {
        [Key]
        4 references
        public int ID_Livros { get; set; }

        4 references
        public string Title { get; set; }
        4 references
        public string Author { get; set; }
        4 references
        public string Genre { get; set; }
        4 references
        public double Ratings { get; set; }
        4 references
        public string Cover_type { get; set; }
        4 references
        public double Price { get; set; }
        4 references
        public int Quantidade { get; set; }

    }
}
```

Figura 9 - book

Definimos cada uma das classes da seguinte maneira, como é exemplo a figura 9.

API Externa



```
//Nesta função, realizamos uma ligação a uma API externa e realizamos uma operação que permite receber todos os best-sellers do NYT d
1 reference
public string GetBestSellers(int value)
{
    string myApiKey = GetMyApiKey();
    string connectionTest = string.Format("https://api.nytimes.com/svc/books/v3/lists/best-sellers/history.json?price=" + value.ToStr
    WebRequest request = WebRequest.Create(connectionTest);
    request.Method = "GET";
    HttpWebResponse response = null;
    response = (HttpWebResponse)request.GetResponse();

    using (Stream stream = response.GetResponseStream())
    {
        StreamReader sr = new StreamReader(stream);
        var strResult = sr.ReadToEnd();
        sr.Close();
        if (strResult == null) return null;
        else
        {
            // Parse the JSON object
            JObject json = JObject.Parse(strResult);

            // Initialize an empty string to store the formatted book information
            string bookInfo = "";

            // Iterate through the list of books
            foreach (var book in json["results"])
            {
                // Extract the book information
                string title = book["title"].ToString();
                string author = book["author"].ToString();
```

Figura 10: Método que realiza a ligação a API externa

A Sharing_Knowledge consegue comunicar com uma API externa que está no https://developer.nytimes.com/. Aqui, conseguimos realizar duas operações GET que nos permitem reunir dados dos best-seller do ano, informação reunida pela a NYT.

SOAP

Arquitetura

Para o serviço SOAP, utilizamos uma única camada. Foram encontradas algumas dificuldades em implementar em múltiplas camadas, por isso foi optado por colocar os serviços numa única camada.

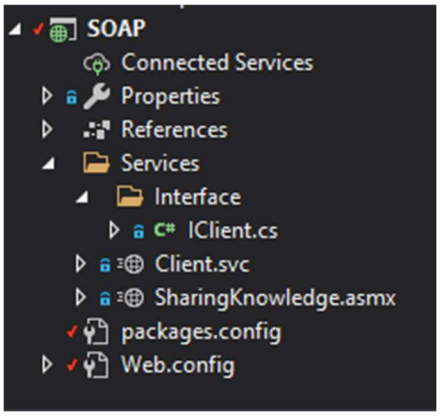


Figura 11 – SOAP

Funcionalidades

As funcionalidades do serviço são simples. As funcionalidades utilizadas foram:

- Listar
- Inserir
- Delete
- Update

Foram realizadas estas opções para os livros, clientes, encomendas e pedidos.

Figura 12 - ASMX

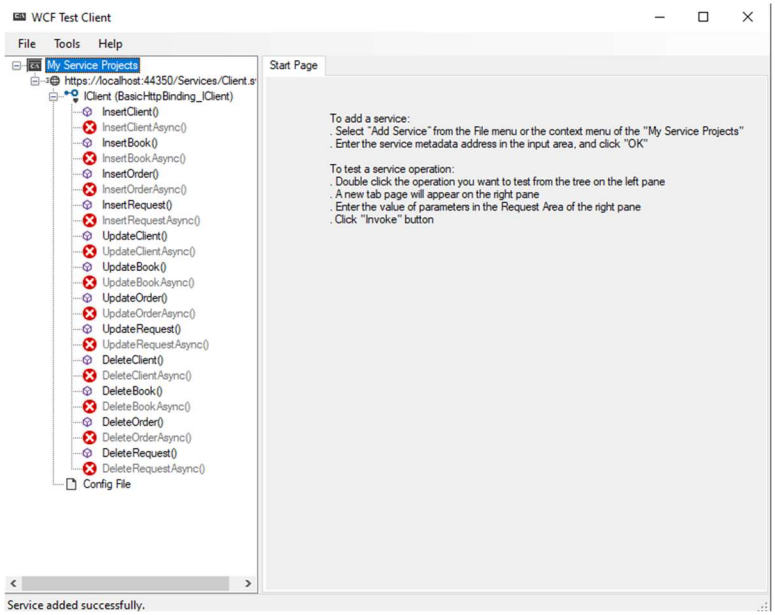


Figura 13 -WCF

Cloud

Para que esteja mais próximo da realidade, uma base de dados alojada localmente numa máquina local não é de todo útil se se tratar de um trabalho em equipa, portanto a base de dados em questão e a API vão ser alojadas na *Cloud* através da ferramenta AZURE.

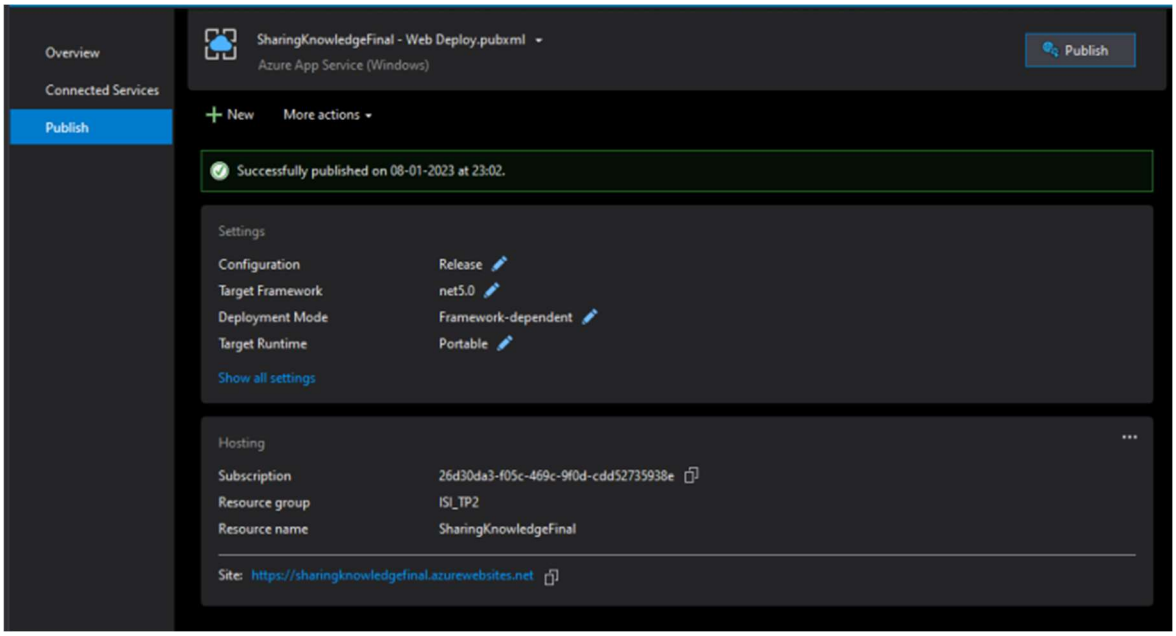


Figura 14: Ligação feita ao Azure

Cliente

Para testar os serviços da API vamos utilizar como ferramentas o Swagger.

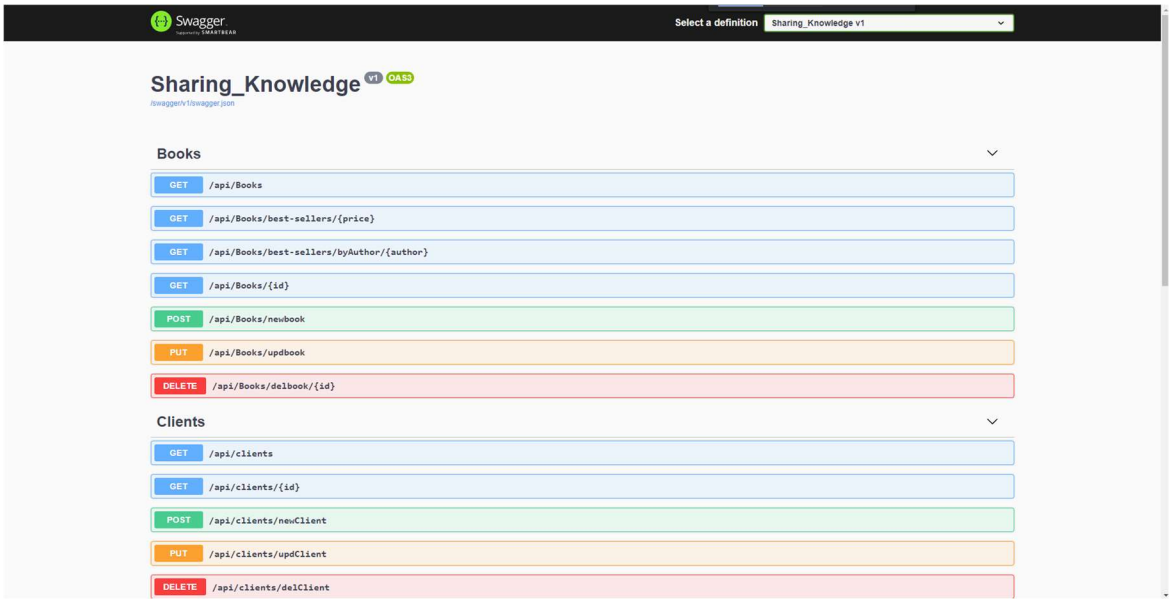


Figura 15 - Swagger

Conclusão

Em jeito de conclusão, o trabalho realizado foi vantajoso para a unidade curricular, uma vez que permitiu pôr em prática a matéria na disciplina de uma maneira mais prática e num contexto real.

O trabalho permitiu ainda melhorar e adquirir conhecimentos relativos aos processos de

De um modo geral, posso dar este trabalho como concluído, pois foram atingidos quase todos os objetivos propostos.