



Relatório Final da Prática Simulada

WorkStation

Andrei Latis

170019, 01, TGPSI

Lisboa 23 de julho de 2020

Conteúdo

Introdução:	1
Prática Simulada:	1
Fundamentação Técnico-científica:	2
Tabelas da base de dados:	3
Tabela 1 – UserTable	3
Tabela 2 – UserTechnicalLanguage	4
Tabela 3 – TechnicalLanguage.....	4
Tabela 4 – UserTechnology	5
Tabela 5 – Technology.....	5
Tabela 6 – UserLanguage	5
Tabela 7 – LanguageTable.....	6
Tabela 8 – Level.....	6
Tabela 9 – Project.....	6
Tabela 10 – AdTable	7
Tabela 11 – TypeProject.....	8
Descrição do produto:	9
Login:	10
Register:	13
Profile do Desenvolvedor/Admin:	19
Métodos usados método Profile().....	20
Conclusão:	24
Webgrafia:.....	24

Introdução:

No âmbito da Prova de Aptidão Profissional (PAP) do Curso Técnico de Gestão e Programação de Sistemas Informáticos (TGPSI) os alunos do mesmo curso são submetidos à realização de um projeto final. O projeto consiste em que os alunos demonstrem os conhecimentos adquiridos ao longo do curso e do estágio ou prática simulada.

A ideia do projeto surgiu depois que o professor Jorge Carvalho sugeriu aos alunos, para fazerem um projeto que facilitasse a vida das pessoas. Por isso o projeto realizado foi um *website* em que pessoas com conhecimento na área de Tecnologia da Informação (TI) mais especificamente programação, possam vender *websites*, programas para computadores, programas para telemóveis ou bases de dados, criados pelos mesmos. Esse projeto facilitara muito a vida de pessoas com conhecimento na área de Tecnologia da Informação, principalmente os juniores que podem ter dificuldade em encontrar emprego sem nenhuma experiência de trabalho.

Prática Simulada:

Sem um orientador para indicar as tarefas a serem executadas, a realização do projeto foi uma tarefa complicada, devido a várias dúvidas e distrações. A expectativa era acabar o projeto muito antes do que o tempo limite dado pelo professor, mas não foi possível devido a várias dificuldades ao longo do percurso como, indecisões a cerca do projeto.

Para a realização do projeto passei por várias etapas:

1. Planeamento do projeto - Onde determinei como o *website* funcionaria.
2. *Design* e Criação da Base de Dados - Determinação da estrutura e dos valores que a base de dados guardara e a implementação da base de dados no SQL Serve com base na estrutura e valores determinados.
3. Criação da DAL (*Data Access Layer*) – Criação da camada de acesso a dados em C# na plataforma ASP.NET.

4. Criação do *Website* – Usando o modelo MVC, a sintaxe Razor.
5. *Design* do *Website* - Usando *Bootstrap* para melhorar o interface do utilizador.
6. Correção de erros e bugs do *website*.

Fundamentação Técnico-científica:

As tecnologias utilizadas no desenvolvimento do projeto foram: *HTML/CSS*, *BootStrap*, *C#*, *Razor*, *ASP.NET MVC* e *SQL*.

- *HTML/CSS* - *Hypertext Markup Language/ Cascading Style Sheets* ou em português, linguagem de marcação de hipertexto/ Folhas de Estilo em Cascata são linguagens necessárias na criação de *websites*. Enquanto o *HTML* serve para a criação da pagina o *CSS* serve para melhorar o estilo da pagina *HTML*.
- *BootStrap* – Conjunto de *CSS* e códigos em *javascript* para facilitar a vida dos desenvolvedores deixando as paginas *web* com mais estilo e funcionalidades
- *C#* - E uma linguagem de programação que foi utilizada nesse projeto para a criação da *DAL* e de algumas partes do *website* como por exemplo do *MVC*.
- *Razor* – *Razor* é uma sintaxe que usa a linguagem de programação *c#* para a criação de páginas *web* dinâmicas.
- *ASP.NET MVC* - *ASP.NET* é uma tecnologia da Microsoft para criação de *websites* dinâmicos, e *MVC (Model–View–Controller)* é um modelo de 3 camadas para organizar melhor a aplicação e desenvolver uma melhor interfaces para o utilizador.
- *SQL* - *Structured Query Language*, é uma linguagem de programação para manipular dados em *SGBD*, a linguagem de programação foi utilizado para criar é manipular a base de dados em *SQL Server 2012* que dá suporte ao *website*.

Os Programas utilizadas no desenvolvimento do projeto foram: *Microsoft Visual Studio 2015* para a criação da *DAL* e do *website* e *SQL Server 2012* para a criação da base de dados.

Tabelas da base de dados:

Tabela 1 – UserTable

A tabela UserTable guarda todos os utilizadores que se registraram no *website*, UserTable tem um campo chamado user_level que serve para definir o nível de um utilizador já que existem vários tipos de utilizadores como por exemplo o comprador/cliente, desenvolvedor e o *Admin*. A tabela UserTable esta relacionada as tabelas UserTechnicalLinguagem, UserTechnology e UserLanguage pelo campo user_id que é a chave primária da Tabela UserTable.

A Tabela UserTable esta ainda relacionada com a tabela Project pelo campo UserTable(user_id, Primary Key) com o campo Project(id_cliente) e também esta relacionada com a AdTable pelo campo UserTable(user_id, Primary Key) com o campo AdTable(ad_creatorID).

Nome do campo	Tipo de dados	Chaves	Descrição
user_id	Int	Primary Key	Id do utilizador, Valor Único
username	Varchar(50)		Nome do Utilizador, Valor Único
user_first_name	Varchar(50)		Nome Próprio do utilizador
user_last_name	Varchar(50)		Apelido do utilizador
user_email	Varchar(50)		Correio eletrónico do utilizador, Valor Único
user_password	Char(128)		Palavra-passe do utilizador

profile_img	Nvarchar(MAX)		Imagem do utilizador, pode ser NULL
user_birthday	Date		Data de nascimento do utilizador
user_level	Tinyint		Nível do utilizador
user_bio	Varchar(200)		Biografia do utilizador, pode ser NULL
blocked	Bit		Pode ser NULL

Tabela 2 – UserTechnicalLanguage

A tabela UserTechnicalLanguage serve como intermediário entre a tabela UserTable e a tabela TechnicalLanguage, já que um utilizador pode estar relacionado a várias Linguagens/Linguagens de Programação e vice-versa.

Nome do campo	Tipo de dados	Chaves	Descrição
Id_user	Int		Id do utilizador
id_TechnicalLanguage	Int		Id da Linguagem Técnica

Tabela 3 – TechnicalLanguage

A tabela TechnicalLanguage guarda as linguagens/linguagens de programação como por exemplo *html*, *c#*, *javascript* etc.

Nome do campo	Tipo de dados	Chaves	Descrição
id_technical_language	Int	Primary Key	Id da linguagem técnica, Valor Único
txt_technical_language	Varchar(50)		Nome da linguagem

			tecnica
--	--	--	---------

Tabela 4 – UserTechnology

A tabela UserTechnology serve como intermediário entre a tabela UserTable e a tabela Technology, já que um utilizador pode estar relacionado a várias Tecnologias e vice-versa.

Nome do campo	Tipo de dados	Chaves	Descrição
Id_user	Int		Id do utilizador
id_technology	Int		Id da Tecnologia

Tabela 5 – Technology

A tabela Technology guarda as tecnologias como por exemplo ASP.NET, MVC, Ajax etc.

Nome do campo	Tipo de dados	Chaves	Descrição
id_technology	Int	Primary Key	Id da Tecnologia, Valor Único
txt_technology	Varchar(50)		Nome da Tecnologia

Tabela 6 – UserLanguage

A tabela UserLanguage serve como intermediário entre a tabela UserTable e a tabela *Language*, já que um utilizador pode estar relacionado a várias Línguas e vice-versa.

Nome do campo	Tipo de dados	Chaves	Descrição
Id_user	Int		Id do utilizador
id_language	Int		Id da Linguagem

Tabela 7 – LanguageTable

A tabela LanguageTable guarda as línguas como por exemplo Inglês, Português, Japonês etc.

Nome do campo	Tipo de dados	Chaves	Descrição
id_language	Int	Primary Key	Id da Linguagem, Valor Único
txt_language	Varchar(50)		Nome da Linguagem

Tabela 8 – Level

A tabela Level guarda os tipos de níveis como por exemplo o comprador, o desenvolvedor e o Admin.

Nome do campo	Tipo de dados	Chaves	Descrição
level_id	tinyint	Primary Key	Id do nível, Valor Único
level_txt	Varchar(50)		Nome do nível

Tabela 9 – Project

A tabela Project guarda todos os projetos mesmo aqueles que foram negados.

Nome do campo	Tipo de dados	Chaves	Descrição
Id_project	Int	Primary Key	Id do projeto, Valor Único
id_ad	Int		Id da publicidade
id_cliente	Int		Id do cliente
requirements_and_data	Varchar(8000)		Requisitos do Dados do

			projeto
start_date	datetime		Data e hora em que o projeto foi inicializado, pode ser NULL
end_date	datetime		Data e hora em que o projeto foi pago e entregue, pode ser NULL
stars	int		Classificação do projeto em estrelas de 1 a 5 (Classificado pelo cliente), pode ser NULL
txt_review	Varchar (500)		Críticas do cliente, pode ser NULL
status	int		Estado do projeto (Reprovado = 0, A espera = 1, Aceito e em processo = 2, Terminado = 3, Pago e entregue = 4)

Tabela 10 – AdTable

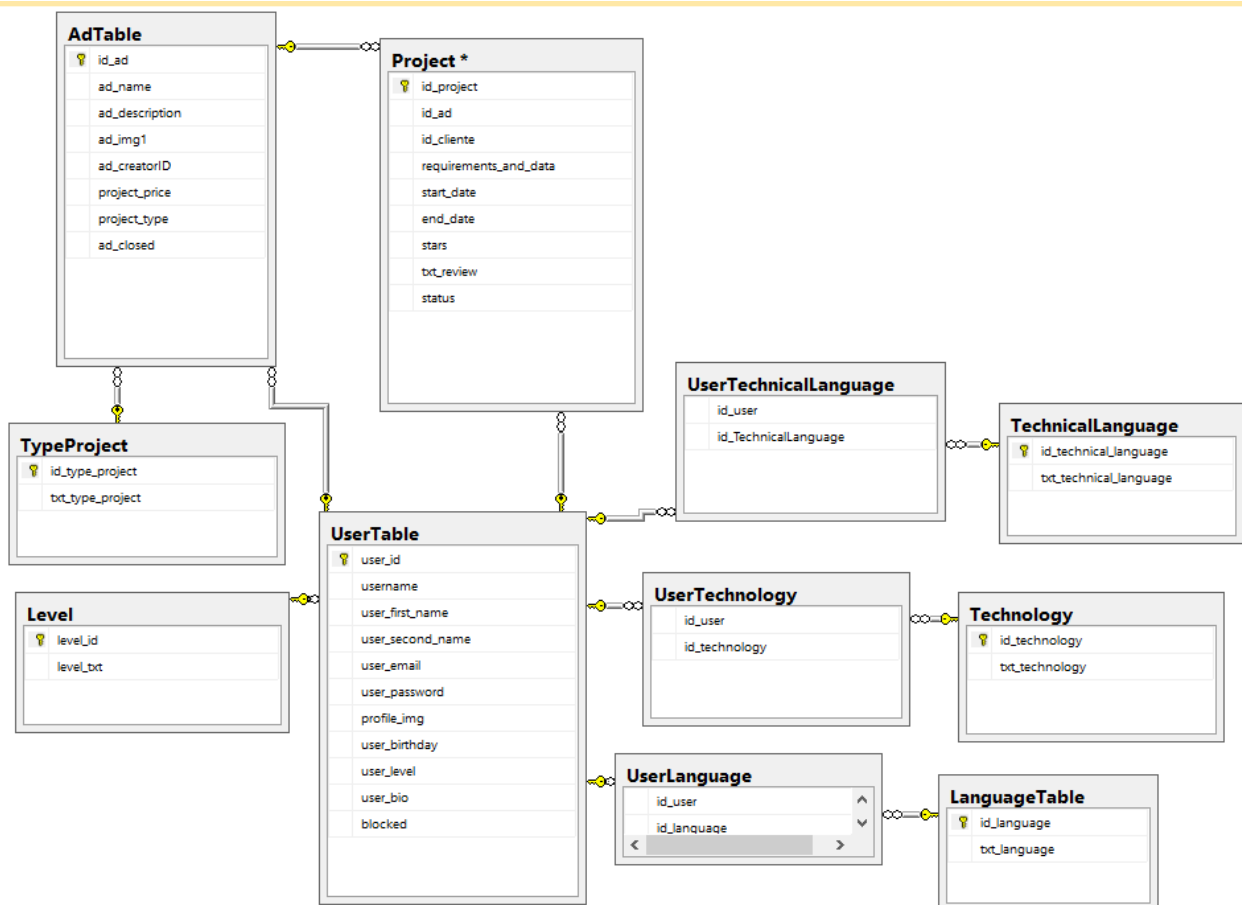
A tabela AdTable guarda todos os dados das publicidades já feitas. A tabela AdTable esta ligada a tabela Project pelo id_ad.

Nome do campo	Tipo de dados	Chaves	Descrição
id_ad	Int	Primary Key	Id da publicidade, Valor Único
ad_name	varchar(50)		Nome da publicidade
ad_description	varchar(200)		Mini descrição do projeto
ad_img1	nvarchar(MAX)		Imagem da publicidade
ad_creatorID	int		Id do criador da publicidade
project_price	money		Preço do projeto
project_type	int		Tipo de projeto
ad_closed	bit		Publicidade aberta = 0, Publicidade fechada = 1

Tabela 11 – TypeProject

A tabela TypeProject guarda todos os tipos de projetos que podem ser vendidos pelo *Website* como por exemplo *Android/IOS App, Database, Scripts* etc.

Nome do campo	Tipo de dados	Chaves	Descrição
id_type_project	Int	Primary Key	Id do tipo de projeto
txt_type_project	Varchar(50)		Tipo de projeto



Descrição do produto:

WorkStation – WorkStation é um *website* onde desenvolvedores com conhecimentos na área de TI, podem trabalhar fazendo websites, programas para PC ou Base de dados para outras pessoas.

No *website* para vender ou compra temos que ter uma conta criada, por isso no *website* temos login e o *register*.

Login:

Login

Email	<input type="text"/>
Password	<input type="password"/>
<input type="button" value="Create"/>	

Depois de inserir os dados e clicar no botão *Create* os dados são enviados para o *controller*, que chama o método `LoginAcc()` que por sua vez tem acesso a base de dados para verificar se o email e a palavra-passe combinam.

```
[HttpPost]
public ActionResult LogIn(string Email, string Password)
{
    if (ModelState.IsValid)
    {
        WorkStationDAL.User User = new WorkStationDAL.User();
        List<WorkStationDAL.User> str = new
List<WorkStationDAL.User>();
        str = User.LoginAcc(Email, PasswordString(Password));
        if (str != null)
        {
            Session["Username"] = str[0].userName;
            Session["UserLevel"] = str[0].userLevelId;
            return View("~/Views/Home/Index.cshtml");
        }
        else
        {
            return View("~/Views/Alert/LoginFailed.cshtml");
        }
    }
    return View();
}
```

O método `LoginAcc()` recebe o email, e a palavra-passe encriptada, se o email e a palavra-passe combinarem o método retornara uma lista com o Nome e o nível do utilizador, caso contrario retornara *null*.

```

public List<User> LoginAcc(string Email, string Password)
{
    userEmail = Email;
    userPassword = Password;

    List<User> UserN_LevelI = new List<User>();
    int idx = 0;
    connec.Open();

    SqlCommand cmd = new SqlCommand("spAccount_Login", connec);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.AddWithValue("@UserEmail", userEmail);
    cmd.Parameters.AddWithValue("@UserPassword", userPassword);

    SqlDataReader Rd = cmd.ExecuteReader();

    try
    {
        while (Rd.Read())
        {
            if (Convert.ToInt32(Rd.GetValue(0)) == -1)
            {
                connec.Close();
                return null;
            }

            UserN_LevelI.Add(new User());
            UserN_LevelI[idx].userLevelId =
Convert.ToInt32(Rd.GetValue(0));
            UserN_LevelI[idx].userName = Rd.GetValue(1).ToString();
        }
        connec.Close();
        return UserN_LevelI;
    }
    catch (Exception)
    {
        connec.Close();
        return null;
    }
}

```

Stored Procedure usada no método LoginAcc(), A seguinte Stored Procedure verifica se o Email e a palavra-passe combinam.

```

create procedure dbo.spAccount_Login
    @UserEmail varchar(50),
    @UserPassword varchar(128)

as
begin
    set nocount on

    declare @UserPassword2 varchar(128) = '0'

    select @UserPassword2 = user_password from UserTable where user_email =
@UserEmail and blocked = 0


    if @UserPassword2 != @UserPassword
begin

```

```
        select * from TableMinusOne
    end
    else
    begin
        select user_level, username from UserTable where user_email =
@UserEmail
    end
end
```

Register:

Register

First Name	<input type="text"/>
Second Name	<input type="text"/>
Username	<input type="text"/>
Birthday	<input type="text" value="dd/mm/aaaa"/> 
Email	<input type="text"/>
Confirm Email	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
Profile Image	<input type="button" value="Escolher Ficheiros"/> Nenhum ficheiro selecionado
<input type="button" value="Create"/>	

Depois de inserir os dados e clicar no botão *Create* os dados são enviados para o *controller*, que chama o método `RegisterAcc()`.

```
[HttpPost]
public ActionResult Register(Register model) //registra quando clico no
    butao create
    {
        System.IO.Stream stream = model.imgURL.InputStream;
        System.Drawing.Image image =
        System.Drawing.Image.FromStream(stream);

        if (image.Height > 128 || image.Width > 128)
        {
            return RedirectToAction("LoginFailed", "Alert");
        }

        string FullPath = AppDomain.CurrentDomain.BaseDirectory +
        System.Configuration.ConfigurationManager.AppSettings["ImgPath"] + @"\" +
        model.Username + Path.GetExtension(model.imgURL.FileName);
```

```

        string PathInProject =
System.Configuration.ConfigurationManager.AppSettings["ImgPath"] + @"\" +
model.Username + Path.GetExtension(model.imgURL.FileName);

        if (ModelState.IsValid) //Para ver se esta tudo preenchid sei la
crl
        {
            WorkStationDAL.User User = new WorkStationDAL.User();
            if (User.RegisterAcc(model.Username, model.FirstName,
model.SecondName, model.Email, PasswordString(model.Password), PathInProject,
model.Birthday) != -1)
            {
                try
                {
                    model.imgURL.SaveAs(FullPath);
                    Session["Username"] = model.Username;
                }
                catch
                {
                    User.DeletRegister(model.Username);
                    return View("~/Views/Alert/RegisteredFailed.cshtml");
                }
            }
            else
            {
                return View("~/Views/Alert/RegisteredFailed.cshtml");
            }
            return View("~/Views/Alert/SuccessfullyRegistered.cshtml");
        }

        return View();
    }
}

```

O método RegisterAcc() tem acesso a base de dados para verificar se o email ou o nome de utilizador já existem, se existirem a conta não será criada .

```

public int RegisterAcc(string Username, string FirstName, string SecondName,
string Email, string Password, string ProfileImg, DateTime Birthday)
{
    userName = Username;
    userFirstName = FirstName;
    userSecondName = SecondName;
    userEmail = Email;
    userPassword = Password;
    userProfileImg = ProfileImg;
    userBirthday = Birthday;

    try
    {
        connec.Open();
        SqlCommand cmd = new SqlCommand("spAccount_Create", connec);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@Username", userName);
        cmd.Parameters.AddWithValue("@FirstName", userFirstName);
        cmd.Parameters.AddWithValue("@SecondName", userSecondName);
        cmd.Parameters.AddWithValue("@UserEmail", userEmail);
        cmd.Parameters.AddWithValue("@UserPassword", userPassword);
        cmd.Parameters.AddWithValue("@UserProfileImg", userProfileImg);
        cmd.Parameters.AddWithValue("@UserBirthday", userBirthday);
    }
}

```



```

        //cmd.ExecuteNonQuery();
        SqlDataReader Rd = cmd.ExecuteReader();

        while (Rd.Read())
        {
            if (Convert.ToInt32(Rd.GetValue(0)) == -1)
            {
                connec.Close();
                return -1;
            }
        }
        connec.Close();
        return 0;
    }
    catch (Exception)
    {
        connec.Close();
        return -1;
    }
}

```

Stored Procedure usada no método RegisterAcc() .

```

alter procedure dbo.spAccount_Create
    @Username varchar(50),
    @FirstName varchar(50),
    @SecondName varchar(50),
    @UserEmail varchar(50),
    @UserPassword varchar(128),
    @UserProfileImg nvarchar(MAX),
    @UserBirthday date
as
begin
    set nocount on

    declare @user_id1 int = -1
    declare @user_id2 int = -1

    select @user_id1 = user_id from UserTable where username = @Username
    select @user_id2 = user_id from UserTable where user_email = @UserEmail


    if(@user_id1 != -1 OR @user_id2 != -1)
    begin
        select * from TableMinusOne
    end
    else
    begin
        INSERT INTO [UserTable] (username, user_first_name,
user_second_name, user_email, user_password, profile_img, user_birthday,
user_level, blocked)
            VALUES (@Username, @FirstName, @SecondName, @UserEmail,
@UserPassword, @UserProfileImg, @UserBirthday, 0, 0)
    end
end

```

Para um desenvolvedor começar a trabalhar precisa de criar uma anúncio, apenas os utilizadores com o nível desenvolvedor ou *admin* e que podem criar anúncios, o anúncio será valido enquanto o desenvolvedor quiser. Para ver os anúncios basta escolher que tipo de projeto quer ver, como por exemplo *Database*, *Website Computer Application* etc.



Depois de clicar em um tipo de projeto vão aparecer todos os anúncios relacionados a esse tipo de projeto.



andrei.latis.00

WebSite + DataBase + App Android e los
Website
Faço Website pro tenho 5 andos de experiencia na area de TI
img
€4000

See More

Ao clicar *See More* o cliente será direcionado para uma página onde poder comprar o projeto ou serviço.

Dentro da DAL na *class Ads.cs* temos o método `Ad()` para buscar todos os anúncios de um certo tipo.

```
public List<Ads> Ad(string TypeProject)
{
    this.TypeProjectTxt = TypeProject;

    List<Ads> tryin = new List<Ads>();
    int idx = 0;

    connec.Open();
    SqlCommand cmd = new SqlCommand("GetAds", connec);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.AddWithValue("@ProjectType", TypeProjectTxt);

    SqlDataReader Rd = cmd.ExecuteReader();

    try
    {
        while (Rd.Read())
        {
            tryin.Add(new Ads());
            tryin[idx].AdName = Rd.GetValue(0).ToString();
            tryin[idx].TypeProjectTxt = Rd.GetValue(1).ToString();
            tryin[idx].ProjectPrice = Convert.ToDouble(Rd.GetValue(2));
            tryin[idx].AdImg1 = Rd.GetValue(3).ToString();
            tryin[idx].AdDescription = Rd.GetValue(4).ToString();
            tryin[idx].userName = Rd.GetValue(5).ToString();
            tryin[idx].userProfileImg = Rd.GetValue(6).ToString();
        }
    }
}
```

```

        idx++;
    }

    connec.Close();
    return tryin;
}
catch (Exception)
{
    connec.Close();
    return null;
}
}

```

Stored Procedure usada no método Ad () .

```


alter procedure dbo.GetAds
    @ProjectType varchar(50)
as
begin
    declare @ProjectTypeId int = -1
    select @ProjectTypeId = id_type_project from TypeProject where
txt_type_project = @ProjectType

    if @ProjectTypeId = -1
    begin
        select * from TableMinusOne
    end
    else
    begin
        select [AdTable].[ad_name], [TypeProject].[txt_type_project],
[AdTable].[project_price], [AdTable].[ad_img1], [AdTable].[ad_description],
[UserTable].[username], [UserTable].profile_img
        from [AdTable]
        inner join [TypeProject] on [AdTable].[project_type] =
[TypeProject].[id_type_project]
        inner join [UserTable] on [AdTable].[ad_creatorID] =
[UserTable].[user_id]
        where [TypeProject].[id_type_project] = @ProjectTypeId and
[AdTable].[ad_closed] = 0
    end
end

```

Profile do Desenvolvedor/Admin:

admin admin Username: admin123	Projects	Create Ad	More Skills
-----------------------------------	----------	-----------	-------------

	Trabho com base de dados e com aplicações android ha 7 anos sou muito experiente e gosto do meu trabalho
---	--


Technical Language:

JavaScript
HTML
CSS

	Pedro Padre 4 Stars
Project :	Database
Muito obrigado	


Technology:

Bootstrap
jQuery

	Simun Madeira 5 Stars
Project :	Database
gostei muito do trabalho o meu negocios envolveu muito desde ent	

Language:

Portuguese
English
Spanish

	Andrei Latis 3 Stars
Project :	Android App
O app está perfeito muito obrigado.	

Dentro do *profile* do desenvolvedor/admin podemos criar anúncios, ver projetos já criados ou adicionar novas habilidades. Também podemos ver as habilidades já existentes do desenvolvedor/admin e a opinião do cliente nos últimos 3 projetos.

Dentro do *controller* temos o método `Profile()` que chama outros métodos da DAL para buscar dados da base de dados.

```
public ActionResult Profile()
{
    if (Session["Username"] != null)
    {
        WorkStationDAL.User User = new WorkStationDAL.User();
        WorkStationDAL.Skills Skills = new WorkStationDAL.Skills();
        WorkStationDAL.Project Project = new WorkStationDAL.Project();

        ViewBag.UserProfile =
        User.UserProfile(Session["Username"].ToString());
        ViewBag.TechnicalLanguageSkills =
        Skills.TechnicalLanguageSkills(Session["Username"].ToString());
        ViewBag.TechnologySkills =
        Skills.TechnologySkills(Session["Username"].ToString());
        ViewBag.LanguageSkills =
        Skills.LanguageSkills(Session["Username"].ToString());
        ViewBag.ProjectReview =
        Project.ProjectReview(Session["Username"].ToString());
    }
}
```

```

        else
        {
            return RedirectToAction("LogIn", "User");
        }
        return View();
    }
}

```

Métodos usados método Profile ()

UserProfile().

```

public List<User> UserProfile(string Username)
{
    userName = Username;

    List<User> ProfileData = new List<User>();
    int idx = 0;

    try
    {
        connec.Open();
        SqlCommand cmd = new SqlCommand("spUserProfile", connec);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@username", userName);

        SqlDataReader Rd = cmd.ExecuteReader();

        while (Rd.Read())
        {
            ProfileData.Add(new User());
            ProfileData[idx].userProfileImg =
Rd.GetValue(0).ToString();
            ProfileData[idx].userBio = Rd.GetValue(1).ToString();
            ProfileData[idx].userFirstName = Rd.GetValue(2).ToString();
            ProfileData[idx].userSecondName =
Rd.GetValue(3).ToString();
            ProfileData[idx].userName = Rd.GetValue(4).ToString();
        }

        connec.Close();
        return ProfileData;
    }
    catch
    {
        connec.Close();
        return null;
    }
}

```

Stored Procedure usada no método UserProfile ()

```

create procedure dbo.spUserProfile
    @username varchar(50)
as
begin
    select [profile_img], [user_bio], [user_first_name], [user_second_name],
[username]
    from [UserTable]

```

```
where [username] = @username  
end
```

TechnicalLanguageSkills(), TechnologySkills(), LanguageSkills().

```
public List<Skills> TechnologySkills(string UserName)  
{  
    userName = UserName;  
  
    int idx = 0;  
    List<Skills> TechnologyS = new List<Skills>();  
  
    connec.Open();  
    SqlCommand cmd = new SqlCommand("spTechnology", connec);  
    cmd.CommandType = CommandType.StoredProcedure;  
    cmd.Parameters.AddWithValue("@username", userName);  
    SqlDataReader Rd = cmd.ExecuteReader();  
  
    try  
    {  
        while (Rd.Read())  
        {  
            TechnologyS.Add(new Skills());  
            TechnologyS[idx].Technology = Rd.GetValue(0).ToString();  
  
            idx++;  
        }  
        connec.Close();  
        return TechnologyS;  
    }  
    catch (Exception)  
    {  
        connec.Close();  
        return null;  
    }  
}  
public List<Skills> TechnicalLanguageSkills(string UserName)  
{  
    userName = UserName;  
  
    int idx = 0;  
    List<Skills> TechnicalLanguageS = new List<Skills>();  
  
    connec.Open();  
    SqlCommand cmd = new SqlCommand("spTechnicalLanguage", connec);  
    cmd.CommandType = CommandType.StoredProcedure;  
    cmd.Parameters.AddWithValue("@username", userName);  
    SqlDataReader Rd = cmd.ExecuteReader();  
  
    try  
    {  
        while (Rd.Read())  
        {  
            TechnicalLanguageS.Add(new Skills());  
            TechnicalLanguageS[idx].TechnicalLanguage = Rd.GetValue(0).ToString();  
  
            idx++;  
        }  
    }  
}
```

```

        connec.Close();
        return TechnicalLanguageS;
    }
    catch (Exception)
    {
        connec.Close();
        return null;
    }
}
public List<Skills> LanguageSkills(string UserName)
{
    userName = UserName;

    int idx = 0;
    List<Skills> LanguageS = new List<Skills>();

    connec.Open();
    SqlCommand cmd = new SqlCommand("spLanguage", connec);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.AddWithValue("@username", userName);
    SqlDataReader Rd = cmd.ExecuteReader();

    try
    {
        while (Rd.Read())
        {
            LanguageS.Add(new Skills());
            LanguageS[idx].Language = Rd.GetValue(0).ToString();

            idx++;
        }
        connec.Close();
        return LanguageS;
    }
    catch (Exception)
    {
        connec.Close();
        return null;
    }
}
}

```

Stored Procedures usadas nos métodos
 TechnicalLanguageSkills(), TechnologySkills(), LanguageSkills().

```

alter procedure dbo.spTechnology
    @username varchar(50)
as
begin
    select [Technology].[txt_technology]
    from [UserTable]
    inner join [UserTechnology] on [UserTable].[user_id] =
[UserTechnology].[id_user]
    inner join [Technology] on [UserTechnology].[id_technology] =
[Technology].[id_technology]
    where [UserTable].[username] = @username
end

alter procedure dbo.spTechnicalLanguage
    @username varchar(50)
as
begin

```



```

        select [TechnicalLanguage].[txt_technical_language]
        from [UserTable]
        inner join [UserTechnicalLanguage] on [UserTable].[user_id] =
[UserTechnicalLanguage].[id_user]
        inner join [TechnicalLanguage] on
[UserTechnicalLanguage].[id_TechnicalLanguage] =
[TechnicalLanguage].[id_technical_language]
        where [UserTable].[username] = @username
end

alter procedure dbo.spLanguage
    @username varchar(50)
as
begin
    select [LanguageTable].[txt_language]
    from [UserTable]
    inner join [UserLanguage] on [UserTable].[user_id] =
[UserLanguage].[id_user]
    inner join [LanguageTable] on [UserLanguage].[id_language] =
[LanguageTable].[id_language]
    where [UserTable].[username] = @username
end

```

ProjectReview() .

```

public List<Project> ProjectReview(string UserName)
{
    userName = UserName;

    int idx = 0;
    List<Project> ProjectR = new List<Project>();

    connec.Open();
    SqlCommand cmd = new SqlCommand("spGetProjectReview", connec);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.AddWithValue("@username", userName);
    SqlDataReader Rd = cmd.ExecuteReader();

    try
    {
        while (Rd.Read())
        {
            ProjectR.Add(new Project());
            ProjectR[idx].ProfileImg = Rd.GetValue(0).ToString();
            ProjectR[idx].userFirstName = Rd.GetValue(1).ToString();
            ProjectR[idx].userSecondName = Rd.GetValue(2).ToString();
            ProjectR[idx].Stars =
Convert.ToInt32(Rd.GetValue(3).ToString());
            ProjectR[idx].TypeProject = Rd.GetValue(4).ToString();
            ProjectR[idx].Review = Rd.GetValue(5).ToString();
            idx++;
        }
        connec.Close();
        return ProjectR;
    }
    catch (Exception)

```

```

    {
        connec.Close();
        return null;
    }
}

```

Stored Procedure usada em `ProjectReview()` .

```

alter procedure dbo.spGetProjectReview
@username varchar(50)
as
begin
    declare @DeveloperId int = 0
    set nocount on
    select @DeveloperId = user_id from UserTable where [UserTable].[username]
= @username;

    select [UserTable].[profile_img], [UserTable].user_first_name,
[UserTable].[user_second_name], [Project].[Stars],
[typeproject].[txt_type_project], [Project].[txt_review], [Project].[end_date]
    from [UserTable]
    inner join [Project] on [UserTable].[user_id] = [Project].[id_cliente]
    inner join [AdTable] on [Project].[id_ad] = [AdTable].[id_ad]
    inner join [TypeProject] on [AdTable].[project_type] =
[TypeProject].[id_type_project]
    where [AdTable].ad_creatorID = @DeveloperId and [Project].[status] = 5
    order by [Project].[end_date]
end

```

Conclusão:

No início do projeto pensava que iria acabar o projeto em um mês, mas isso não foi possível, logo no início do projeto mais especificamente na construção da base de dados tive muitas dificuldades com a organização da base de dados e com os dados que eu iria guardar na tabela. De seguida tive dificuldades na criação da DAL pois não sabia qual era a melhor opção de a DAL buscar os dados da base de dados e enviar para o *website* e vice-versa e também não sabia de que forma organizar a DAL. Mais a frente tive muita dificuldade na criação do *website* nunca foi muito bom com o design e também a utilização do modelo MVC para a construção do site me deixou confuso e com muitas dúvidas acerca de como criar um website usando modelo MVC.

Webgrafia:

stackoverflow.com

youtube.com