UNIVERSIDADE
LUSÓFONA

Parque de Estacionamento Back End

Trabalho Final de curso

Relatório Final

Nome do Aluno: Jaime Pedro Baptista Germano

Nome do Orientador: Professor Auxiliar João Pedro Leal Abalada de Matos Carvalho

Trabalho Final de Curso | LEI | 21/07/2022

Direitos de cópia

(*Parque de estacionamento – Back End*), Copyright de Jaime Pedro Baptista Germano, ULHT.

A Escola de Comunicação, Arquitetura, Artes e Tecnologias da Informação (ECATI) e a Universidade Lusófona de Humanidades e Tecnologias (ULHT) têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Resumo

Cada vez mais os alunos e funcionários da universidade Lusófona pretendem utilizar o parque de estacionamento da ULHT para estacionar as suas viaturas, tornando-se evidente que em períodos de maior fluxo, para aceder a este parque, os seus utentes teriam de aguardar algum tempo à entrada deste. Isto porque a identificação dos utilizadores, para validar a sua autorização de acesso, é feita manualmente pelos seguranças, o que torna processo um pouco moroso.

O objetivo deste projeto é desenvolver a aplicação de Back End de uma solução para automatizar o acesso ao parque de estacionamento da ULHT.

Esta solução é composta pelo Back End, Front End (mobile e web) e Detetor de Matriculas e pretende utilizar, respetivamente, estes componentes para: Aceder à base de dados e ser o elo de comunicação entre as várias aplicações; servir de interface para os utilizadores ("clientes" do parque, seguranças e administradores do sistema); e câmaras para detetar as matriculas dos automóveis que pretendem aceder ao parque.

Trata-se de uma aplicação centralizada e intermedia que, mais concretamente que o descrito acima, permitirá não só, mas também, autenticar os utilizadores, validar autorizações de entrada no parque, disponibilizar os resultados das consultas à Base de Dados às aplicações do Front End, entre outras funcionalidades.

Palavras-chave: Back End, Front End, Deteção de Matriculas, gRPC, C#, SQL, Stored Procedures.

Abstract

More and more students and employees of the Lusófona University intend to use the ULHT car park to park their vehicles, making it evident that in periods of greater traffic, to access this car park, its users would have to wait for some time at entry of this. This is because the identification of users, to validate their access authorization, is done manually by the security guards, which makes the process a little time consuming.

The objective of this project is to develop the Back End application of a solution to automate access to the ULHT car park.

This solution comprises the Back End, Front End (mobile and web) and Car License Plates Detector and aims to use, respectively, these components to: Access the database and be the communication link between the various applications; serve as an interface for users ("clients" of the park, security guards and system administrators); and cameras to detect the license plates of the cars that want to access the parking lot.

It is a centralized and intermediary application that, more specifically than the one described above, will not only allow but also authenticate users, validate authorizations to enter the park, make the results of consultations to the Database to Front End applications, among other features.

Keywords: Back End, Front End, Car Licence Plates Detection, gRPC, C#, SQL, Stored Procedures.

Índice

Lista de Figuras

1 Identificação do Problema

Hoje em dia a ULHT não detém um sistema automático de deteção de utilizadores autorizados a estacionar as suas viaturas no parque de estacionamento desta universidade, um registo de quantos automóveis estão no parque num dado momento ou ainda um registo de que viaturas entraram e saíram do parque.

Atualmente, um utilizador deste estacionamento tem de parar a viatura ao lado da cabine do segurança e identificar-se. Após esta identificação ao segurança, este vai consultar uma lista, onde tenta encontrar a identificação deste utilizador e verificar se este tem autorização para estacionar.

Todo este processo é moroso e ineficiente especialmente pelo tempo que demora a validar as autorizações de acesso.

No caso de existir um maior fluxo de utentes que pretendam ter acesso ao parque, pode formar-se uma grande fila devido a este método de verificação, podendo causar diversos transtornos como por exemplo, possíveis atrasos de professores e alunos às aulas.

2 Viabilidade e Pertinência

Uma das formas de solucionar este problema, passa pela automatização de todo este processo, desde a identificação dos utilizadores através de uma aplicação, como das próprias viaturas (através da identificação da matricula), validando automaticamente se estão autorizados a entrar no parque ou não, minimizando assim a necessidade de intervenção humana.

Esta solução permite não só agilizar todo este processo de identificação/autorização de estacionamento como também permite identificar quantas viaturas estão no parque, quais os seus proprietários/condutores. Quantos lugares permanecem disponíveis entre outras.

Será ainda possível no futuro, no entanto fora do objetivo deste projeto, implementar estatísticas com base nos dados da utilização do parque, como por exemplo quais os dias ou horas com mais afluência.

A implementação desta solução é de baixo custo para a ULHT uma vez que é apenas necessário investir em alguns equipamentos:

- Um computador para alojar a base de dados e instalação das aplicações de Manutenção, Deteção de Matriculas e Back End;
- Duas câmaras para a deteção das matriculas;
- Um router para a implementação de uma rede Wi-Fi local, onde os utilizadores se possam autenticar para a devida identificação dos mesmos.
- Sensores de “passagem”, que indiquem se a viatura já entrou no parque para que a cancela não se feche no momento em que esta está a passar.

A ULHT não incorrerá em quaisquer custos com equipamento de identificação/acesso, tais como leitores de cartões ou outros, uma vez que existirá uma aplicação para mobile que fará isso mesmo e que irá correr nos telemóveis dos utilizadores.

Esta solução poderá ser implementada em qualquer parque de estacionamento que seja gerido nos mesmo moldes que o parque da ULHT.

3 Levantamento e Análise dos Requisitos

Para a elaboração deste projeto foi necessário proceder ao levantamento e análise dos requisitos de forma a ter uma noção, o mais real possível, do que seria necessário para a sua implementação.

3.1 Requisitos Funcionais

ID	Requisito	Descrição	Imple-mentado
RF1	Comunicar com o Front End.	O Back End deve aceitar pedidos de comunicação das aplicações do Front End	Sim
RF2	Autenticação de utilizadores.	Deve ser capaz de receber um pedido de autenticação dos utilizadores, de qualquer uma das apps do Front End (mobile ou plataforma de gestão), e retornar o resultado.	Sim
RF3	Permitir alterar Password's.	Deve aceitar os pedidos de alteração da password por parte dos utilizadores das apps do Front End e regista-la na base de dados.	Sim
RF4	Pedidos para abrir as cancelas.	Deve aceitar os pedidos dos utentes para entrar ou sair do parque de estacionamento e registar a hora e data desse pedido.	Sim
RF5	Inserir/alterar utilizadores.	Inserir ou alterar os dados dos utilizadores, enviados pela app web do Front End, na base de dados.	Sim
RF6	Inserir/alterar o registo de pagamento.	Inserir ou alterar os registos de pagamento dos utentes enviados pela app web do Front End, na base de dados.	Sim
RF7	Revalidar matricula.	Permitir verificar uma matricula novamente a pedido da app web do Front End no caso da matricula detetada pela Deteção de Matriculas não corresponder à matricula da foto.	Sim
RF8	Guardar fotografias das matriculas.	Guardar uma copia das fotografias das matriculas enviadas pela app da Deteção de Matriculas.	Sim

RF9	Enviar link da fotografia.	Deve enviar para a aplicação web do Front End um link para a fotografia da matricula que fez o pedido de entrada ou saída.	Sim
RF10	Comunicar com a Deteção de Matriculas.	O Back End deve estabelecer comunicação com a app da Deteção de Matriculas.	Sim
RF11	Pedir matricula e fotografia.	Pedir uma foto e a respetiva matricula detetada à app da Deteção de Matriculas, cada vez que recebe um pedido de entrada ou saída do parque.	Sim

3.2 Requisitos Não Funcionais

ID	Requisito	Descrição	Imple-mentado
RNF1	"Correr" numa rede local.	Deve "correr" numa rede local para que a aplicação dos dispositivos moveis possa contactar o Back end.	Sim1
RNF2	Base de Dados	Será necessária uma base de dados para guardar a informação.	Sim
RNF3	Acesso à base de dados reservado.	O Back end deverá ser a única aplicação a poder aceder diretamente à informação contida na base de dados.	Sim

4 Solução Desenvolvida

4.1 Armazenamento da informação

A solução encontrada e implementada neste TFC para armazenar a informação foi a criação de uma base dados, com recurso ao Microsoft SQL Server Express, onde é guardada toda a informação dos utentes, viaturas e respetivo registo de entradas e saídas do parque de estacionamento.

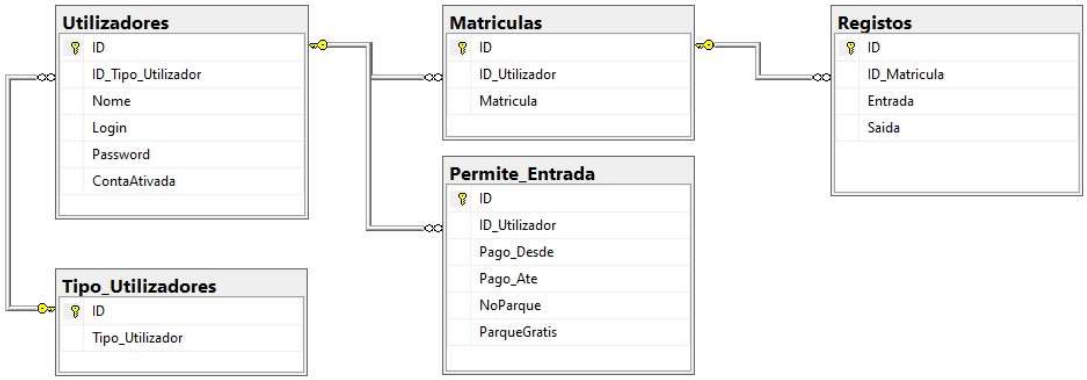

Figura 1 - Diagrama da estrutura de dados

Na Figura 1 pode-se observar a estrutura de dados da base de dados que é composta por apenas 5 tabelas. Não é armazenada muita informação relativamente aos utilizadores tal como se é aluno, professor ou funcionário. Foi desenhada desta forma caso mais tarde a universidade a queira unir à base de dados já existente na instituição.

Cada uma destas tabelas é constituída por um identificador único (ID), gerado automaticamente na introdução de um novo registo. Todas as tabelas têm uma relação de "um para muitos" com exceção da relação entre a tabela de Utilizadores e a tabela Permite_Entrada que é de "um para um". Os dados contidos nestas duas tabelas poderiam estar numa única tabela caso este projeto fosse standalone. Como existe a possibilidade de mais tarde integrar a base de dados desta solução na base de dados geral da Lusófona, manteve-se os dados a guardar separados nestas duas tabelas para permitir mais facilmente essa integração.

Na tabela de Tipo_Utilizadores são guardados os tipos de utilizadores existentes, Admin, Segurança e Utilizador.

Na tabela Utilizadores são guardados os dados referentes à conta do utilizador como, login, password etc.

Já na tabela Permite_Entrada são guardados os dados referentes às autorizações de entrada no parque, no caso de ter direito a parque grátis, datas entre as quais o pagamento está valido ou se o utilizador já se encontra no parque de estacionamento.

A tabela Matriculas contém todas as matriculas associadas a cada um dos utilizadores.

Por fim, é na tabela Registos que são registadas todas as entradas e saídas dos utilizadores.

Para o acesso e manipulação da informação na base de dados recorreu-se a stored procedures em virtude das várias vantagens que esta abordagem apresenta, nomeadamente:

- A camada de segurança entre a base de dados e a interface dos utilizadores.
- Redução da largura de banda da rede caso se pretenda, no futuro, colocar a base de dados noutro servidor que não aquele onde a aplicação do Back End está a correr.
- Eliminação de possíveis bugs nas queries de SQL caso estas fossem criadas no código, devido à dificuldade em validá-las rapidamente.

4.2 Desenvolvimento da aplicação

A aplicação de Back End foi desenvolvida na linguagem C#, e serve de "ponte" entre o Front End (aplicação móvel para os utilizadores do parque e aplicação de gestão do sistema para os administradores e seguranças) e a Deteção de Matriculas, sendo estes últimos desenvolvidos noutros TFC.

Esta aplicação é a responsável por autenticar e registar toda a informação enviada pelas outras aplicações desta solução na base de dados.

4.3 Comunicação entre as diversas aplicações da solução

A comunicação entre as varias aplicações, para troca de dados e autenticações/validações, é feita recorrendo ao gRPC (Google Remote Procedure Call), que foi desenvolvido para Google em 2015.

Esta framework usa ficheiros proto que funcionam como um dicionário, o que nos permitiu estabelecer as comunicações entre os diversos projetos, em que quase todas as aplicações foram desenvolvidas usando linguagens diferentes.

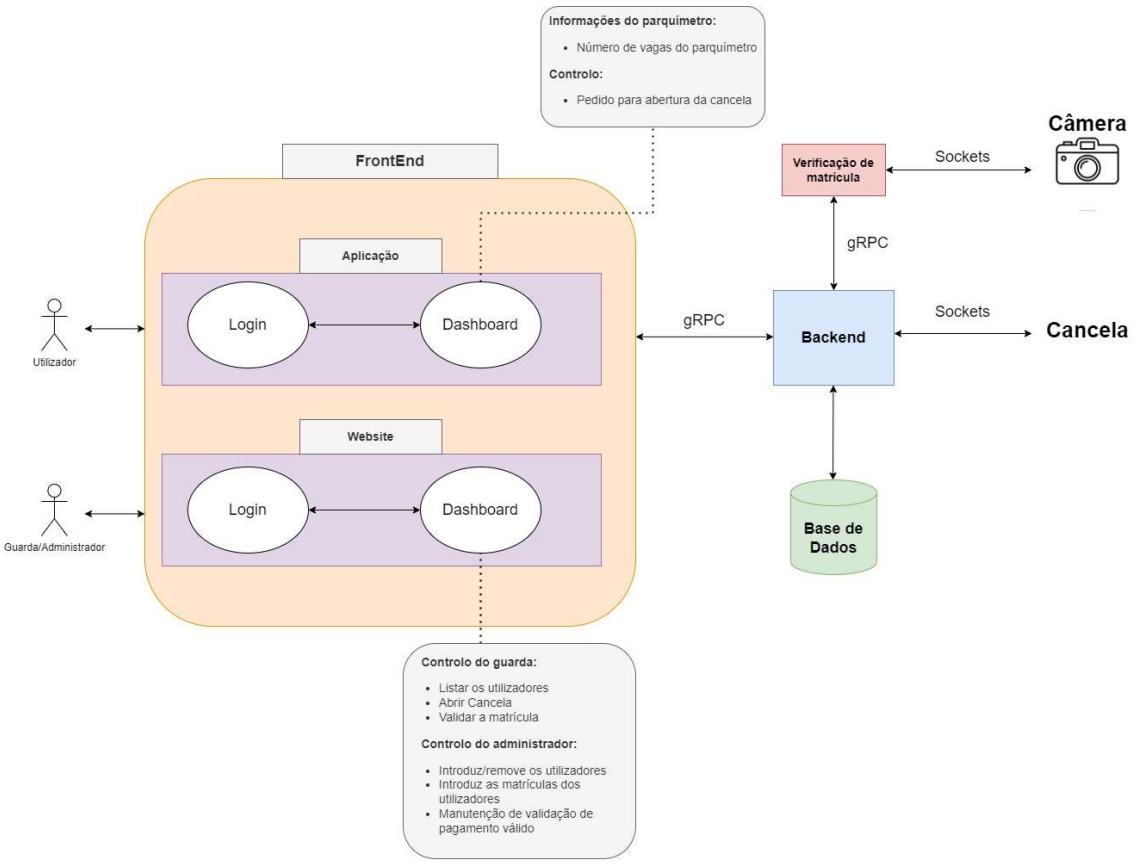

Figura 2 - Arquitetura de solução

4.4 Workflow de Entrada/Saída

Um Utilizador que queira acesso ao parque ou sair deste deve ligar-se à rede local, acessível perto da cabine do segurança que se encontra junto às cancelas.

Feito isto, deverá abrir a aplicação de acesso ao parque no seu telemóvel e proceder ao login, caso nunca o tenha feito antes na aplicação, ou caso tenha feito logout nesta.

Figura 3 – Ecrã de login na aplicação mobile

Este procedimento fará com que a aplicação de mobile envie um pedido de autenticação ao Back end com os dados do utilizador para que este valide se as credencias enviadas são válidas.

Caso os dados de login estejam incorretos o utilizador é notificado que estes estão incorretos, caso contrario são mostradas opções para pedir da abertura das cancelas de entrada e saída, devendo o utilizador pressionar a que pretende.

Figura 4 - Ecrã na app mobile para pedir para entrar ou sair do parque

Após pressionar uma das opções é enviado um pedido ao Back end com a opção selecionada. Este por sua vez envia um pedido à aplicação da plataforma de gestão a "perguntar" se esta está disponível para "processar" um novo pedido de entrada ou saída, dependendo do caso, que "responde" em conformidade com a situação.

Caso o Back end receba a resposta de "processamento" indisponível avisa o utilizador que já existe outro pedido a ser processado naquele momento e pede para aguardar a sua vez conforme é mostrado na figura 5.

Figura 5 - Informação na app mobile que existe outro pedido em processamento

Obtendo uma resposta de disponibilidade por parte da plataforma, o Back end (no caso de ser um pedido de entrada) verifica se o utilizador tem o pagamento em dia para aquele mês (caso não tenha acesso gratuito ao parque) ou se não está a tentar entrar com outra viatura sem ter ainda saído com a primeira.

De seguida, se ambas as verificações anteriores forem verdade, o Back end envia um pedido de matricula à aplicação de Deteção de Matriculas, indicando se é uma entrada ou uma saída.

Este por sua vez, tira uma fotografia à matricula do automóvel da cancela indicada, corre o algoritmo para detetar a matricula e enviar tanto a foto como a matricula detetada para o Back end.

Figura 6 - Fotografia de uma viatura que pretende entrar no parque

Figura 7 – Imagem que a app da Deteção de Matriculas cria para detetar a matricula

Após receber a fotografia, o Back end arquiva-a e procede à verificação da matricula. Caso esta exista em sistema e se pertence ao utilizador em questão.

O resultado, juntamente com a matricula, identificação do utilizador e link para a Foto são enviados para a plataforma de gestão do Front end.

Ao receber esta informação na plataforma o segurança, caso a matricula detetada não corresponda com a da fotografia, insere-a de novo e faz um pedido de reverificação desta ao Back End, que por sua vez devolve o resultado.

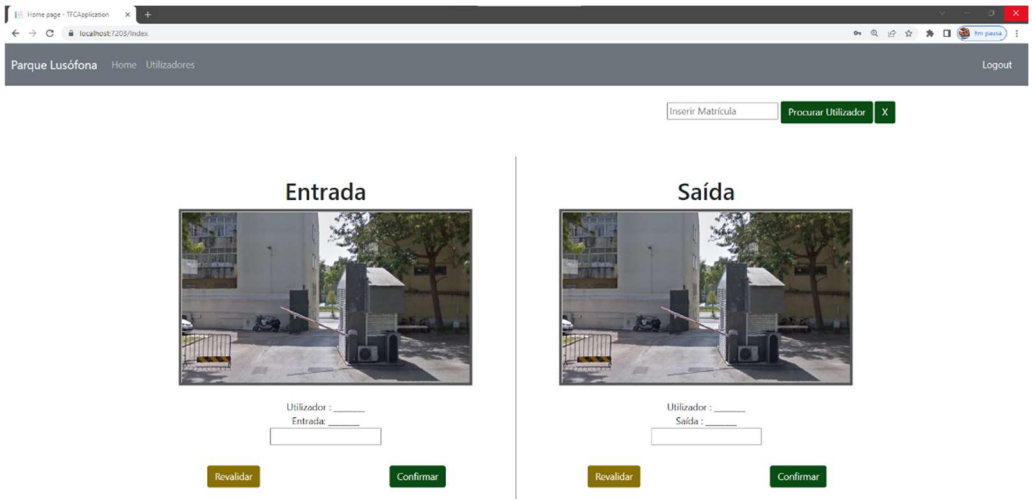

Figura 8 – Pagina da app web do Front End onde o segurança vê a fotografia e a matricula detetada

Por fim o segurança pressiona a opção de confirmação do resultado, e consoante o mesmo procede à abertura da cancela ou não. Esta confirmação de resultado é enviada para o Back end que regista a entrada ou saída conforme e se for caso disso.

Figura 9 – Informa o utente na app mobile que pode entrar

Sempre que exista algum problema com uma verificação como pagamento em falta ou outra, o Back end envia um aviso para a aplicação de Mobile do utilizador para o alertar da situação.

4.5 Segurança

A autenticação dos utilizadores é feita com recurso à encriptação da password, no Front end, antes desta ser enviada para o Back end.

Desta forma, se alguém estiver "à escuta" na rede apenas verá a password encriptada, não sendo possível assim utiliza-la.

4.6 Gestão do parque automóvel

A gestão da informação referente ao parque automóvel é feita pela aplicação desenvolvida pelo grupo do Front end, tal como:

- Gestão dos utilizadores com acesso ao parque como mostrado na figura 10.
- Se o pagamento está em dia ou se é gratuito

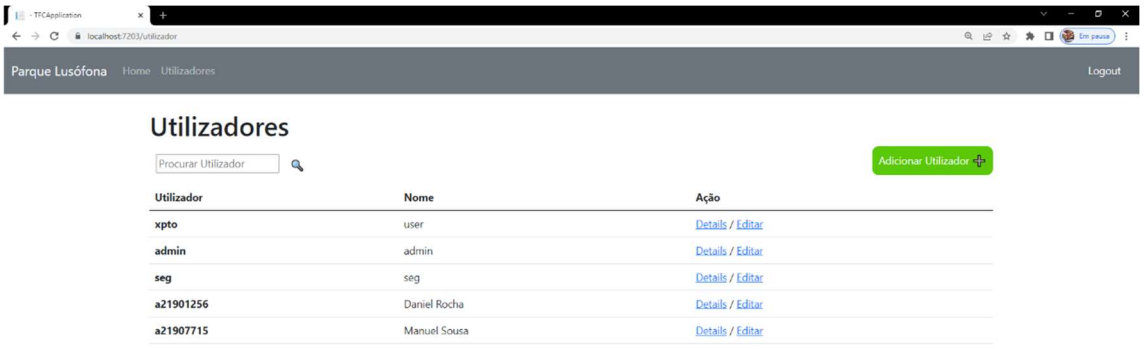

Figura 10 – Lista de utilizadores do parque na app web do Front End

4.7 Funcionalidade Extra

Foi ainda criada uma funcionalidade extra neste projeto que permite ao Front End procurar uma viatura por matricula, caso seja necessário por alguma razão, saber quem é o proprietário desta.

5 Benchmarking

Existem varias soluções de gestão de parques de estacionamento no mercado, como a da Prevalta ou TicketCode. No entanto as vantagens entre esta solução e as demais existentes no mercado são:

- O baixo custo de implementação, considerando não existem custos com o software utilizado e desenvolvido.
- É uma solução feita à medida e que irá colmatar as necessidades da ULHT na gestão do parque de estacionamento.
- Podem ser acrescentadas mais funcionalidades às aplicações sem que exista um acréscimo de custos.
- Outras soluções como a utilização de cartões com NFC's trás outros custos como o dos próprios cartões e respetivos leitores, que dependendo do tipo de tecnologia podem ser bastante caros. Isto sem contar com o facto de os utilizadores poderem perder o cartão, obrigado à emissão de um novo mais a respetiva configuração.

Estas são apenas algumas das razões pelas quais acreditamos que a melhor opção de autenticação passa por uma aplicação para dispositivos moveis em conjunto com as outras aplicações desenvolvidas entre estes três projetos de TFC.

6 Método e Planeamento

O planeamento deste projeto foi pensado desde o primeiro momento após a sua aceitação por parte da coordenação dos TFC's. Houve uma tentativa de previsão do tempo necessário para o desenvolvimento de cada tarefa, quais as melhores tecnologias a utilizar, qual a melhor abordagem para o seu desenvolvimento e como seria feita a comunicação dentre os diferentes TFC's.

Para a criação do planeamento deste projeto e respetiva calendarização das várias fases deste foi utilizado um diagrama de Gantt, figura 11 mostrada abaixo, devido à sua fácil interpretação.

Figura 11 – Calendário de desenvolvimento do TFC

Inicialmente existiu algum otimismo, relativamente à comunicação com os outros dois grupos, no entanto houve pouco dialogo com o grupo da Deteção de Matriculas, o que dificultou um pouco a implementação da comunicação entre estes dois projetos (Deteção de Matriculas e Back End).

Como foi planeado desde o inicio criar aplicações de teste para simular as comunicações entre as diversas aplicações dos vários projetos envolvidos nesta solução, foi possível mitigar a situação da pouca comunicação anteriormente descrita, uma vez que estas aplicações permitiram avançar com o projeto de uma forma mais célere que o que aconteceria caso tivessem sido criadas.

A figura 12 mostra uma das aplicações de teste criadas para testar diversas funcionalidades.

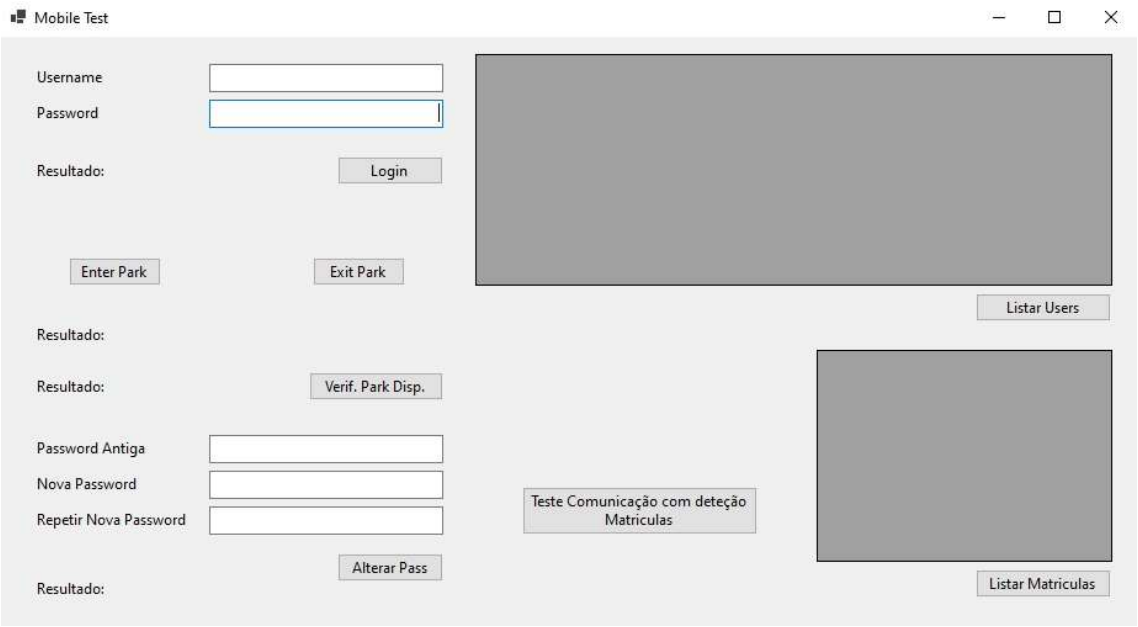

Figura 12 - Uma das app's criadas para testar funcionalidades

Felizmente, o dialogo com esse grupo foi estabelecido mais para o fim do desenvolvimento, conseguindo-se assim testar as comunicações não só entre o Back End e o Front End, como também entre Back End e Deteção de Matriculas, com sucesso em qualquer um dos casos.

7 Resultados

Na entrega final, este projeto encontra-se com todos os requisitos propostos inicialmente implementados, com exceção da implementação da encriptação das passwords dos utilizadores no Back End, uma vez que não faria qualquer sentido que assim fosse.

Isto porque caso existisse um ataque do género “Man in the Middle”, o atacante ficava a saber quais as password’s dos utilizadores antes destas estarem encriptadas, não adiantando em nada a encriptação das mesmas à posteriori, no Back End, tendo sido esta a razão pelo qual este requisito não foi implementado e foi retirado do Back End.

Para além dos principais requisitos implementados, a comunicação com as outras aplicações desta solução e o acesso aos dados na Base de Dados, quero também mostrar como foram divididos os serviços prestados pelo Back End.

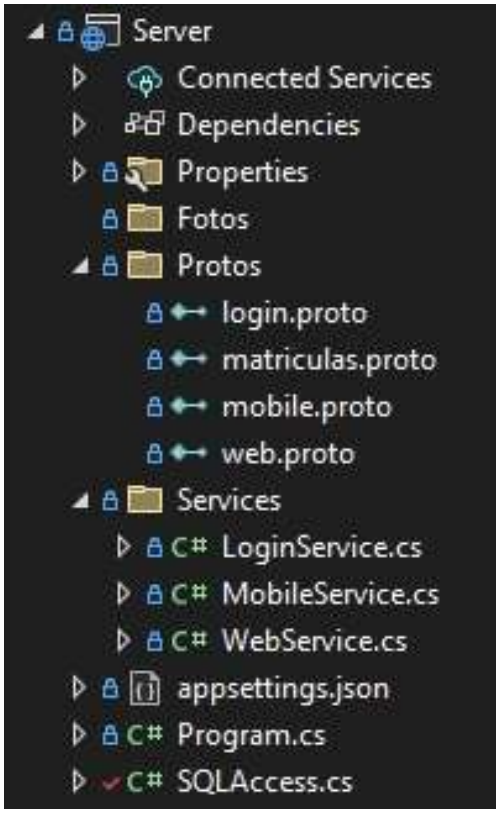

Figura 13 – Estrutura de Ficheiros que compõe este projeto

Na figura 13 podemos observar quatro ficheiros proto que servem como estrutura de dados para a comunicação entre os vários projetos desta solução, a class SQLAccess que estabelece a comunicação entre o Back End e a Base de Dados, e ainda as classes com os diversos serviços que o Back End disponibiliza como podemos ver nas imagens 14, 15 e 16.

```
public class LoginService : Login.LoginBase
{
    3 references
    public override Task<LoginAutorization> LoginRequest(LoginData pedido, ServerCallContext context)...

    3 references
    public override Task<PasswordChanged> ChangePassword(NewLoginData pedido, ServerCallContext context)...
}
```

Figura 14 – Serviços Login

Os serviços de Login, figura 14, são disponibilizados para qualquer uma das aplicações de Front End de forma a que os seus utilizadores se possam autenticar e alterar a sua password, caso sintam essa necessidade.

```
public class MobileService : Mobile.MobileBase
{
    3 references
    public override async Task<Resposta> AbrirCancela(Pedido pedido, ServerCallContext context)...
}
```

Figura 15 – Serviços Mobile

O serviço de Mobile, figura 15, é disponibilizado unicamente para as aplicações mobile do Front End, e é este o serviço utilizado para pedir autorização para entrar ou sair do parque de estacionamento. É este serviço o responsável, não só, mas também, por pedir à aplicação de Deteção de Matriculas que detete e forneça a matricula detetada e a fotografia da desta.

```
public class WebService : Web.WebBase
{
    3 references
    public override async Task GetAllUsers(Nothing pedido, IServerStreamWriter<DadosUser> responseStream, ServerCallContext context)...

    3 references
    public override async Task GetAllMatriculas(User pedido, IServerStreamWriter<ListaMatriculas> responseStream, ServerCallContext context)...

    3 references
    public override Task<Nothing> InsertUser(DadosUserInsert pedido, ServerCallContext context)...

    3 references
    public override Task<Nothing> EditUser(DadosUser pedido, ServerCallContext context)...

    3 references
    public override Task<DadosUser> SearchUser(User pedido, ServerCallContext context)...

    3 references
    public override Task<Nothing> InsertMatricula(DadosAuto pedido, ServerCallContext context)...

    3 references
    public override Task<Nothing> EditMatricula(DadosAuto pedido, ServerCallContext context)...

    3 references
    public override Task<Nothing> InsertEntryPermition(DadosPermEntrada pedido, ServerCallContext context)...

    3 references
    public override Task<Nothing> EditEntryPermition(DadosPermEntrada pedido, ServerCallContext context)...

    3 references
    public override Task<DadosPermEntrada> GetEntryPermitionData(User pedido, ServerCallContext context)...

    3 references
    public override Task<Nothing> RegisterEntry(DadosAuto pedido, ServerCallContext context)...

    3 references
    public override Task<Nothing> RegisterExit(DadosAuto pedido, ServerCallContext context)...

    3 references
    public override Task<MatriculaValida> PlateRevalidation(DadosAuto pedido, ServerCallContext context)...
}
```

Figura 16 – Serviços Web

Os serviços Web, figura 16, são disponibilizados para a aplicação Web do Front End e servem para permitir a gestão do parque nomeadamente dos utilizadores

autorizados, associação de matriculas das viaturas aos utilizadores, pagamentos entre outros.

```
syntax = "proto3";

option csharp_namespace = "Server";

package login;

service Login {
  rpc LoginRequest (LoginData) returns (LoginAutorization);
  rpc ChangePassword (NewLoginData) returns (PasswordChanged);
}

// The request message
message LoginData {
  string login = 1;
  string password = 2;
}

// The response message
message LoginAutorization {
  bool loginEfetuado = 1;
  string tipo_user = 2;
}

// The request message
message NewLoginData {
    string login = 1;
    string oldPass = 2;
    string newPass = 3;
}

// The response message
message PasswordChanged {
    bool passAlterada = 1;
}
```

Figura 17 – Ficheiro proto Login

Na figura 17 temos o ficheiro proto criado para a comunicação dos dados de autenticação ou alteração da password dos utilizadores.

```
public override Task<LoginAutorization> LoginRequest(LoginData pedido, ServerCallContext context)
{
    Boolean loginAutorizado = false;

    Console.WriteLine("Recebido pedido de autenticação do utilizador " + pedido.Login);

    // Validar os dados de login na BD
    if (Boolean.Parse(SQLAccess.ExecSQLReturnData("exec spUtilizadores_LoginValido '" + pedido.Login + "', '" + pedido.Password + "'")))
    {
        if (Boolean.Parse(SQLAccess.ExecSQLReturnData("spUtilizadores_ContaAtivada '" + pedido.Login + "'")))
        {
            loginAutorizado = true;
            Console.WriteLine(value: "Autenticação do utilizador bem sucedida!");
        }
        else
        {
            Console.WriteLine(value: "A sua conta encontra-se desativada!");
        }
    }
    else
    {
        Console.WriteLine(value: "Autenticação invalida!");
    }

    // Obter tipo de utilizador
    string tipoUser = SQLAccess.ExecSQLReturnData("spUtilizadores_GetTipoUtilizador '" + pedido.Login + "'");

    return Task.FromResult(new LoginAutorization { LoginEfetuado = loginAutorizado, TipoUser = tipoUser });
}
```

Figura 18 – Pedido de autenticação

O código da figura 18 demostra como são processados os pedidos de autenticação feitos pelos utilizadores. Na linha 31 do código é feito um pedido à base de dados para que proceda à autenticação do utilizador e devolva resultado dessa autenticação.

Caso a autenticação seja bem-sucedida, na linha 33, é feito um novo pedido à Base de Dados para verificar se aquela conta se encontra ativa.

Na linha 49, é feito um ultimo pedido à Base de Dados para tentar determinar que tipo de utilizador se está a tentar "logar". A BD devolve o tipo de utilizador ou uma string vazia caso este não exista.

Por fim é devolvido o resultado à aplicação que fez o pedido.

```
21 references
public static class SQLAccess
{
    private static readonly string connStr = "Data Source=Localhost\\PARKINGLOTDB;Initial Catalog=ParkingLotDB;User ID=sa;pwd=Lusofona";

    5 references
    public static DataTable ExecSqlDt(string storedProcName)
    {
        SqlConnection conn = new (connStr);
        conn.Open();
        SqlCommand sqlCommand = new (storedProcName, conn);
        DataTable dt = new ();
        dt.Load(sqlCommand.ExecuteReader());
        conn.Close();
        return dt;
    }

    17 references
    public static string ExecSQLReturnData(string storedProcName)
    {
        var value = string.Empty;
        try
        {
            DataRow row = ExecSqlDt(storedProcName).Rows[index: 0];
            value = row[columnIndex: 0].ToString();
        }
        catch
        {

        }
        #pragma warning disable CS8603 // Possible null reference return.
        return value;
        #pragma warning restore CS8603 // Possible null reference return.
    }
}
```

Figura 19 – Class SQLAccess

Conforme mencionado anteriormente, a Class SQLAccess é a responsável pelas comunicações entre o Back End e a Base de Dados, através das duas funções que podemos ver na figura 19.

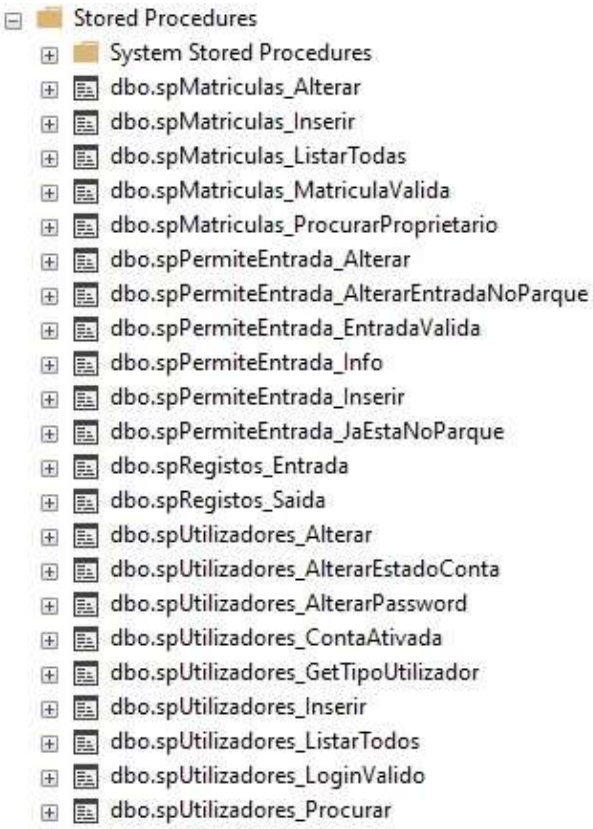

Figura 20 – Stored Procedures

A figura 20 mostra todos os Stored Procedures criados para este projeto.

```
ALTER procedure [dbo].[spUtilizadores_LoginValido]
    @Login Varchar(10),
    @Password varchar (20)
as
begin
    declare @LoginValido int;

    select @LoginValido = ID from Utilizadores where Login = @Login and Password = @Password;

    if @LoginValido > 0
    begin
        select 'true' as Resultado
    end
    else
    begin
        select 'false' as Resultado
    end
end
```

Figura 21 – Stored Procedure spUtilizadores_LoginValido

A figura 21 ilustra um dos Stored Procedures criados para este projeto. É aqui nos Stored Procedures que são verdadeiramente verificados todos os pedidos de dados referentes a informações, autenticações, listagens, entre outros.

8 Conclusão e Trabalhos Futuros

Apesar deste projeto estar funcional, existem sempre melhorias que podem ser efetuadas para o melhorar, nomeadamente ao nível da configuração, da segurança, implementação de novas funcionalidades como estatísticas entre outras.

Ao nível da Segurança:

- Criar um utilizador na Base de Dados que tenha apenas acesso aos Stored Procedures, invés de se utilizar o utilizador de administração desta.

Ao nível da configuração, passaria pela criação de um ficheiro de configuração:

- Que seria "lido" pelo Back End sempre que fosse inicializado;
- Que conteria qual o utilizador da BD a utilizar e qual a password deste. (Apesar de não ser obrigatório, seria conveniente que este utilizador tivesse apenas acesso aos Stored Procedures e não um "Power User" ou "Admin" da BD, por motivos de segurança);
- Conteria também os endereços IP e respetivas portas de todos os serviços que iria necessitar como por exemplo da Deteção de Matriculas e da Base de Dados.
- Poderia também conter a localização da partilha de rede onde seriam guardadas as fotos das matriculas caso no futuro se desejasse guardar estes ficheiros noutro servidor que não aquele onde o Back End está a "correr".

Esta unidade curricular permitiu-me desenvolver um projeto com uma abordagem diferente de todos os outros projetos em que já trabalhei, estimulando-me a pensar em novas formas de ultrapassar os desafios encontrados e a integrar soluções que até aqui nunca tinha sentido necessidade de utilizar.

Sabendo o que sei hoje quando iniciei este projeto, existem algumas coisas que teria feito de outra forma, nomeadamente a forma como criei os Stored Procedures, que poderia ter sido de uma forma mais "elegante". No entanto existem outras em que fico feliz por ter tomado as decisões que tomei.

Foi uma experiencia bastante profícua não só a nível académico como também a nível pessoal e profissional, uma vez que me permitirá abordar futuros projetos de uma nova forma.

Bibliografia

[DEISI21] DEISI, Regulamento de Trabalho Final de Curso, Set. 2021.

[TaWe20] Tanenbaum,A. e Wetherall,D., *Computer Networks*, 6ª Edição, Prentice Hall, 2020.

[ULHT21] Universidade Lusófona de Humanidades e Tecnologia, www.ulusofona.pt, acedido em Out. 2021.

[gRPC21] gRPC, Google Remote Procedure Call Framework, gRPC, acedido em Out. 2021.

[SQL21] SQL, Structured Query Language, SQL Introduction (w3schools.com), acedido em Out. 2021.

[SP21] Stored Procedure, SQL Stored Procedures (w3schools.com), acedido em Out. 2021.

[Prevalta22] Prevalta, Prevalta Alarmes Camaras Incêndio CTRL Acessos, Portugal, acedido em Jul. 2022.

[TicketCode] TicketCode, Ticketcode | Soluções para parques de estacionamento, acedido em Jul. 2022.

Anexo - Recomendações

Recomenda-se que os ou o computador, onde esta solução for implementada, esteja protegido por uma Firewall da rede local os utilizadores dos dispositivos moveis se vão ligar.

Uma forma de atingir isto com baixo custo pode passar por colocar no computador, onde a aplicação de Back end for instalada, duas placas de rede, para que exista uma separação entre a rede local onde os utilizadores das aplicações dos dispositivos moveis se ligam e a rede onde estão os demais computadores, caso existam, onde poderia estar a base de dados instalada ou onde a aplicação de Front end (para gestão do parque) e/ou a aplicação de Deteção de Matriculas poderiam estar a correr, por exemplo instaladas.

Nesta mesma solução de segurança, conforme mencionado anteriormente, deveria também ser colocada/ativada uma firewall no computador com o Back end e dita rede local, bloqueando todo o trafego entre este pc e os dispositivos moveis com exceção da porta onde estabelecida para a comunicação entres os ditos dispositivos e a aplicação.

Outra solução para proteger o Back end e demais infraestruturas, um pouco mais cara, poderia ser a colocação de um segundo router com Firewall.

Neste router estaria ligado ao pc com a aplicação do Back end e os outros pc's com a base de dados e ou as demais aplicações desta solução, caso não sejam instaladas no mesmo pc.

Este router por sua vez estaria ligado ao primeiro e teria configurada a Firewall para permitir o acesso da rede local, apenas à porta necessária para que as aplicações dos dispositivos moveis comuniquem com o Back end. O primeiro router é cria a rede local onde os dispositivos moveis se conectam.

Glossário

LEI	Licenciatura em Engenharia Informática
TFC	Trabalho Final de Curso
gRPC	Google Remote Procedure Call Framework
SQL	Structured Query Language