Instituto Superior de Engenharia Informática

Engenharia Informática


Relatório Projeto

# PWEB

Pedro Alexandre Valente Leal nº2018011482

João Luís Silva Miguel nº2018018852

# Conteúdo

# Introdução

Foi-nos proposto para esta unidade curricular um projeto para o âmbito da mesma, este projeto foi realizado aplicando o que nos foi lecionado ao longo do semestre.

Este projeto baseou-se em fomentar uma aplicação WEB do genérico Airbnb, onde os utilizadores podem alugar um certo imóvel que estejam disponíveis para tal.

# Utilizadores

Esta aplicação opera com um nível “hierárquico”, ou seja, existe os utilizadores, funcionários, os gestores e os administradores.

## Clientes

São aqueles que usam a aplicação exatamente para o seu objetivo, estes são para onde queremos apontar, é o nosso publico alvo.

O cliente quando entra é mostrado o seguinte:


*Figura 1*

Este simplesmente tem permissão para ver os alojamentos e se quiser, criar conta ou ingressar na aplicação, como amostrado em baixo onde o cliente tem uma conta criada. Pode obviamente alugar um espaço para uma certa data.


*Figura 2*

Para este projeto foi criado os seguintes clientes:

C1: cliente1@pweb.pt password: 1qazZAQ!

C2: cliente2@pweb.pt password 1qazZAQ!

## Funcionários

Os funcionários tratam da burocracia vinda do cliente de aluguer/reserva do imóvel, tais como datas de reserva, avaliações sobre que outros gestores tenham feito e entre outras.


*Figura 4*

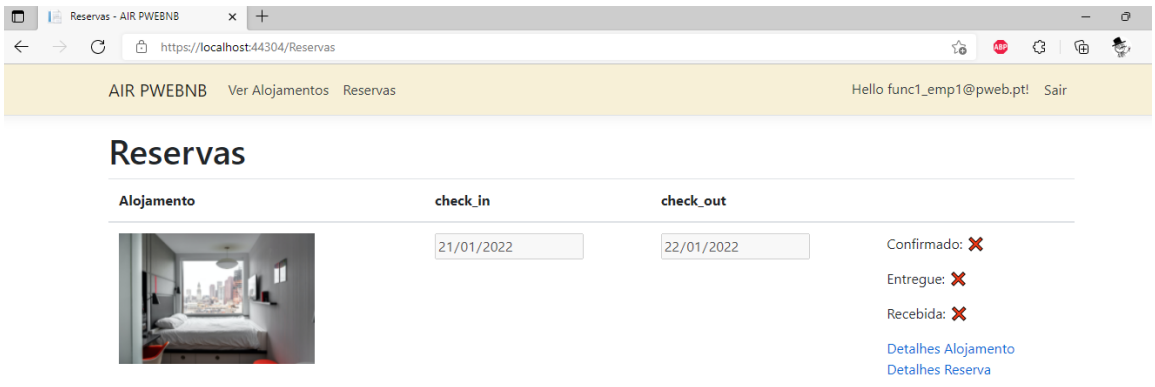


*Figura 3*

A credencial para entrar como funcionário é a seguinte:

F1: func1_emp1@pweb.pt password: 1qazZAQ!

## Gestor

O gestor é responsável pelos imoveis e pela “contratação” de funcionários, também pode fazer avaliações aos clientes que usaram os seus imóveis.


*Figura 4*

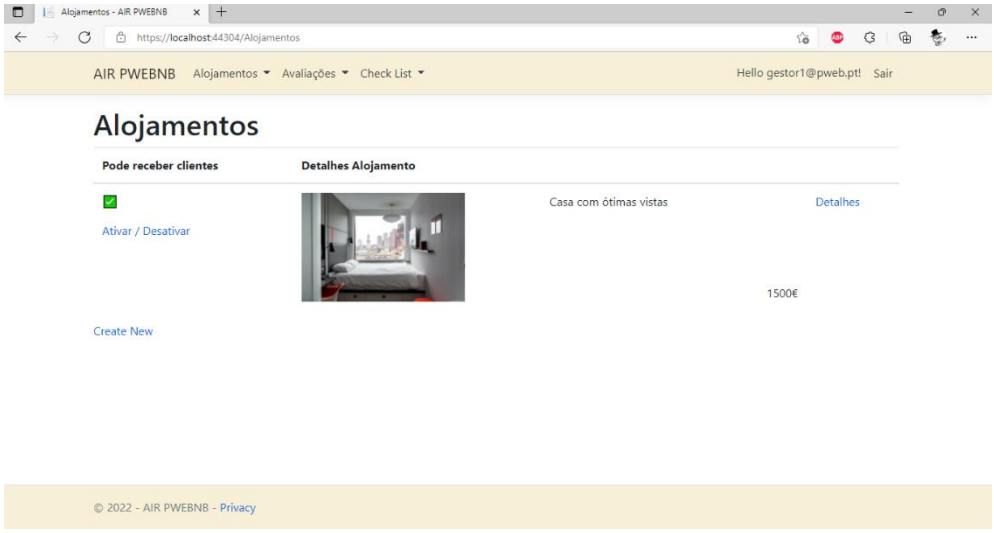


*Figura 5*

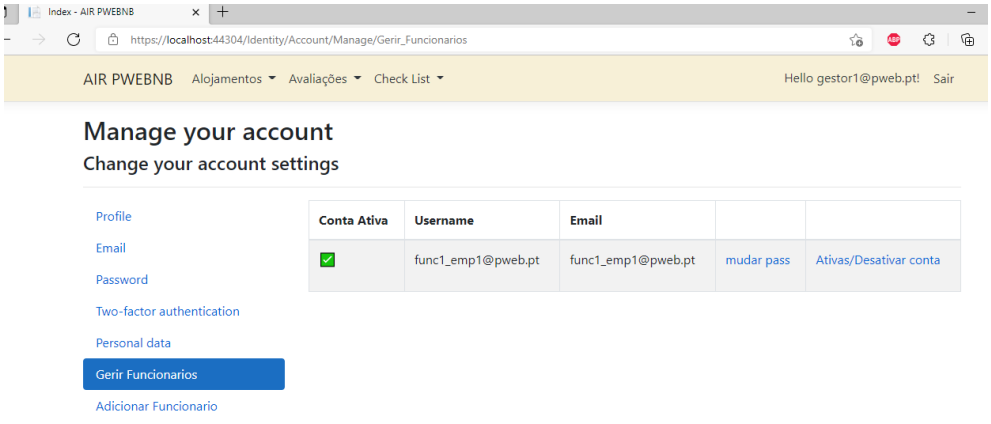


*Figura 6*

Credenciais de Gestores:

G1: gestor1@pweb.pt password:1qazZAQ!

G2: gestor2@pweb.pt password:1qazZAQ!

### Administrador

Este como tem maior poder no programa faz a gestão de todos os outros utilizadores(gestores/empresas) e também das categorias.

Credenciais:

A2: admin@pweb.pt password:1qazZAQ!

## Código Relevante

Este código figura 7 e 8 é responsável pela definição das tabelas na base dados de acordo com a classe.

```
public class Categoria
{
    public Categoria(){
        CategoriaCheck_List = new HashSet<CategoriaCheck_List>();
        alojamentos = new HashSet<Alojamento>();
    }

    [Key]
    public int CategoriaId { get; set; }
    public string nome { get; set; }

    public Boolean Ativo { get; set; }

    public ICollection<Alojamento> alojamentos { get; set; }
    public string AlojamentoId { get; set; }

    //lista Checklist
    public ICollection<CategoriaCheck_List> CategoriaCheck_List { get; set; }

}


public class CategoriaCheck_List
{
    public int CategoriaId { get; set; }
    public Categoria Categoria { get; set; }


    public int Check_ListId { get; set; }
    public Check_List Check { get; set; }
}
```

*Figura 7*

```
public class Check_List
{
    public Check_List(){
        CategoriaCheck_List = new HashSet<CategoriaCheck_List>();
    }

    [Key]
    public int id { get; set; }
    public string texto { get; set; }
    public Boolean Confirmado { get; set;  }

    public ICollection<CategoriaCheck_List> CategoriaCheck_List { get; set; }
}
```

*Figura 8*

O código a baixo é responsável pela criação de uma relação muitos para muitos entre a classe categorias/tabela e a classe checklist.

```
protected override void OnModelCreating(ModelBuilder builder)
{

    builder.Entity<CategoriaCheck_List>()
        .HasKey(x => new { x.CategoriaId, x.Check_ListId });

    builder.Entity<CategoriaCheck_List>()
        .HasOne(x => x.Categoria)
        .WithMany(x => x.CategoriaCheck_List)
        .HasForeignKey(x => x.CategoriaId);

    builder.Entity<CategoriaCheck_List>()
        .HasOne(x => x.Check)
        .WithMany(x => x.CategoriaCheck_List)
        .HasForeignKey(x => x.Check_ListId);
    builder.Entity<Categoria>().HasData(new Categoria{ nome = "quarto" , CategoriaId=1});
    base.OnModelCreating(builder);
}
```

*Figura 9*

Esta função é responsável por preencher as tabelas necessárias para o funcionamento mínimo do programa.

```
1 reference
private async Task CreateRolesAndUsers(IServiceProvider serviceProvider)
{
    var roleManager = serviceProvider.GetRequiredService<RoleManager<IdentityRole>>();
    var userManager = serviceProvider.GetRequiredService<UserManager<AppUser>>();
    string[] rolesNames = { "Admin", "Gestor", "Funcionario", "Cliente"};
    IdentityResult result;
    foreach (var namesRole in rolesNames){
        var roleExist = await roleManager.RoleExistsAsync(namesRole);
        if (!roleExist){
            result = await roleManager.CreateAsync(new IdentityRole(namesRole));
        }
    }
    var userAdmin = new AppUser{
        UserName = "admin@pweb.pt",
        Email = "admin@pweb.pt",
        pNome = "Admin",
        uNome = "Da Plataforma",
        EmailConfirmed = true
    };

    if (userManager.Users.Where(u => u.UserName == userAdmin.UserName).Count() == 0)
    {
        result = await userManager.CreateAsync(userAdmin, "1qazZAQ!");
        if (result.Succeeded)
        {
            await userManager.AddToRoleAsync(userAdmin, "Admin");
        }
    }
}
```

*Figura 10*

## Credenciais Utilizadores

C1: cliente1@pweb.pt password: 1qazZAQ!

C2: cliente2@pweb.pt password 1qazZAQ!

F1: func1_emp1@pweb.pt password: 1qazZAQ!

G1: gestor1@pweb.pt password:1qazZAQ!

G2: gestor2@pweb.pt password:1qazZAQ!

A2: admin@pweb.pt password:1qazZAQ!

# Conclusão

No âmbito da unidade curricular Programação WEB foi nos proposto este projeto para o desenvolvimento de uma Aplicação web com recursos a base de dados. Apesar de termos encontrado dificuldades ao longo do seu desenvolvimento pudemos dizer que maioria das implementações foram realizadas.