

Relatório do trabalho da disciplina de POO

# Relatório IPCAbitA - POO

---

António Jorge Magalhães da Rocha – a26052

Adelino Daniel da Rocha Vilaça – a16939

LESI-PL

Novembro de 2023

Afirmo por minha honra que não recebi qualquer apoio não autorizado na realização deste trabalho prático.  
Afirmo igualmente que não copieei qualquer material de livro, artigo, documento web ou de qualquer outra fonte exceto onde a origem estiver expressamente citada.

António Jorge Magalhães da Rocha - 26052

Adelino Daniel da Rocha Vilaça - 16939

## Índice

|   |    |
|---|----|
| 1. INTRODUÇÃO .....   | 1  |
| 2. CLASSES .....  | 2  |
| 2.1. Classe Pessoa.....                                     | 2  |
| 2.2. Classe Aluno.....                                      | 3  |
| 2.2.1. Classe Aluno – Método GerarAlunoID .....             | 4  |
| 2.2.2. Classe Aluno – Método RegistrarNovoAluno .....       | 4  |
| 2.2.3. Classe Aluno – Método GuardaAluno .....              | 5  |
| 2.2.4. Classe Aluno – Método ValidarLoginAluno .....        | 5  |
| 2.2.5. Classe Aluno – Método LerAlunos.....                 | 6  |
| 2.2.6. Classe Aluno – Método EliminarAluno .....            | 6  |
| 2.3. Classe Senhorio .....                                  | 7  |
| 2.3.1. Classe Senhorio - Método RegistrarNovoSenhorio ..... | 7  |
| 2.3.2. Classe Senhorio - Método GuardaSenhorio .....        | 8  |
| 2.3.3. Classe Senhorio - Método ValidarLoginSenhorio .....  | 8  |
| 2.3.4. Classe Senhorio– Método LerSenhorios .....           | 9  |
| 2.3.5. Classe Senhorio– Método EliminarSenhorio .....       | 10 |
| 2.4. Classe Admin.....                                      | 10 |
| 2.4.1. Classe Admin - Método GerarAdminId .....             | 11 |
| 2.4.2. Classe Admin - Método RegistrarNovoAdmin.....        | 11 |
| 2.4.3. Classe Admin - Método GuardaAdmin.....               | 11 |
| 2.4.4. Classe Admin - Método ValidarLoginAdmin.....         | 12 |
| 2.4.5. Classe Admin - Método LerAdmins .....                | 13 |
| 2.5. Classe Pagamento (Não implementado) .....              | 14 |
| 2.6. Classe Quarto.....                                     | 15 |
| 2.6.1. Classe Quarto - Método RegistrarNovoQuarto .....     | 15 |
| 2.6.2. Classe Quarto - Método GuardaQuarto .....            | 16 |

|   |    |
|---|----|
| 2.6.3. Classe Quarto - Método LerQuartos .....                | 16 |
| 2.6.4. Classe Quarto - Método EliminarQuarto.....             | 17 |
| 2.7. Classe Servico .....                                     | 17 |
| 2.7.1. Classe Servico - Método RegistrarNovoServico.....      | 18 |
| 2.7.2. Classe Servico- Método GuardaServico.....              | 18 |
| 2.7.3. Classe Servico- Método LerServicos .....               | 19 |
| 2.7.4. Classe Servico- Método EliminarServico.....            | 20 |
| 2.8. Classe Menu.....   | 20 |
| 2.8.1. Classe Menu - Método MenuPrincipal .....               | 20 |
| 2.8.2. Classe Menu - Método MenuRegistrarAluno.....           | 21 |
| 2.8.3. Classe Menu - Método MenuAluno .....                   | 22 |
| 2.8.4. Classe Menu - Método DisplayMenuAluno .....            | 23 |
| 2.8.5. Classe Menu - Método MenuRegistrarSenhorio.....        | 25 |
| 2.8.6. Classe Menu - Método MenuSenhorio.....                 | 25 |
| 2.8.7. Classe Menu - Método DisplayMenuSenhorio.....          | 26 |
| 2.8.8. Classe Menu - Método MenuRegistrarQuarto.....          | 28 |
| 2.8.9. Classe Menu - Método MenuRegistrarServico .....        | 29 |
| 2.8.10. Classe Menu - Método MenuAdmin .....                  | 30 |
| 2.8.11. Classe Menu - Método DisplayMenuAdmin .....           | 30 |
| 2.9. Classe Program .....                                     | 33 |
| 2.9.1. Classe Program- Método Main.....                       | 33 |
| 2.10. Testes .....  | 35 |
| 2.10.1. Testes - Ecrã Inicial.....                            | 35 |
| 2.10.2. Testes – Novo Aluno (Registrar) .....                 | 35 |
| 2.10.3. Testes – Novo Senhorio (Registrar) .....              | 36 |
| 2.10.4. Testes – Login Aluno .....                            | 36 |
| 2.10.5. Testes – Menu Login Aluno .....                       | 37 |
| 2.10.6. Testes – Menu Login Aluno (Ver dados do Quarto) ..... | 37 |

|   |    |
|---|----|
| 2.10.7. Testes – Menu Login Aluno (Ver Serviços) .....          | 38 |
| 2.10.8. Testes – Menu Login Aluno (Contactar Senhorio) .....    | 38 |
| 2.10.9. Testes – Menu Login Aluno (Contactar Admin) .....       | 39 |
| 2.10.10. Testes – Login Senhorio .....                          | 39 |
| 2.10.11. Testes – Menu Login Senhorio .....                     | 40 |
| 2.10.12. Testes – Menu Login Senhorio (Adicionar Quarto) .....  | 40 |
| 2.10.13. Testes – Menu Login Senhorio (Ver Quartos) .....       | 41 |
| 2.10.14. Testes – Menu Login Senhorio (Adicionar Serviço) ..... | 41 |
| 2.10.15. Testes – Menu Login Senhorio (Ver Serviços) .....      | 42 |
| 2.10.16. Testes – Login Admin .....                             | 42 |
| 2.10.17. Testes – Menu Login Admin .....                        | 43 |
| 2.10.18. Testes – Menu Login Admin (Ver Alunos) .....           | 43 |
| 2.10.19. Testes – Menu Login Admin (Ver Senhorios) .....        | 44 |
| 2.10.20. Testes – Menu Login Admin (Ver Quartos) .....          | 44 |
| 2.10.21. Testes – Menu Login Admin (Ver Serviços) .....         | 45 |
| 3. ESTRUTURAS DE DADOS .....                                    | 45 |
| 3.1. LISTAS .....   | 45 |
| 3.2. FICHEIROS CSV .....  | 45 |
| 4. CONCLUSÃO .....  | 48 |

## Lista de Tabelas

Tabela 1 — <descrição da tabela>

2

## Lista de Figuras

Figura 1 — <descrição da figura>

2

## 1. Introdução

O objetivo deste trabalho é implementar um programa em C# para servir como plataforma e representação do projeto IPCAbitA, referente ao aluguer de quartos para alunos do IPCA. Irá ser desenvolvido usando Classes como Pessoa, Aluno, Senhorio, Admin, etc para melhorar a organização e acesso aos dados correspondentes. Irá também usar listas e ficheiros CSV como estruturas de dados do projeto.

Este trabalho seguirá uma metodologia conforme apreendida em aula que inclui desde a definição de requisitos até a documentação do trabalho.

## 2. Classes

### 2.1. Classe Pessoa

```
7 referências
public class Pessoa
{
    7 referências
    public string Nome { get; set; }
    7 referências
    public string DataNascimento { get; set; }
    11 referências
    public string Email { get; set; }
    10 referências
    public string Password { get; set; }
    0 referências
    public bool IsBlocked { get; set; }

    3 referências
    public Pessoa(string nome, string dataNascimento, string email, string password)
    {
        Nome = nome;
        DataNascimento = dataNascimento;
        Email = email;
        Password = password;
    }
}
```

Representação de um Utilizador com as propriedades base de uma Pessoa



## 2.2. Classe Aluno

```
25 referências
public class Aluno : Pessoa
{
    List<Aluno> listaAlunos = new List<Aluno>();

    private static int proximoAlunoId = 16000;

    3 referências
    public string AlunoId { get; set; }
    3 referências
    public string Curso { get; set; }
    3 referências
    public string Instituicao { get; set; }

    2 referências
    public Aluno(string alunoId, string nome, string dataNascimento, string email, string password, string curso, string instituicao)
        : base(nome, dataNascimento, email, password)
    {
        AlunoId = alunoId;
        Nome = nome;
        DataNascimento = dataNascimento;
        Curso = curso;
        Email = email;
        Password = password;
        Instituicao = instituicao;
    }

    1 referência
    public override string ToString()
    {
        return $"{AlunoId},{Nome},{DataNascimento},{Curso},{Instituicao},{Email},{Password}";
    }
}
```

Representação de um Aluno tendo propriedades herdadas da Classe Pessoa com propriedades em acréscimo como AlunoId, Curso, etc.

### 2.2.1. Classe Aluno – Método GerarAlunoId

```
//gerar id incrementável a partir de 16000 até 26999
0 referências
public static int GerarAlunoId()
{
    int alunoId = proximoAlunoId; // Default ID is now proximoAlunoId
    if (File.ReadLines("../../Dados/dadosaluno.csv").Any())
    {
        var lastLine = File.ReadLines("../../Dados/dadosaluno.csv").Last();
        var lastAlunoId = int.Parse(lastLine.Split(',')[0]);
        alunoId = lastAlunoId + 1;
    }
    return alunoId;
}
```

Método GerarAlunoId para gerar um ID único novo, sempre que for feito um Registo de um Aluno, incrementando a partir do primeiro campo da última linha do ficheiro dadosaluno.csv.

### 2.2.2. Classe Aluno – Método RegistrarNovoAluno

Método RegistrarNovoAluno para registar um novo aluno inquilino guardando num ficheiro CSV, usando o Método GuardaAluno, os dados da lista correspondente.

```
//registar novo aluno
1 referência
public static void RegistrarNovoAluno(List<Aluno> listaAlunos, string alunoId, string nome, string dataNascimento,
    string curso, string instituicao, string email, string password)
{
    Aluno novoAluno = new Aluno(alunoId, nome, dataNascimento, curso, instituicao, email, password);
    listaAlunos.Add(novoAluno);
    GuardaAluno(listaAlunos, "../../Dados/dadosaluno.csv");

    Console.WriteLine("Aluno registado com sucesso!");
    Console.WriteLine("\n-----\n");
}
```

### 2.2.3. Classe Aluno – Método GuardaAluno

```
2 referências
public static void GuardaAluno(List<Aluno> listaAlunos, string filePath)
{
    using (StreamWriter sw1 = new StreamWriter(filePath, true))
    {
        foreach (Aluno aluno in listaAlunos)
        {
            sw1.WriteLine($"{aluno.AlunoId},{aluno.Nome},{aluno.DataNascimento},{aluno.Curso},{aluno.Instituicao}," +
                           $"{aluno.Email},{aluno.Password}\n");
        }
    }
}
```

Método GuardaAluno para guardar dados da lista num ficheiro CSV.

### 2.2.4. Classe Aluno – Método ValidarLoginAluno

```
//validar login pelo csv
1 referência
public static bool ValidarLoginAluno(string email, string password)
{
    List<Aluno> listaAlunos = LerAlunos("../../Dados/dadosaluno.csv");

    foreach (var aluno in listaAlunos)
    {
        if (aluno.Email == email && aluno.Password == password)
        {
            return true;
        }
        else
        {
            Console.WriteLine("Erro: Email ou Password incorretos. Tente novamente.");
        }
    }
    return false;
}
```

Método ValidarLoginAluno para validar Login usando email e password guardados no ficheiro CSV.

### 2.2.5. Classe Aluno – Método LerAlunos

```
//mostrar lista
4 referências
public static List<Aluno> LerAlunos(string filePath)
{
    List<Aluno> alunos = new List<Aluno>();

    string[] lines = File.ReadAllLines(filePath);

    foreach (string line in lines)
    {
        string[] splitLine = line.Split(',');
        string[] fields = new string[7];

        for (int i = 0; i < splitLine.Length && i < fields.Length; i++)
        {
            fields[i] = splitLine[i];
        }

        string id = fields[0];
        string nome = fields[1];
        string dataNascimento = fields[2];
        string curso = fields[3];
        string instituicao = fields[4];
        string email = fields[5];
        string password = fields[6];

        Aluno aluno = new Aluno(id, nome, dataNascimento, curso, instituicao, email, password);
        alunos.Add(aluno);
    }

    return alunos;
}
```

Método LerAlunos para mostrar lista com os dados guardados do ficheiro CSV.

### 2.2.6. Classe Aluno – Método EliminarAluno

```
0 referências
public static bool EliminarAluno(string email)
{
    List<Aluno> listaAlunos = LerAlunos("../../Dados/dadosaluno.csv");

    foreach (var aluno in listaAlunos)
    {
        if (aluno.Email == email)
        {
            listaAlunos.Remove(aluno);
            GuardaAluno(listaAlunos, "../../Dados/dadosaluno.csv");
            return true;
        }
    }

    return false;
}
```

Método EliminarAluno para eliminar dados do aluno no ficheiro CSV, através do email.

## 2.3. Classe Senhorio

```

24 referências
public class Senhorio : Pessoa
{
    List<Senhorio> listaSenhorios = new List<Senhorio>();

    2 referências
    public Senhorio(string nome, string dataNascimento, string email, string password)
        : base(nome, dataNascimento, email, password)
    {
    }

    1 referência
    public override string ToString()
    {
        return $"{Nome},{DataNascimento},{Email},{Password}";
    }
}

```

Representação de um Senhorio tendo propriedades herdadas da Classe Pessoa e construtor vazio.

### 2.3.1. Classe Senhorio - Método RegistrarNovoSenhorio

```

//registar novo senhorio
1 referência
public static void RegistrarNovoSenhorio(List<Senhorio> listaSenhorios, string nome, string dataNascimento,
    string email, string password)
{
    Senhorio novoSenhorio = new Senhorio(nome, dataNascimento, email, password);
    listaSenhorios.Add(novoSenhorio);
    GuardaSenhorio(listaSenhorios, "../../../Dados/dadosseñhorio.csv");

    Console.WriteLine("Senhorio registado com sucesso!");
    Console.WriteLine("\n-----\n");
}

```

Método RegistrarNovoSenhorio para registar um novo Senhorio guardando num ficheiro CSV, usando o Método GuardaSenhorio, a lista de dados correspondente ao mesmo.

### 2.3.2. Classe Senhorio - *Método GuardaSenhorio*

```
1 referência
public static void GuardaSenhorio(List<Senhorio> listaSenhorios, string filePath)
{
    using (StreamWriter sw2 = new StreamWriter(filePath, true))
    {
        foreach (Senhorio senhorio in listaSenhorios)
        {
            sw2.WriteLine($"{senhorio.Nome},{senhorio.DataNascimento},{senhorio.Email},{senhorio.Password}\n");
        }
    }
}
```

Método GuardaSenhorio para guardar dados da lista num ficheiro CSV.

### 2.3.3. Classe Senhorio - *Método ValidarLoginSenhorio*

```
//validar login do Senhorio
1 referência
public static bool ValidarLoginSenhorio(string email, string password)
{
    List<Senhorio> listaSenhorios = LerSenhorios("../Dados/dadosenhorio.csv");

    foreach (var senhorio in listaSenhorios)
    {
        if (senhorio.Email == email && senhorio.Password == password)
        {
            return true;
        }
        else
        {
            Console.WriteLine("Erro: Email ou Password incorretos. Tente novamente.");
        }
    }

    return false;
}
```

Método ValidarLoginSenhorio para validar Login usando email e password guardados no ficheiro CSV.

### 2.3.4. Classe Senhorio- Método LerSenhorios

```
//mostrar lista
3 referências
public static List<Senhorio> LerSenhorios(string filePath)
{
    List<Senhorio> lerSenhorios = new List<Senhorio>();

    string[] lines = File.ReadAllLines(filePath);

    foreach (string line in lines)
    {
        string[] splitLine = line.Split(',');
        string[] fields = new string[4];

        for (int i = 0; i < splitLine.Length && i < fields.Length; i++)
        {
            fields[i] = splitLine[i];
        }

        string nome = fields[0];
        string dataNascimento = fields[1];
        string email = fields[2];
        string password = fields[3];

        Senhorio senhorio = new Senhorio(nome, dataNascimento, email, password);
        lerSenhorios.Add(senhorio);
    }

    return lerSenhorios;
}
```

Método LerSenhorios para mostrar lista com os dados guardados do ficheiro CSV.

### 2.3.5. Classe Senhorio– Método *EliminarSenhorio*

```

0 referências
public static bool EliminarSenhorio(string email)
{
    string filePath = "../../../Dados/dadossenhario.csv";
    List<string> lines = new List<string>(File.ReadAllLines(filePath));

    // Find the line with the user's email
    int lineIndex = lines.FindIndex(line => line.Split(',')[0] == email);

    // If the line was found, remove it
    if (lineIndex != -1)
    {
        lines.RemoveAt(lineIndex);
        File.WriteAllLines(filePath, lines);
        return true;
    }

    return false;
}

```

Método EliminarAluno para eliminar dados do aluno no ficheiro CSV, através do email.

## 2.4. Classe Admin

```

17 referências
public class Admin : Pessoa
{
    private static int proximoAdminId = 1;
    1 referência
    public int AdminId { get; private set; }

    2 referências
    private Admin(string nome, string dataNascimento, string email, string password) :
        base(nome, dataNascimento, email, password)
    {
        AdminId = GerarAdminId();
    }
}

```

Representação de um Admin tendo propriedades herdadas da Classe Pessoa como propriedades estáticas/predefinidas.



### 2.4.1. Classe Admin - Método GerarAdminId

```
//gerar id incrementável
1 referência
private int GerarAdminId()
{
    int novoAdminId = proximoAdminId;
    proximoAdminId++;

    return novoAdminId;
}
```

Método GerarAdminId para gerar um ID único novo, sempre que for feito um Registo de um Admin.

### 2.4.2. Classe Admin - Método RegistrarNovoAdmin

```
//registar novo admin
0 referências
public static void RegistrarNovoAdmin(List<Admin> listaAdmins, string nome, string dataNascimento, string email, string password)
{
    Admin novoAdmin = new Admin(nome, dataNascimento, email, password);
    listaAdmins.Add(novoAdmin);
    GuardaAdmin(listaAdmins, "../../../Dados/dadosadmin.csv");

    Console.WriteLine("Admin registado com sucesso!");
    Console.WriteLine("\n-----\n");
}
```

Método RegistrarNovoAdmin para registar um novo Admin guardando num ficheiro CSV, usando o Método GuardaAdmin, a lista de dados correspondente ao mesmo.

### 2.4.3. Classe Admin - Método GuardaAdmin

```
//guardar dados do admin
1 referência
public static void GuardaAdmin(List<Admin> listaAdmins, string fileName)
{
    using (StreamWriter sw3 = new StreamWriter(fileName, true))
    {
        foreach (Admin admin in listaAdmins)
        {
            sw3.WriteLine($"{admin.Nome},{admin.DataNascimento},{admin.Email},{admin.Password}\n");
        }
    }
}
```

Método GuardaAdmin para guardar dados da lista num ficheiro CSV.

#### 2.4.4. Classe Admin - Método *ValidarLoginAdmin*

```
//validar login do Admin
1 referência
public static bool ValidarLoginAdmin(string email, string password)
{
    List<Admin> listaAdmins = LerAdmins("../../Dados/dadosadmin.csv");

    foreach (var admin in listaAdmins)
    {
        if (admin.Email == email && admin.Password == password)
        {
            return true;
        }
        else
        {
            Console.WriteLine("Erro: Email ou Password incorretos. Tente novamente.");
        }
    }

    return false;
}
```

Método ValidarLoginAdmin para validar Login usando email e password guardados no ficheiro CSV.

### 2.4.5. Classe Admin - Método LerAdmins

```
//mostrar lista
2 referências
public static List<Admin> LerAdmins(string fileName)
{
    List<Admin> lerAdmins = new List<Admin>();

    string[] lines = File.ReadAllLines(fileName);

    foreach (string line in lines)
    {
        string[] splitLine = line.Split(',');
        string[] fields = new string[4];

        for (int i = 0; i < splitLine.Length && i < fields.Length; i++)
        {
            fields[i] = splitLine[i];
        }

        string nome = fields[0];
        string dataNascimento = fields[1];
        string email = fields[2];
        string password = fields[3];

        Admin admin = new Admin(nome, dataNascimento, email, password);
        lerAdmins.Add(admin);
    }

    return lerAdmins;
}
```

Método LerSenhorios para mostrar lista com os dados guardados do ficheiro CSV.

## 2.5. Classe Pagamento (Não implementado)

```
1 referência
public class Pagamento
{
    1 referência
    public int PagamentoId { get; set; }
    1 referência
    public decimal Valor { get; set; }
    1 referência
    public DateTime DataPagamento { get; set; }
    1 referência
    public string Descricao { get; set; }

    // Constructor
    0 referências
    public Pagamento(int pagamentoId, decimal valor, DateTime dataPagamento, string descricao)
    {
        PagamentoId = pagamentoId;
        Valor = valor;
        DataPagamento = dataPagamento;
        Descricao = descricao;
    }
}
```

Representação da Classe Pagamento tendo como propriedades o PagamentoId, Valor, Descrição, etc.

## 2.6. Classe Quarto

```

27 referências
public class Quarto
{
    List<Quarto> listaQuartos = new List<Quarto>();

    3 referências
    public string QuartoId { get; set; }
    4 referências
    public string Numero { get; set; }
    3 referências
    public string Capacidade { get; set; }
    3 referências
    public string Descricao { get; set; }

    2 referências
    public Quarto(string quartoId, string numero, string capacidade, string descricao)
    {
        QuartoId = quartoId;
        Numero = numero;
        Capacidade = capacidade;
        Descricao = descricao;
    }

    3 referências
    public override string ToString()
    {
        return $"QuartoId: {QuartoId}, Numero: {Numero}, Capacidade: {Capacidade}, Descricao: {Descricao}";
    }
}

```

Representação da Classe Quarto tendo como propriedades o QuartoId, Número do Quarto, Capacidade, etc.

### 2.6.1. Classe Quarto - Método RegistrarNovoQuarto

```

//registar novo Quarto
1 referência
public static void RegistrarNovoQuarto(List<Quarto> listaQuartos, string quartoId, string numero,
    string capacidade, string descricao)
{
    Quarto novoQuarto = new Quarto(quartoId, numero, capacidade, descricao);
    listaQuartos.Add(novoQuarto);
    GuardaQuarto(listaQuartos, "../../../Dados/dadosquarto.csv");

    Console.WriteLine("Quarto registado com sucesso!");
    Console.WriteLine("\n-----\n");
}

```

Método RegistrarNovoQuarto para registar um novo Quarto guardando num ficheiro CSV, usando o Método GuardaQuarto, a lista de dados correspondente ao mesmo.

### 2.6.2. Classe Quarto - Método GuardaQuarto

```
2 referências
public static void GuardaQuarto(List<Quarto> listaQuartos, string filePath)
{
    using (StreamWriter sw4 = new StreamWriter(filePath, true))
    {
        foreach (Quarto quarto in listaQuartos)
        {
            sw4.WriteLine($"{quarto.QuartoId},{quarto.Numero},{quarto.Capacidade},{quarto.Descricao}\n");
        }
    }
}
```

Método GuardaQuarto para guardar dados da lista num ficheiro CSV.

### 2.6.3. Classe Quarto - Método LerQuartos

```
//mostrar lista
4 referências
public static List<Quarto> LerQuartos(string filePath)
{
    List<Quarto> lerQuartos = new List<Quarto>();

    string[] lines = File.ReadAllLines(filePath);

    foreach (string line in lines)
    {
        string[] splitLine = line.Split(',');
        string[] fields = new string[4];

        for (int i = 0; i < splitLine.Length && i < fields.Length; i++)
        {
            fields[i] = splitLine[i];
        }

        string quartoId = fields[0];
        string numero = fields[1];
        string capacidade = fields[2];
        string descricao = fields[3];

        Quarto quarto = new Quarto(quartoId, numero, capacidade, descricao);
        lerQuartos.Add(quarto);
    }

    return lerQuartos;
}
```

Método LerSenhorios para mostrar lista com os dados guardados do ficheiro CSV.

### 2.6.4. Classe Quarto - Método EliminarQuarto

```

0 referências
public static bool EliminarQuarto(string numero)
{
    List<Quarto> listaQuartos = LerQuartos("../../Dados/dadosquarto.csv");

    foreach (var quarto in listaQuartos)
    {
        if (quarto.Numero == numero)
        {
            listaQuartos.Remove(quarto);
            GuardaQuarto(listaQuartos, "../../Dados/dadosquarto.csv");
            return true;
        }
    }
    return false;
}

```

Método EliminarQuarto para eliminar dados do quarto no ficheiro CSV, através do número do quarto.

### 2.7. Classe Servico

```

25 referências
public class Servico
{
    3 referências
    public string ServicoId { get; set; }
    4 referências
    public string Nome { get; set; }
    3 referências
    public string Preco { get; set; }
    3 referências
    public string Descricao { get; set; }

    2 referências
    public Servico(string servicoId, string nome, string preco, string descricao)
    {
        ServicoId = servicoId;
        Nome = nome;
        Preco = preco;
        Descricao = descricao;
    }

    3 referências
    public override string ToString()
    {
        return $"ServicoId: {ServicoId}, Nome: {Nome}, Preco: {Preco}, Descricao: {Descricao}";
    }
}

```

Representação da Classe Serviço tendo como propriedades o ServicoId, Nome, Preço, etc.

### 2.7.1. Classe *Service* - Método *RegistarNovoService*

```
//registar novo Service
1 referência
public static void RegistarNovoService(List<Service> listaServicos, string servicoId, string nome,
    string preco, string descricao)
{
    Service novoService = new Service(servicoId, nome, preco, descricao);
    listaServicos.Add(novoService);
    GuardaService(listaServicos, "../../../Dados/dadoservice.csv");

    Console.WriteLine("Service registado com sucesso!");
    Console.WriteLine("\n-----\n");
}
```

Método `RegistarNovoService` para registar um novo Serviço guardando num ficheiro CSV, usando o Método `GuardaService`, a lista de dados correspondente ao mesmo.

### 2.7.2. Classe *Service*- Método *GuardaService*

```
//Guardar Service
2 referências
public static void GuardaService(List<Service> listaServicos, string filePath)
{
    using (StreamWriter sw5 = new StreamWriter(filePath, true))
    {
        foreach (Service servico in listaServicos)
        {
            sw5.WriteLine($"{servico.ServicoId},{servico.Nome},{servico.Preco},{servico.Descricao}\n");
        }
    }
}
```

Método `GuardaService` para guardar dados da lista num ficheiro CSV.



### 2.7.3. Classe Service- Método LerServicos

```
//mostrar lista
4 referências
public static List<Servico> LerServicos(string filePath)
{
    List<Servico> lerServicos = new List<Servico>();

    string[] lines = File.ReadAllLines(filePath);

    foreach (string line in lines)
    {
        string[] splitLine = line.Split(',');
        string[] fields = new string[4];

        for (int i = 0; i < splitLine.Length && i < fields.Length; i++)
        {
            fields[i] = splitLine[i];
        }

        string servicoId = fields[0];
        string nome = fields[1];
        string preco = fields[2];
        string descricao = fields[3];

        Servico servico = new Servico(servicoId, nome, preco, descricao);
        lerServicos.Add(servico);
    }

    return lerServicos;
}
```

Método LerServicos para mostrar lista com os dados guardados do ficheiro CSV.

### 2.7.4. Classe Servico- Método EliminarServico

```
0 referências
public static bool EliminarServico(string nome)
{
    List<Servico> listaServicos = LerServicos("../../Dados/dadoservico.csv");

    foreach (var servico in listaServicos)
    {
        if (servico.Nome == nome)
        {
            listaServicos.Remove(servico);
            GuardaServico(listaServicos, "../../Dados/dadoservico.csv");
            return true;
        }
    }
    return false;
}
```

Método EliminarServico para eliminar dados do serviço no ficheiro CSV, através do nome do mesmo.

## 2.8. Classe Menu

### 2.8.1. Classe Menu - Método MenuPrincipal

```
1 referência
public static void MenuPrincipal()
{
    Console.WriteLine("Bem-vindo ao IPCAbitA:");
    Console.WriteLine("1. Novo Aluno (Registar)");
    Console.WriteLine("2. Login Aluno");
    Console.WriteLine("3. Novo Senhorio (Registar)");
    Console.WriteLine("4. Login Senhorio");
    Console.WriteLine("5. Login Admin");
    Console.WriteLine("6. Refresh Menu");
    Console.WriteLine("7. Ajuda");
    Console.WriteLine("0. Sair");
}
```

Método MenuPrincipal que servirá de menu inicial do programa para apresentar ao user.

### 2.8.2. Classe Menu - Método MenuRegistrarAluno

```
//Menu Registrar Aluno
1 referência
public static void MenuRegistrarAluno(List<Aluno> listaAlunos)
{
    //registar um novo aluno
    Console.WriteLine("----- Registrar Novo Aluno -----");

    //alunoId
    string alunoId = "16000";

    //nome
    Console.Write("Nome: ");
    string nome = Console.ReadLine();

    //datanasc
    Console.Write("Data de Nascimento (dd/MM/yyyy): ");
    string dataNascimento = Console.ReadLine();

    Console.Write("Instituição Universitária: ");
    string instituicao = Console.ReadLine();

    Console.Write("Curso: ");
    string curso = Console.ReadLine();

    Console.Write("Email da Universidade: ");
    string email = Console.ReadLine();

    Console.Write("Password: ");
    string password = Console.ReadLine();

    Aluno.RegistarNovoAluno(listaAlunos, alunoId, nome, dataNascimento, curso, instituicao, email, password);
}
```

Método MenuRegistrarAluno para recolher os dados do user ao registar-se como aluno.

### 2.8.3. Classe Menu - Método MenuAluno

```
1 referência
public static void MenuAluno(List<Aluno> listaAlunos)
{
    // Login como aluno
    Console.WriteLine("----- Login Aluno -----");

    Console.Write("Email: ");
    string loginAEmail = Console.ReadLine();

    Console.Write("Password: ");
    string loginAPass = Console.ReadLine();

    //verificar aluno
    bool validarAluno = Aluno.ValidarLoginAluno(loginAEmail, loginAPass);
    if (validarAluno)
    {
        Console.Clear();
        Console.WriteLine("Login bem-sucedido! Bem-vindo");
        Console.WriteLine("\n-----\n");
        Menu.DisplayMenuAluno();
    }
    else
    {
        Console.Clear();
        Console.WriteLine("Falha no login. Verifique as suas credenciais.");
        Console.WriteLine("\n-----\n");
    }
}
```

Método MenuAluno para apresentar ao aluno os campos para o Login

### 2.8.4. Classe Menu - Método *DisplayMenuAluno*

```
1 referência
public static void DisplayMenuAluno()
{
    Console.WriteLine("1 - Ver dados do Quarto");
    Console.WriteLine("2 - Ver Serviços");
    Console.WriteLine("3 - Contactar Senhorio");
    Console.WriteLine("4 - Contactar Admin");
    Console.WriteLine("5 - Eliminar Conta");
    Console.WriteLine("0 - Logout");
    Console.Write("Escolha uma opção: ");
    string? escolhaAluno = Console.ReadLine();

    switch (escolhaAluno)
    {
        case "1":
            Console.Clear();
            List<Quarto> quartos = Quarto.LerQuartos("../Dados/dadosquarto.csv");
            foreach (Quarto quarto in quartos)
            {
                Console.WriteLine(quarto.ToString());
            }
            Console.WriteLine();
            Console.WriteLine("\n-----\n");
            break;

        case "2":
            Console.Clear();
            List<Servico> servicos = Servico.LerServicos("../Dados/dadosservico.csv");
            foreach (Servico servico in servicos)
            {
                Console.WriteLine(servico.ToString());
            }
            Console.WriteLine();
            Console.WriteLine("\n-----\n");
            break;
    }
}
```

Método `DisplayMenuAluno` para apresentar ao aluno as opções do menu após efetuar corretamente o Login. Casos 1 e 2 do switchcase correspondem à leitura dos dados dos ficheiros `dadosquarto.csv` e `dadosservico.csv` da pasta `Dados`.

```
case "3":
    Console.Clear();
    Console.WriteLine("Email do senhorio: ");
    string emailSenhorio = Console.ReadLine();
    Console.WriteLine("Mensagem: ");
    string mensagemSenhorio = Console.ReadLine();
    Console.WriteLine("Email enviado com sucesso!");
    Console.WriteLine("\n-----\n");
    break;

case "4":
    Console.Clear();
    Console.WriteLine("Email do admin: ");
    string emailAdmin = Console.ReadLine();
    Console.WriteLine("Mensagem: ");
    string mensagemAdmin = Console.ReadLine();
    Console.WriteLine("Email enviado com sucesso!");
    Console.WriteLine("\n-----\n");
    break;

case "5":
    Console.Clear();
    //Console.WriteLine("Tem a certeza que pretende eliminar a sua conta? (sim/nao): ");
    //string confirm = Console.ReadLine();
    //if (confirm.ToLower() == "sim")
    //{
    //    bool sucesso = Aluno.EliminarAluno(loginAemail);
    //    if (sucesso)
    //    {
    //        Console.WriteLine("\nConta eliminada com sucesso!");
    //    }
    //    else
    //    {
    //        Console.WriteLine("\nErro a eliminar a conta!");
    //    }
    //}
    Console.WriteLine("Resolver");
    Console.WriteLine("\n-----\n");
    break;
```

Casos 3 e 4 correspondem a um exemplo de contacto via email através das opções que a plataforma, eventualmente, poderia ter através do uso de api's/servidores de email. Caso 5 corresponde a uma forma de eliminação de dados de um aluno, mas ainda será necessário rever a forma de chamada do email usado no Login, logo não implementado na versão atual.

### 2.8.5. Classe Menu - Método MenuRegistrarSenhorio

```
1 referência
public static void MenuRegistrarSenhorio(List<Senhorio> listaSenhorios)
{
    //registar um novo senhorio
    Console.WriteLine("----- Registrar Novo Senhorio -----");

    //nome
    Console.Write("Nome: ");
    string nomeSenhorio = Console.ReadLine();

    //datanasc
    Console.Write("Data de Nascimento (dd/MM/yyyy): ");
    string dataNascimentoSenhor = Console.ReadLine();

    Console.Write("Email: ");
    string emailSenhorio = Console.ReadLine();

    Console.Write("Password: ");
    string passwordSenhorio = Console.ReadLine();

    Senhorio.RegistarNovoSenhorio(listaSenhorios, nomeSenhorio, dataNascimentoSenhor, emailSenhorio, passwordSenhorio);
}
```

Método MenuRegistrarSenhorio para recolher os dados do user ao registar-se como senhorio.

### 2.8.6. Classe Menu - Método MenuSenhorio

```
1 referência
public static void MenuSenhorio(List<Senhorio> listaSenhorios)
{
    //login como senhorio
    Console.WriteLine("----- Login Senhorio -----");
    Console.Write("Email: ");
    string loginSEmail = Console.ReadLine();

    Console.Write("Password: ");
    string loginSPass = Console.ReadLine();

    bool validarSenhorio = Senhorio.ValidarLoginSenhorio(loginSEmail, loginSPass);

    //verificar senhorio
    if (validarSenhorio)
    {
        Console.Clear();
        Console.WriteLine("Login bem-sucedido! Bem-vindo");
        Console.WriteLine("\n-----\n");
        Menu.DisplayMenuSenhorio();
    }
    else
    {
        Console.Clear();
        Console.WriteLine("Falha no login. Verifique as suas credenciais.");
        Console.WriteLine("\n-----\n");
    }
}
```

Método MenuSenhorio para apresentar ao senhorio os campos para o Login

### 2.8.7. Classe Menu - Método *DisplayMenuSenhorio*

```
1 referência
public static void DisplayMenuSenhorio()
{
    Console.WriteLine("1 - Adicionar Quarto");
    Console.WriteLine("2 - Ver Quartos");
    Console.WriteLine("3 - Adicionar Serviço");
    Console.WriteLine("4 - Ver serviços");
    Console.WriteLine("5 - Contactar Aluno");
    Console.WriteLine("6 - Contactar Admin");
    Console.WriteLine("7 - Eliminar Conta");
    Console.WriteLine("0 - Logout");
    Console.Write("Escolha uma opção: ");
    string? escolhaAluno = Console.ReadLine();

    switch (escolhaAluno)
    {
        case "1":
            Console.Clear();
            List<Quarto> listaQuartos = new List<Quarto>();
            MenuRegistarQuarto(listaQuartos);
            Console.WriteLine("\n-----\n");
            break;

        case "2":
            Console.Clear();
            List<Quarto> quartos = Quarto.LerQuartos("../../Dados/dadosquarto.csv");
            foreach (Quarto quarto in quartos)
            {
                Console.WriteLine(quarto.ToString());
            }
            Console.WriteLine();
            Console.WriteLine("\n-----\n");
            break;
    }
}
```

Método `DisplayMenuSenhorio` para apresentar ao senhorio as opções do menu após efetuar corretamente o Login. O Caso 1 do switchcase corresponde ao registo de dados de um quarto, através do método `MenuRegistarQuarto` (Ver Abaixo). O Caso 2 corresponde à leitura dos dados do ficheiro `dadosquarto.csv` da pasta `Dados` através do método `LerQuartos`.



```
case "3":
    Console.Clear();
    List<Servico> listaServicos = new List<Servico>();
    MenuRegistrarServico(listaServicos);
    Console.WriteLine("\n-----\n");
    break;

case "4":
    Console.Clear();
    List<Servico> servicos = Servico.LerServicos("../../Dados/dadosservico.csv");
    foreach (Servico servico in servicos)
    {
        Console.WriteLine(servico.ToString());
    }
    Console.WriteLine();
    Console.WriteLine("\n-----\n");
    break;

case "5":
    Console.Clear();
    Console.WriteLine("Email do aluno: ");
    string emailAluno = Console.ReadLine();
    Console.WriteLine("Mensagem: ");
    string mensagemAluno = Console.ReadLine();
    Console.WriteLine("Email enviado com sucesso!");
    Console.WriteLine("\n-----\n");
    break;

case "6":
    Console.Clear();
    Console.WriteLine("Email do admin: ");
    string emailAdmin = Console.ReadLine();
    Console.WriteLine("Mensagem: ");
    string mensagemAdmin = Console.ReadLine();
    Console.WriteLine("Email enviado com sucesso!");
    Console.WriteLine("\n-----\n");
    break;
```

Caso 3 do switchcase corresponde ao registo de dados de um serviço, através do método MenuRegistrarServico (Ver Abaixo). O Caso 4 corresponde à leitura dos dados do ficheiro dadoservico.csv da pasta Dados através do método LerServicos. Casos 5 e 6 correspondem a um exemplo de contacto via email através das opções que a plataforma, eventualmente, poderia ter através do uso de api's/servidores de email.

```

case "7":
    Console.Clear();
    //Console.Write("Tem a certeza que pretende eliminar a sua conta? (sim/nao): ");
    //string confirm = Console.ReadLine();
    //if (confirm.ToLower() == "sim")
    //{
    //    bool sucesso = Senhorio.EliminarSenhorio(loginSemail);
    //    if (sucesso)
    //    {
    //        Console.WriteLine("\nConta eliminada com sucesso!");
    //    }
    //    else
    //    {
    //        Console.WriteLine("\nErro a eliminar a conta!");
    //    }
    //}
    Console.WriteLine("Resolver");
    Console.WriteLine("\n-----\n");
    break;

```

Caso 7 corresponde a uma forma de eliminação de dados de um aluno, mas ainda será necessário rever a forma de chamada do email usado no Login, logo não implementado na versão atual.

### 2.8.8. Classe Menu - Método MenuRegistrarQuarto

```

1 referência
public static void MenuRegistrarQuarto(List<Quarto> listaQuartos)
{
    //registar um novo quarto
    Console.WriteLine("----- Registrar Novo Quarto -----");

    string quartoId = "1";

    //nome
    Console.Write("Numero do Quarto: ");
    string numeroQuarto = Console.ReadLine();

    //datanasc
    Console.Write("Capacidade do Quarto: ");
    string capacidadeQuarto = Console.ReadLine();

    Console.Write("Breve Descricao do Quarto: ");
    string descricaoQuarto = Console.ReadLine();

    Quarto.RegistarNovoQuarto(listaQuartos, quartoId, numeroQuarto, capacidadeQuarto, descricaoQuarto);
}

```

Método MenuRegistrarQuarto para recolher os dados relativos ao quarto a ser registado através da conta do senhorio.

### 2.8.9. Classe Menu - Método MenuRegistrarServico

```
1 referência
public static void MenuRegistrarServico(List<Servico> listaServicos)
{
    //registar um novo servico
    Console.WriteLine("----- Registrar Novo Serviço -----");

    string servicoId = "1";

    //nome
    Console.Write("Nome do Serviço: ");
    string nome = Console.ReadLine();

    //datanasc
    Console.Write("Preço do Serviço: ");
    string preco = Console.ReadLine();

    Console.Write("Descricao do Serviço: ");
    string descricaoServico = Console.ReadLine();

    Servico.RegistarNovoServico(listaServicos, servicoId, nome, preco, descricaoServico);
}
```

Método MenuRegistrarServico para recolher os dados relativos ao serviço a ser registado através da conta do senhorio.

### 2.8.10. Classe Menu - Método MenuAdmin

```
1 referência
public static void MenuAdmin(List<Admin> listaAdmins)
{
    //login como admin
    Console.WriteLine("----- Login Admin -----");
    Console.Write("Email: ");
    string loginEmail = Console.ReadLine();

    Console.Write("Password: ");
    string loginPass = Console.ReadLine();

    //procurar admin
    bool validarAdmin = Admin.ValidarLoginAdmin(loginEmail, loginPass);

    //verificar admin
    if (validarAdmin)
    {
        Console.Clear();
        Console.WriteLine("Login bem-sucedido! Bem-vindo");
        Console.WriteLine("\n-----\n");
        Menu.DisplayMenuAdmin();
    }
    else
    {
        Console.Clear();
        Console.WriteLine("Falha no login. Verifique as suas credenciais.");
        Console.WriteLine("\n-----\n");
    }
}
```

Método MenuAdmin para apresentar ao admin os campos para o Login

### 2.8.11. Classe Menu - Método DisplayMenuAdmin

```

1 referência
public static void DisplayMenuAdmin()
{
    Console.WriteLine("1 - Ver Alunos");
    Console.WriteLine("2 - Ver Senhórios");
    Console.WriteLine("3 - Ver Quartos");
    Console.WriteLine("4 - Ver Serviços");
    Console.WriteLine("5 - Eliminar Aluno");
    Console.WriteLine("6 - Eliminar Senhório");
    Console.WriteLine("7 - Eliminar Quarto");
    Console.WriteLine("8 - Eliminar Serviço");
    Console.WriteLine("0 - Logout");
    Console.Write("Escolha uma opção: ");
    string? escolhaAluno = Console.ReadLine();

    switch (escolhaAluno)
    {
        case "1":
            Console.Clear();
            List<Aluno> alunos = Aluno.LerAlunos("../Dados/dadosaluno.csv");
            foreach (Aluno aluno in alunos)
            {
                Console.WriteLine(aluno.ToString());
            }
            Console.WriteLine();
            Console.WriteLine("\n-----\n");
            break;

        case "2":
            Console.Clear();
            List<Senhorio> senhorios = Senhorio.LerSenhorios("../Dados/dadosenhorio.csv");
            foreach (Senhorio senhorio in senhorios)
            {
                Console.WriteLine(senhorio.ToString());
            }
            Console.WriteLine();
            Console.WriteLine("\n-----\n");
            break;
    }
}

```

Método DisplayMenuSenhorio para apresentar ao admin as opções do menu após efetuar corretamente o Login. Oss Caso 1,2,3 e 4 do switchcase corresponde à leitura dos dados dos alunos guardados nos ficheiros dadosaluno.csv, dadosenhorio.csv, dadosquarto.csv e dadosservico.csv da pasta Dados, respetivamente.

```
case "5":
    Console.Clear();
    //Console.Write("Email do aluno que pretende eliminar: ");
    //string email = Console.ReadLine();
    //bool sucesso = Aluno.EliminarAluno(email);
    //if (sucesso)
    //{
    //    Console.WriteLine("Conta do aluno eliminada com sucesso!");
    //}
    //else
    //{
    //    Console.WriteLine("Erro ao eliminar conta de aluno.");
    //}
    Console.WriteLine("Resolver");
    Console.WriteLine("\n-----\n");
    break;
case "6":
    Console.Clear();
    //Console.Write("Email do senhorio que pretende eliminar: ");
    //string email = Console.ReadLine();
    //bool sucesso = Senhorio.EliminarSenhorio(email);
    //if (sucesso)
    //{
    //    Console.WriteLine("Conta do senhorio eliminada com sucesso!");
    //}
    //else
    //{
    //    Console.WriteLine("Erro ao eliminar conta de senhorio.");
    //}
    Console.WriteLine("Resolver");
    Console.WriteLine("\n-----\n");
    break;
```

Os Caso 5,6,7 e 8 correspondem a uma forma de eliminação de dados de um aluno/senhorio/quarto/serviço, mas ainda será necessário rever a forma de entrada do argumento do método, logo não implementado na versão atual.

## 2.9. Classe Program

### 2.9.1. Classe Program- Método Main

```
0 referências
static void Main(string[] args)
{
    List<Aluno> listaAlunos = new List<Aluno>();
    List<Senhorio> listaSenhorios = new List<Senhorio>();
    List<Admin> listaAdmins = new List<Admin>();

    bool continuarRegisto = true;

    while (continuarRegisto == true)
    {
        Menu.MenuPrincipal();

        Console.Write("Escolha uma opção: ");
        string? escolha = Console.ReadLine();

        switch (escolha)
        {
            case "1":
                Console.Clear();
                Menu.MenuRegistrarAluno(listaAlunos);
                break;

            case "2":
                Console.Clear();
                Menu.MenuAluno(listaAlunos);
                break;

            case "3":
                Console.Clear();
                Menu.MenuRegistrarSenhorio(listaSenhorios);
                break;

            case "4":
                Console.Clear();
                Menu.MenuSenhorio(listaSenhorios);
                break;

            case "5":
                Console.Clear();
                Menu.MenuAdmin(listaAdmins);
                break;
        }
    }
}
```

Método Main para chamar os métodos criados nas classes acima referidas, para manipulação/entrada de dados por parte do utilizador. É usado um ciclo while para garantir que o MenuPrincipal é continuamente apresentado ao user, até que o mesmo opte por sair (Caso 0).

No Caso 1 e 2 são chamados os métodos para registo de dados de um aluno e senhorio, respetivamente. Nos Casos 3, 4 e 5 são apresentados os Menus de Login ao Aluno, Senhorio e Admin, respetivamente.

```
case "6":
    Console.Clear();
    Console.WriteLine("----- RefreshMenu -----");

    if (listaAlunos != null)
    {
        listaAlunos = Aluno.LerAlunos("../ ../ ../Dados/dadosaluno.csv");
        Console.WriteLine("Lista de Alunos atualizada");
    }
    else
    {
        Console.WriteLine("ERRO! Lista de Alunos não atualizada ");
    }

    if (listaSenhorios != null)
    {
        listaSenhorios = Senhorio.LerSenhorios("../ ../ ../Dados/dadossenhorio.csv");
        Console.WriteLine("Lista de Senhorios atualizada");
    }
    else
    {
        Console.WriteLine("ERRO! Lista de Senhorios não atualizada ");
    }

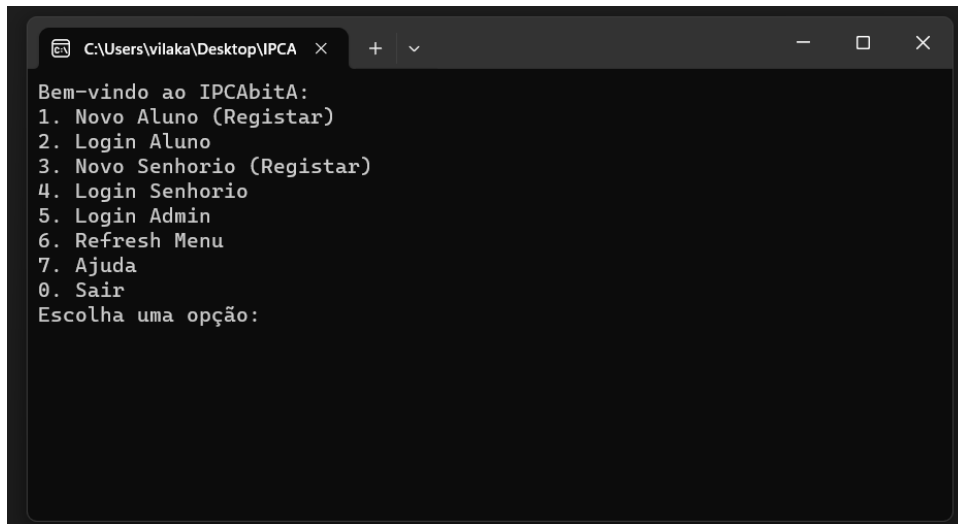
    if (listaAdmins != null)
    {
        listaAdmins = Admin.LerAdmins("../ ../ ../Dados/dadosadmin.csv");
        Console.WriteLine("Lista de Admins atualizada");
    }
    else
    {
        Console.WriteLine("ERRO! Lista de Admins não atualizada ");
    }
    Console.WriteLine("\n-----\n");
    break;
```

No Caso 6 é apresentada uma maneira de forçar uma correção das listas, caso existe alguma falha de leitura de dados previamente inseridos nos ficheiros csv. (Medida Preventiva visto que não aconteceu, ainda)



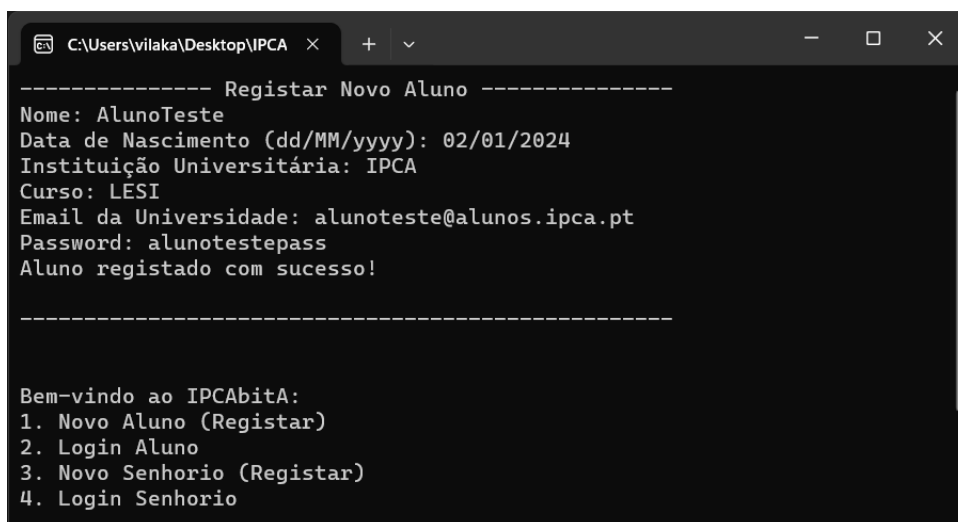
## 2.10. Testes

### 2.10.1. Testes - Ecrã Inicial



```
C:\Users\vilaka\Desktop\IPCA x + v
Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
5. Login Admin
6. Refresh Menu
7. Ajuda
0. Sair
Escolha uma opção:
```

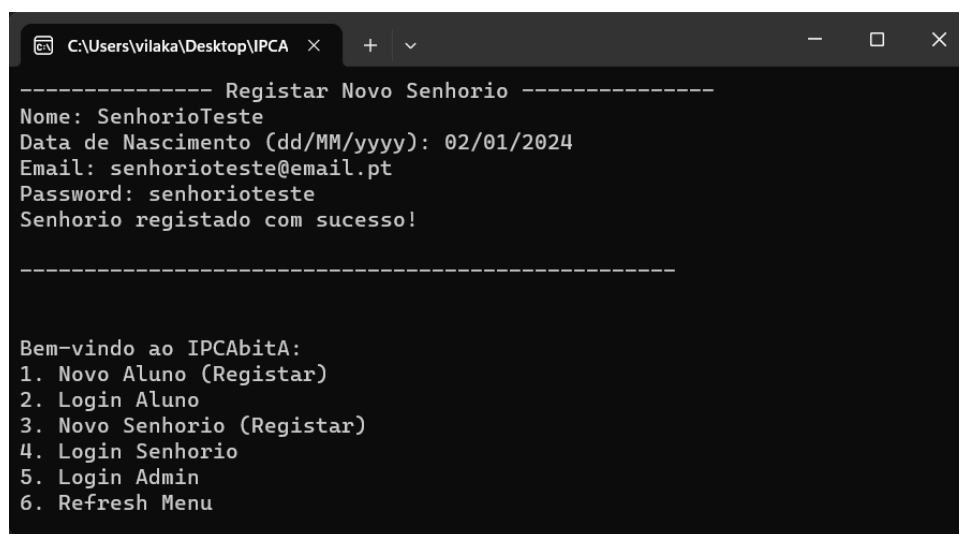
### 2.10.2. Testes – Novo Aluno (Registar)



```
C:\Users\vilaka\Desktop\IPCA x + v
----- Registar Novo Aluno -----
Nome: AlunoTeste
Data de Nascimento (dd/MM/yyyy): 02/01/2024
Instituição Universitária: IPCA
Curso: LESI
Email da Universidade: alunoteste@alunos.ipca.pt
Password: alunotestepass
Aluno registado com sucesso!
-----

Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
```

### 2.10.3. Testes – Novo Senhorio (Registar)

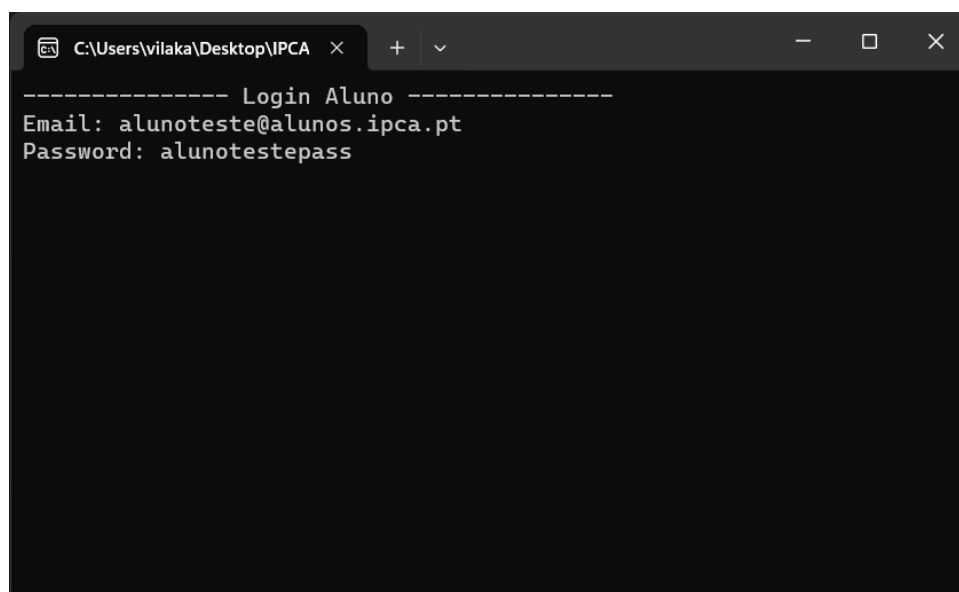


```
C:\Users\vilaka\Desktop\IPCA x + v
----- Registar Novo Senhorio -----
Nome: SenhorioTeste
Data de Nascimento (dd/MM/yyyy): 02/01/2024
Email: senhorioteste@email.pt
Password: senhorioteste
Senhorio registado com sucesso!

-----

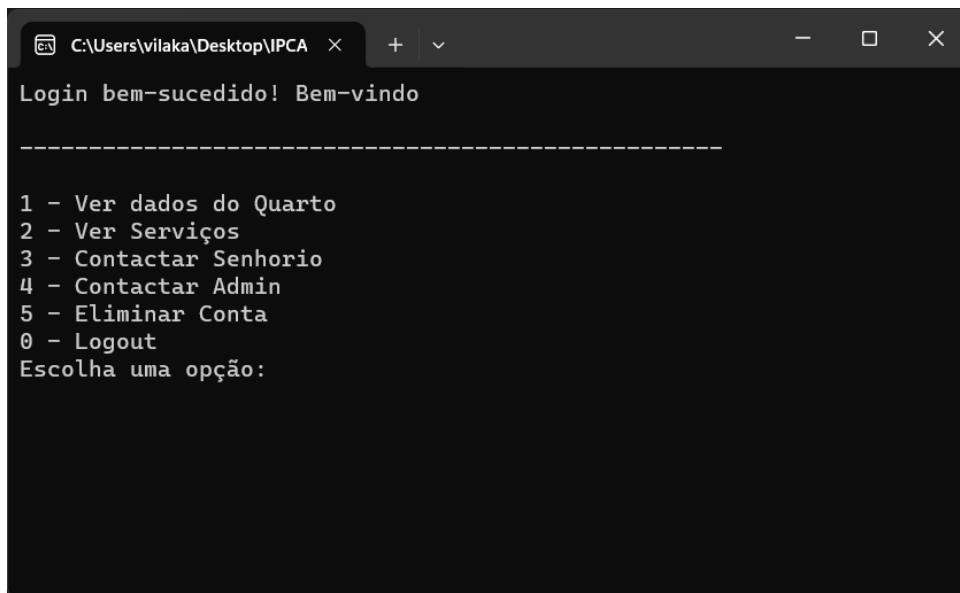
Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
5. Login Admin
6. Refresh Menu
```

### 2.10.4. Testes – Login Aluno



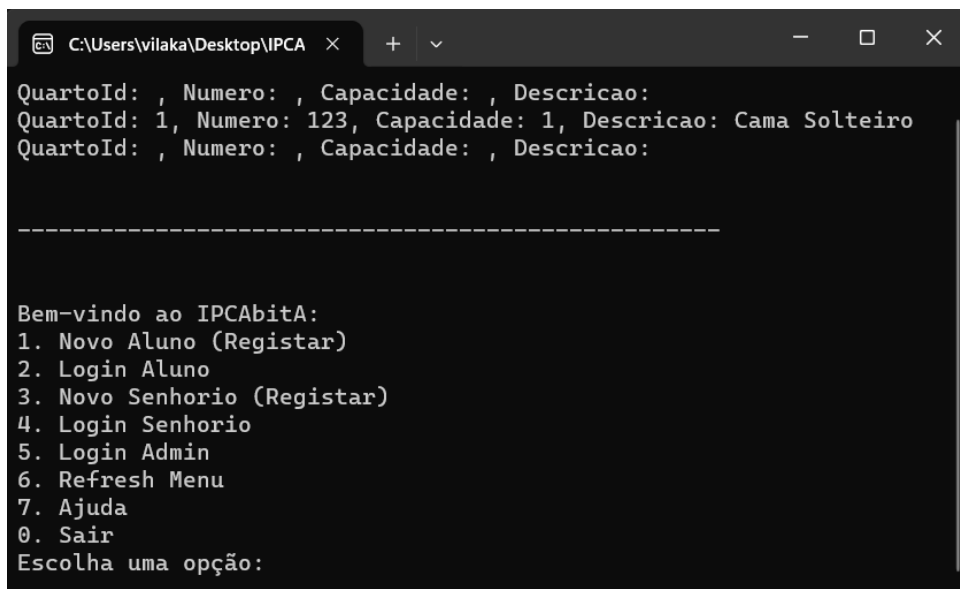
```
C:\Users\vilaka\Desktop\IPCA x + v
----- Login Aluno -----
Email: alunoteste@alunos.ipca.pt
Password: alunotestepass
```

### 2.10.5. Testes – Menu Login Aluno



```
C:\Users\vilaka\Desktop\IPCA x + v
Login bem-sucedido! Bem-vindo
-----
1 - Ver dados do Quarto
2 - Ver Serviços
3 - Contactar Senhorio
4 - Contactar Admin
5 - Eliminar Conta
0 - Logout
Escolha uma opção:
```

### 2.10.6. Testes – Menu Login Aluno (Ver dados do Quarto)



```
C:\Users\vilaka\Desktop\IPCA x + v
QuartoId: , Numero: , Capacidade: , Descricao:
QuartoId: 1, Numero: 123, Capacidade: 1, Descricao: Cama Solteiro
QuartoId: , Numero: , Capacidade: , Descricao:
-----
Bem-vindo ao IPCAbita:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
5. Login Admin
6. Refresh Menu
7. Ajuda
0. Sair
Escolha uma opção:
```

### 2.10.7. Testes – Menu Login Aluno (Ver Serviços)

```
C:\Users\wilaka\Desktop\IPCA x + v - □ ×

ServicoId: 1, Nome: Canalizacao, Preco: 125eur, Descricao: Servico
de canalizacao
ServicoId: , Nome: , Preco: , Descricao:

-----

Bem-vindo ao IPCAbita:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
5. Login Admin
6. Refresh Menu
7. Ajuda
0. Sair
Escolha uma opção:
```

### 2.10.8. Testes – Menu Login Aluno (Contactar Senhorio)

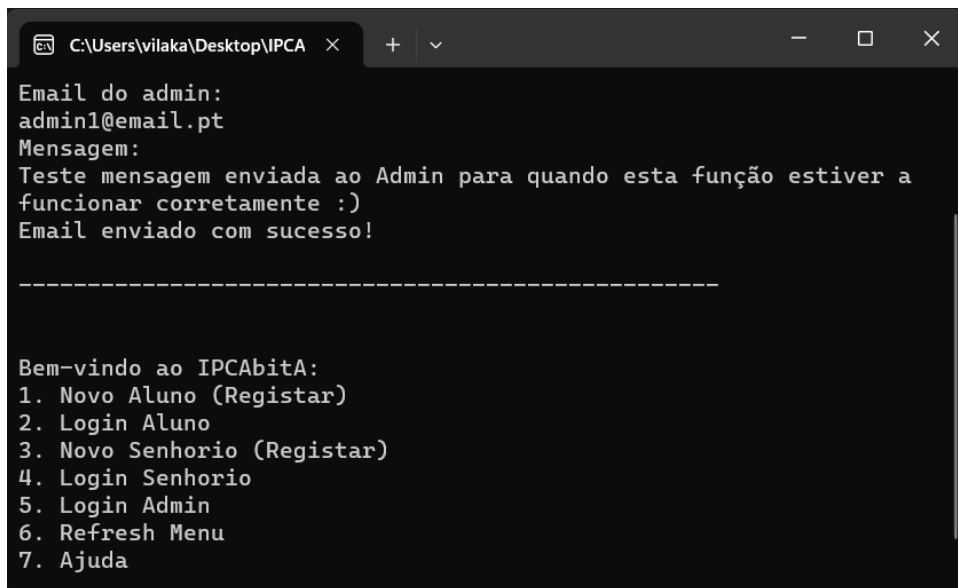
```
C:\Users\wilaka\Desktop\IPCA x + v - □ ×

Email do senhorio:
senhorioteste@email.pt
Mensagem:
Teste mensagem enviada para quando esta função estiver a funcionar
corretamente :)
Email enviado com sucesso!

-----

Bem-vindo ao IPCAbita:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
5. Login Admin
6. Refresh Menu
7. Ajuda
```

### 2.10.9. Testes – Menu Login Aluno (Contactar Admin)

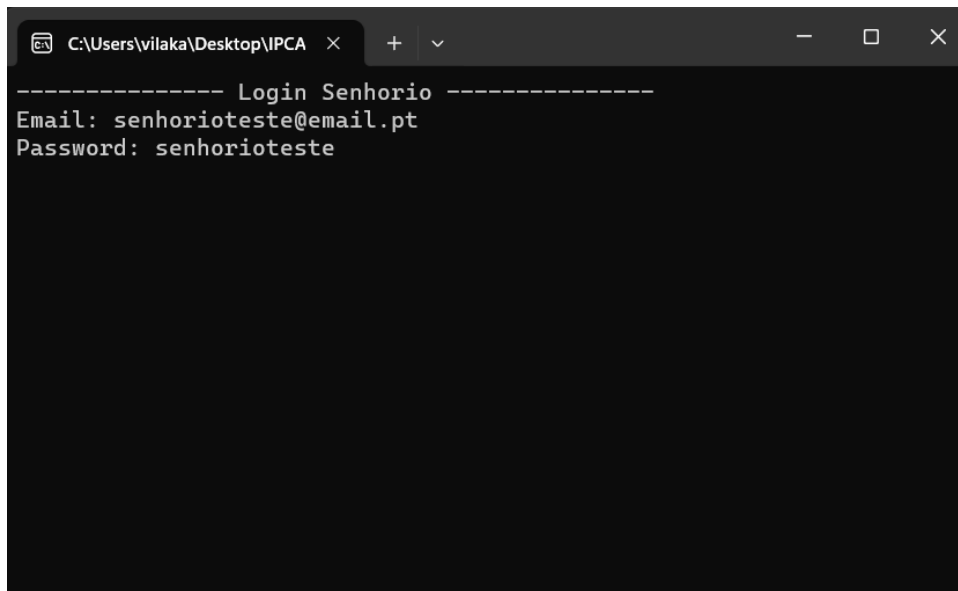


```
C:\Users\vilaka\Desktop\IPCA x + v
Email do admin:
admin1@email.pt
Mensagem:
Teste mensagem enviada ao Admin para quando esta função estiver a
funcionar corretamente :)
Email enviado com sucesso!

-----

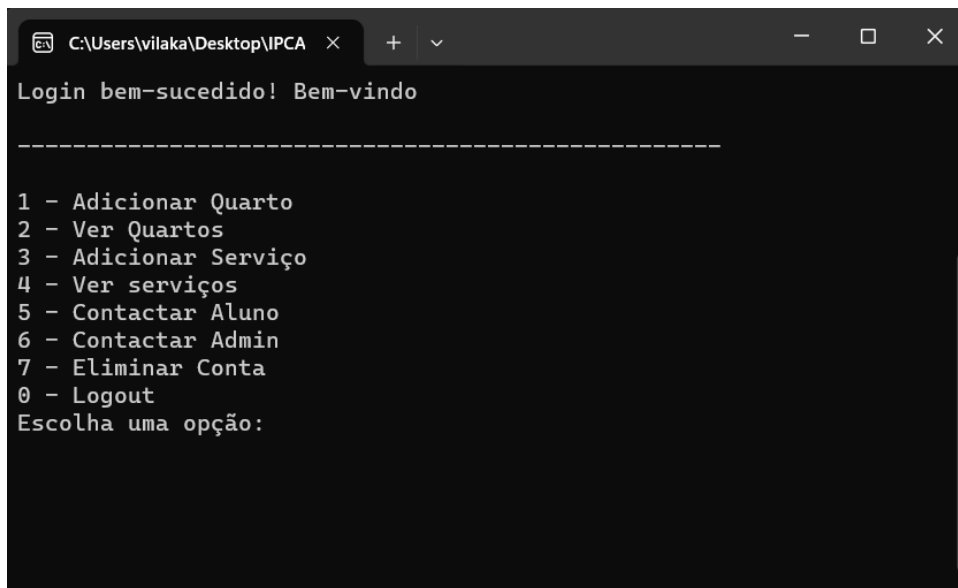
Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
5. Login Admin
6. Refresh Menu
7. Ajuda
```

### 2.10.10. Testes – Login Senhorio



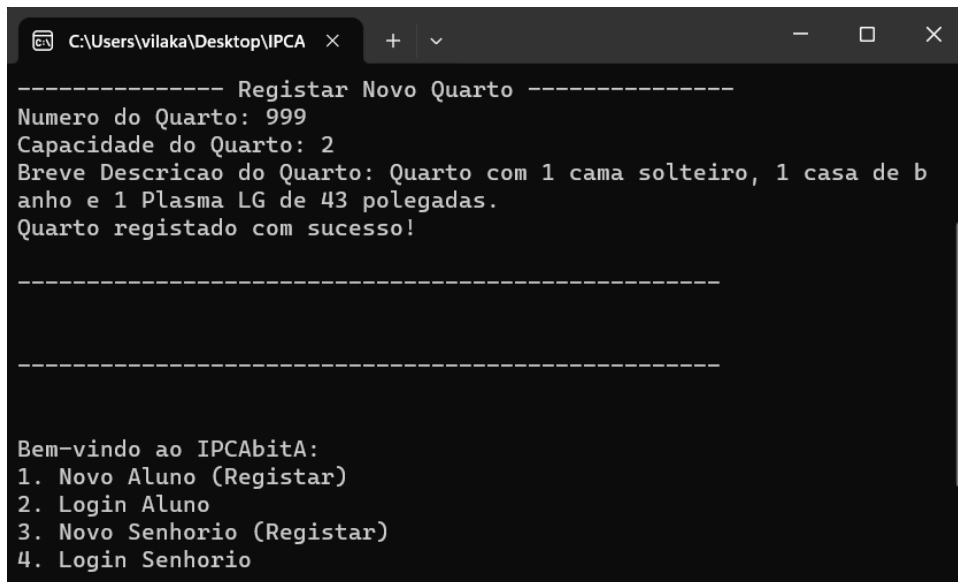
```
C:\Users\vilaka\Desktop\IPCA x + v
----- Login Senhorio -----
Email: senhorioteste@email.pt
Password: senhorioteste
```

### 2.10.11. Testes – Menu Login Senhorio



```
C:\Users\vilaka\Desktop\IPCA x + v
Login bem-sucedido! Bem-vindo
-----
1 - Adicionar Quarto
2 - Ver Quartos
3 - Adicionar Serviço
4 - Ver serviços
5 - Contactar Aluno
6 - Contactar Admin
7 - Eliminar Conta
0 - Logout
Escolha uma opção:
```

### 2.10.12. Testes – Menu Login Senhorio (Adicionar Quarto)



```
C:\Users\vilaka\Desktop\IPCA x + v
----- Registrar Novo Quarto -----
Numero do Quarto: 999
Capacidade do Quarto: 2
Breve Descricao do Quarto: Quarto com 1 cama solteiro, 1 casa de b
anho e 1 Plasma LG de 43 polegadas.
Quarto registado com sucesso!
-----

-----

Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registrar)
2. Login Aluno
3. Novo Senhorio (Registrar)
4. Login Senhorio
```

### 2.10.13. Testes – Menu Login Senhorio (Ver Quartos)

```
C:\Users\vilaka\Desktop\IPCA x + v
QuartoId: 1, Numero: 1, Capacidade: 2, Descricao: Cama de Casal
QuartoId: , Numero: , Capacidade: , Descricao:
QuartoId: 1, Numero: 123, Capacidade: 1, Descricao: Cama Solteiro
QuartoId: , Numero: , Capacidade: , Descricao:
QuartoId: 1, Numero: 999, Capacidade: 2, Descricao: Quarto com 1 c
ama solteiro
QuartoId: , Numero: , Capacidade: , Descricao:

-----

Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
5. Login Admin
```

### 2.10.14. Testes – Menu Login Senhorio (Adicionar Serviço)

```
C:\Users\vilaka\Desktop\IPCA x + v
----- Registrar Novo Serviço -----
Nome do Serviço: Serviço de Manutenção Geral
Preço do Serviço: 95 euros
Descricao do Serviço: Troca de lâmpadas dos Quartos, de halógeno
para LED.
Serviço registado com sucesso!

-----

-----

Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
```

### 2.10.15. Testes – Menu Login Senhorio (Ver Serviços)

```
C:\Users\vilaka\Desktop\IPCA x + v
ServicoId: 1, Nome: Canalizacao, Preco: 125eur, Descricao: Servico
de canalizacao
ServicoId: , Nome: , Preco: , Descricao:
ServicoId: 1, Nome: Serviço de Manutenção Geral, Preco: 95 euros,
Descricao: Troca de lâmpadas dos Quartos
ServicoId: , Nome: , Preco: , Descricao:

-----

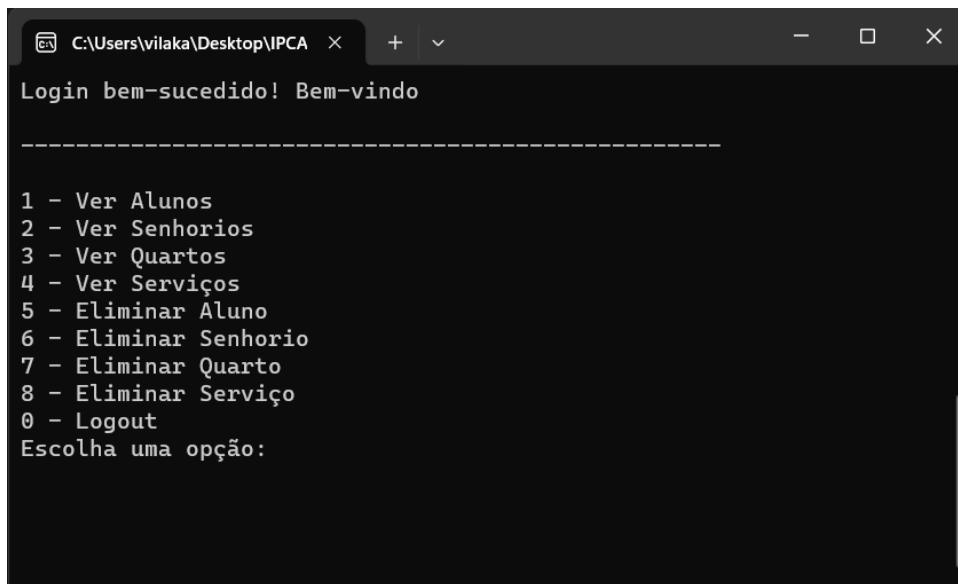
Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
5. Login Admin
6. Refresh Menu
```

### 2.10.16. Testes – Login Admin

```
C:\Users\vilaka\Desktop\IPCA x + v
----- Login Admin -----
Email: admin1@email.pt
Password: adminpass1
```



### 2.10.17. Testes – Menu Login Admin

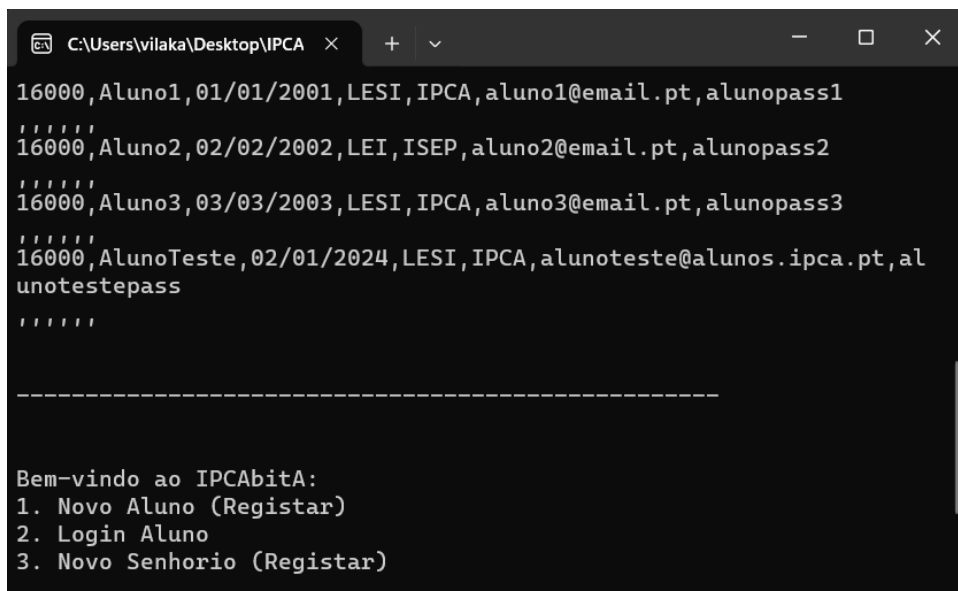


```
C:\Users\vilaka\Desktop\IPCA x + v
Login bem-sucedido! Bem-vindo

-----

1 - Ver Alunos
2 - Ver Senhórios
3 - Ver Quartos
4 - Ver Serviços
5 - Eliminar Aluno
6 - Eliminar Senhório
7 - Eliminar Quarto
8 - Eliminar Serviço
0 - Logout
Escolha uma opção:
```

### 2.10.18. Testes – Menu Login Admin (Ver Alunos)



```
C:\Users\vilaka\Desktop\IPCA x + v
16000,Aluno1,01/01/2001,LESI,IPCA,aluno1@email.pt,alunopass1
''''''
16000,Aluno2,02/02/2002,LEI,ISEP,aluno2@email.pt,alunopass2
''''''
16000,Aluno3,03/03/2003,LESI,IPCA,aluno3@email.pt,alunopass3
''''''
16000,AlunoTeste,02/01/2024,LESI,IPCA,alunoteste@alunos.ipca.pt,al
unotestepass
''''''

-----

Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhório (Registar)
```

### 2.10.19. Testes – Menu Login Admin (Ver Senhores)

```
C:\Users\vilaka\Desktop\IPCA x + v
Senhorio1,01/01/2001,senhorio1@email.pt,senhorio1pass1
'''
SenhorioTeste,02/01/2024,senhorioteste@email.pt,senhorioteste
'''

-----

Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
4. Login Senhorio
5. Login Admin
6. Refresh Menu
```

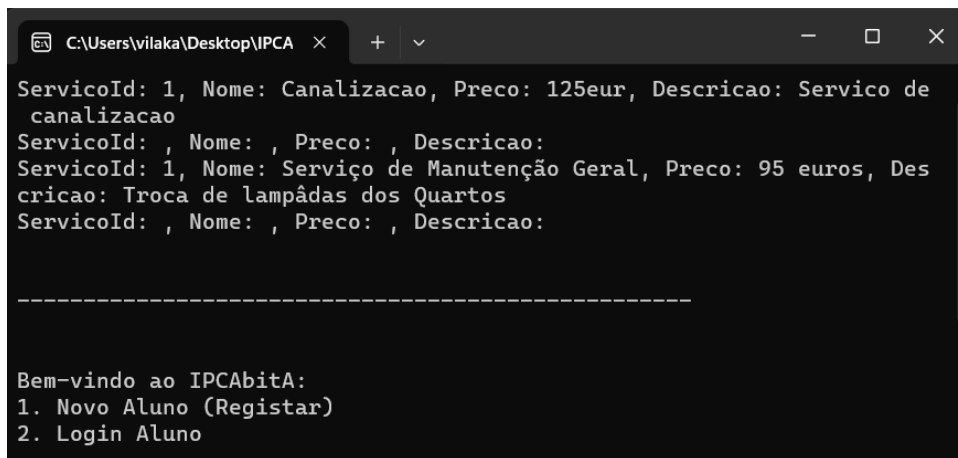
### 2.10.20. Testes – Menu Login Admin (Ver Quartos)

```
C:\Users\vilaka\Desktop\IPCA x + v
QuartoId: 1, Numero: 1, Capacidade: 2, Descricao: Cama de Casal
QuartoId: , Numero: , Capacidade: , Descricao:
QuartoId: 1, Numero: 123, Capacidade: 1, Descricao: Cama Solteiro
QuartoId: , Numero: , Capacidade: , Descricao:
QuartoId: 1, Numero: 999, Capacidade: 2, Descricao: Quarto com 1 c
ama solteiro
QuartoId: , Numero: , Capacidade: , Descricao:

-----

Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registar)
2. Login Aluno
3. Novo Senhorio (Registar)
```

### 2.10.21. Testes – Menu Login Admin (Ver Serviços)



The screenshot shows a web browser window with the address bar displaying 'C:\Users\vilaka\Desktop\IPCA'. The main content area displays a list of services in a monospaced font. The first service is 'Canalizacao' with a price of 125eur. The second service is 'Serviço de Manutenção Geral' with a price of 95 euros. The third service is 'Troca de lâmpadas dos Quartos'. Below the list, there is a horizontal dashed line. At the bottom of the page, there is a welcome message 'Bem-vindo ao IPCAbitA:' followed by a numbered list: '1. Novo Aluno (Registrar)' and '2. Login Aluno'.

```
ServicoId: 1, Nome: Canalizacao, Preco: 125eur, Descricao: Servico de
canalizacao
ServicoId: , Nome: , Preco: , Descricao:
ServicoId: 1, Nome: Serviço de Manutenção Geral, Preco: 95 euros, Des
cricao: Troca de lâmpadas dos Quartos
ServicoId: , Nome: , Preco: , Descricao:

-----

Bem-vindo ao IPCAbitA:
1. Novo Aluno (Registrar)
2. Login Aluno
```

## 3. Estruturas de Dados

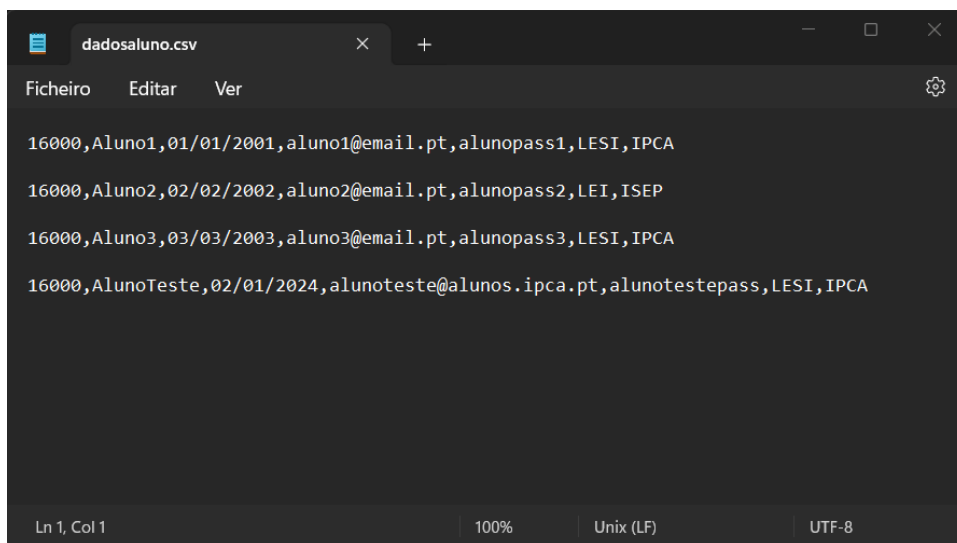
### 3.1. Listas

Como representado em cima nas Classes, iremos usar Listas como um dos tipos de estruturas de dados para organização e acessibilidade de dados necessários para o bom funcionamento da plataforma.

```
List<Aluno> listaAlunos = new List<Aluno>();
List<Senhorio> listaSenhorios = new List<Senhorio>();
List<Admin> listaAdmins = new List<Admin>();
```

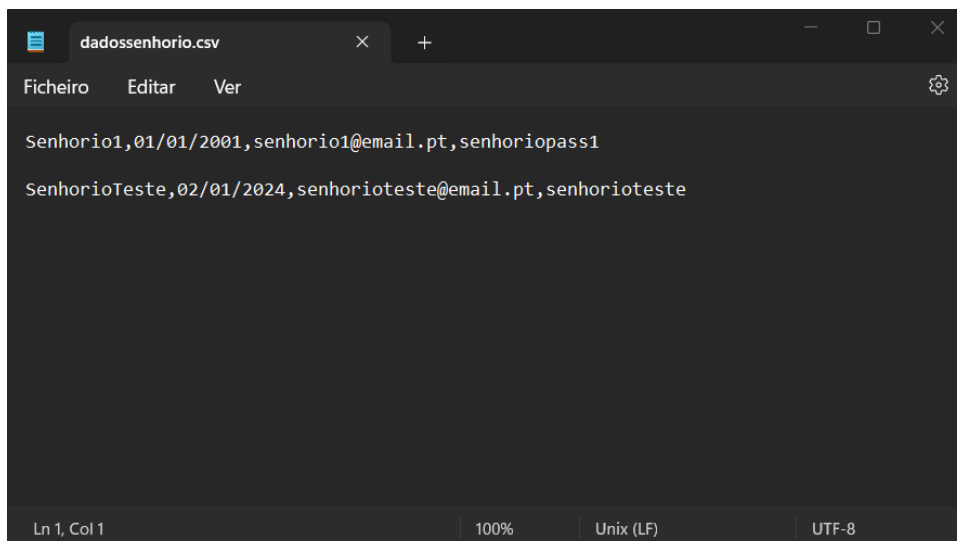
### 3.2. Ficheiros CSV

Outro tipo de estrutura de dados utilizada seriam os ficheiros CSV para testes de armazenamento e carregamento de dados.



A screenshot of a code editor window titled 'dadosaluno.csv'. The editor has a dark theme and a menu bar with 'Ficheiro', 'Editar', and 'Ver'. The main area contains four lines of CSV data. The status bar at the bottom shows 'Ln 1, Col 1', '100%', 'Unix (LF)', and 'UTF-8'.

```
16000,Aluno1,01/01/2001,aluno1@email.pt,alunopass1,LESI,IPCA
16000,Aluno2,02/02/2002,aluno2@email.pt,alunopass2,LEI,ISEP
16000,Aluno3,03/03/2003,aluno3@email.pt,alunopass3,LESI,IPCA
16000,AlunoTeste,02/01/2024,alunoteste@alunos.ipca.pt,alunotestepass,LESI,IPCA
```



A screenshot of a code editor window titled 'dadosenhorio.csv'. The editor has a dark theme and a menu bar with 'Ficheiro', 'Editar', and 'Ver'. The main area contains two lines of CSV data. The status bar at the bottom shows 'Ln 1, Col 1', '100%', 'Unix (LF)', and 'UTF-8'.

```
Senhorio1,01/01/2001,senhorio1@email.pt,senhoriopass1
SenhorioTeste,02/01/2024,senhorioteste@email.pt,senhorioteste
```

```
dadosadmin.csv
Ficheiro  Editar  Ver
Admin1,01/01/2001,admin1@email.pt,adminpass1
Ln 1, Col 1  100%  Windows (CRLF)  UTF-8
```

```
dadosquarto.csv
Ficheiro  Editar  Ver
1,1,2,Cama de Casal, 1 casa de banho, 1 lcd
1,123,1,Cama Solteiro, 1 casa de banho, 1 LCD
1,999,2,Quarto com 1 cama solteiro, 1 casa de banho e 1 Plasma LG de 43 polegadas.
Ln 1, Col 1  100%  Unix (LF)  UTF-8
```

```
dadosservico.csv
Ficheiro  Editar  Ver
1,Canalizacao,125eur,Service de canalizacao
1,Serviço de Manutenção Geral,95 euros,Troca de lâmpadas dos Quartos, de halógeno para LED.
Ln 1, Col 1  100%  Unix (LF)  UTF-8
```

## 4. Conclusão

Em resumo, este projeto foi desenvolvido para POO mas dá continuidade aos processos de negócio e modelação de software das disciplinas de PES e AMS, tendo sido desenvolvidos além plataforma em C#, os requisitos e funcionalidades da mesma.

O projeto na sua totalidade seria composto por um menu de Logins, validação dos mesmos, apresentação do menu correspondente à entidade em causa, funcionalidades de acordo com a entidade, etc.

Nesta última entrega, achamos que o projeto final representa bastante do nosso próprio trabalho ao longo do semestre, apesar de também acharmos que podia estar melhor em termos de teste de entrada de dados (trycatch), mais documentação sobre os métodos e/ou comentários sobre os mesmos, mais detalhe na representação da informação no ecrã do utilizador, etc. Iremos continuar a melhorar o projeto até à defesa, mesmo que essas mudanças já não entrem para a avaliação final.