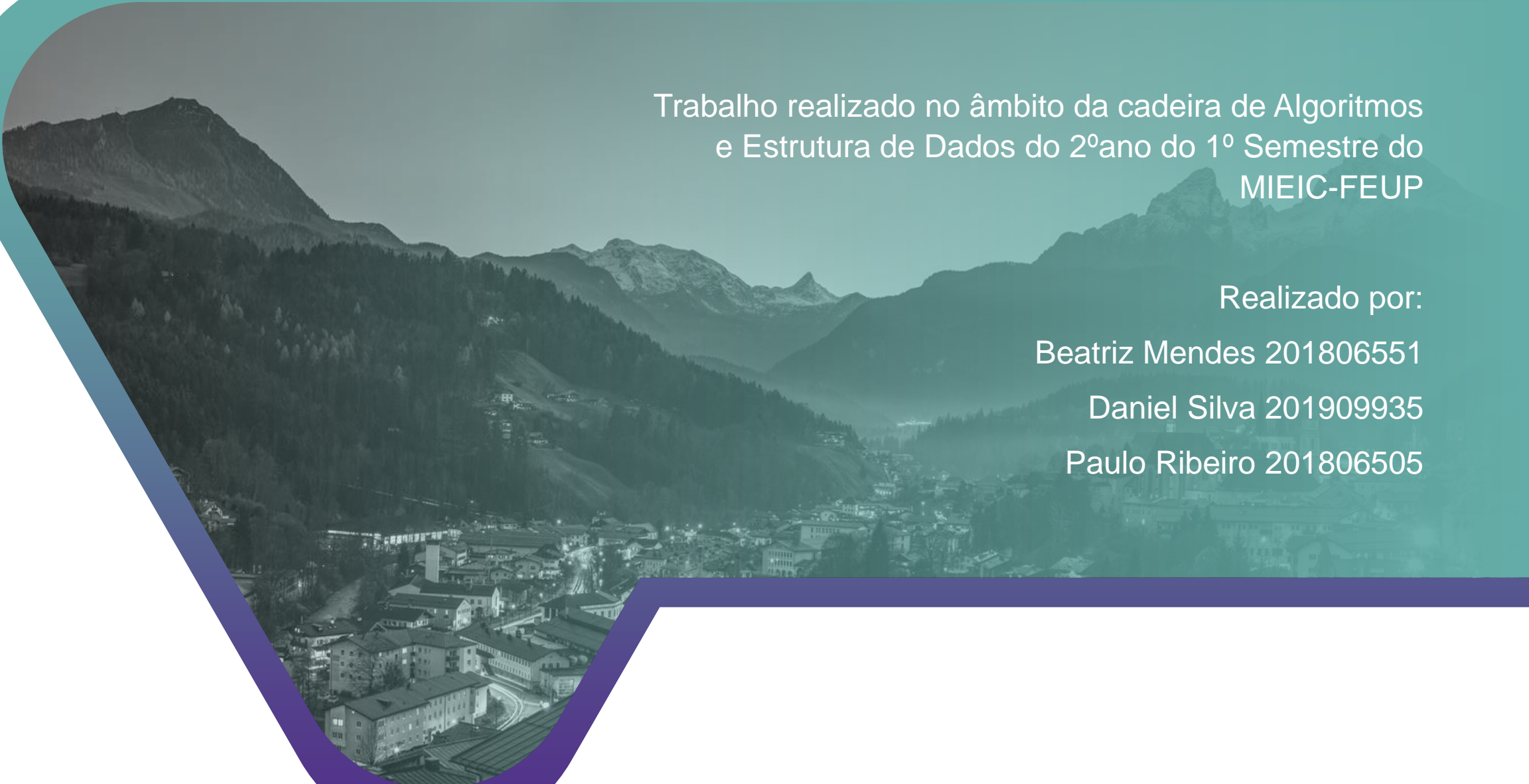




CONDOMI
X

TEMA 6 – GESTÃO DE CONDOMÍNIOS



Trabalho realizado no âmbito da cadeira de Algoritmos
e Estrutura de Dados do 2ºano do 1º Semestre do
MIEIC-FEUP

Realizado por:

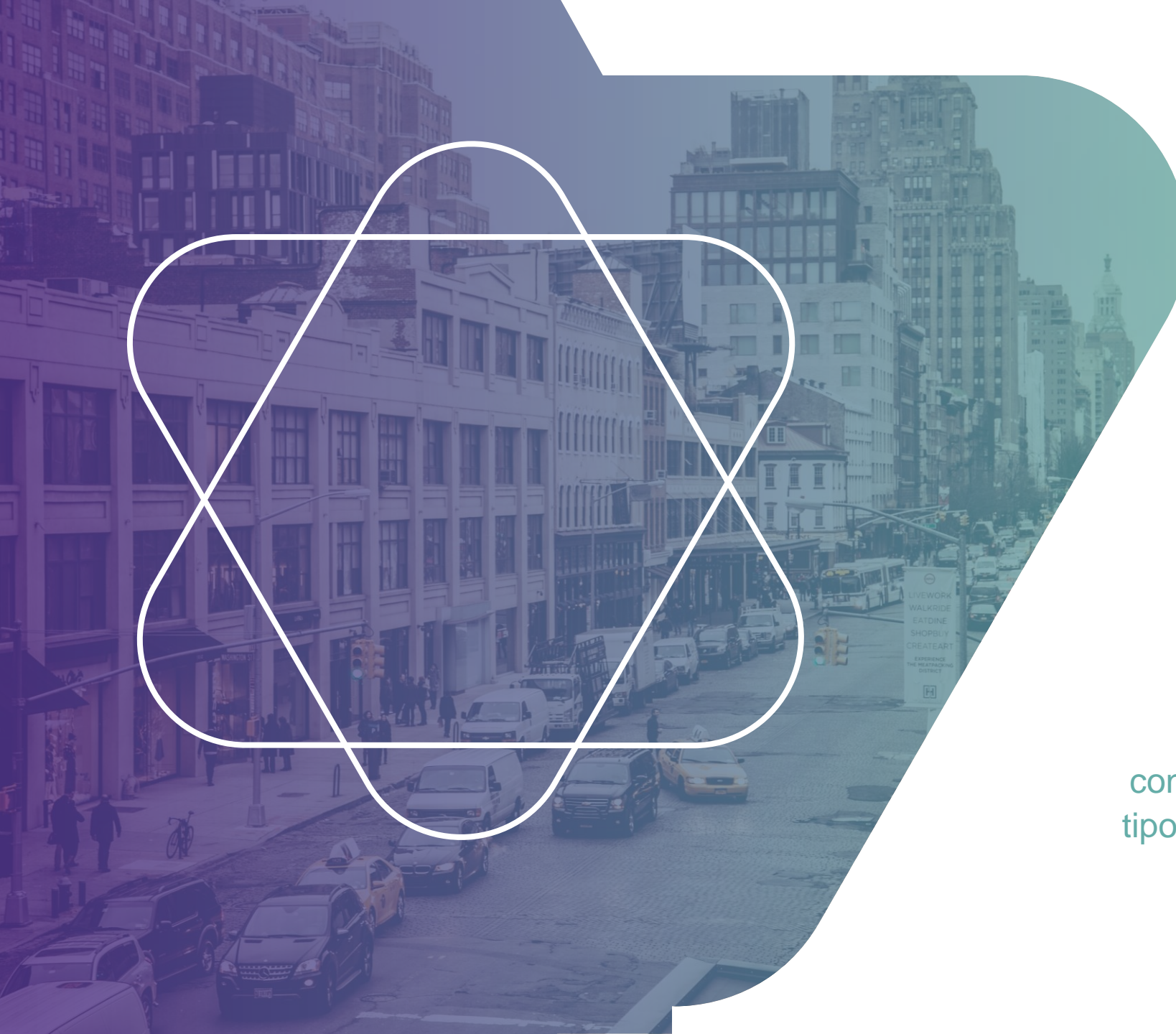
Beatriz Mendes 201806551

Daniel Silva 201909935

Paulo Ribeiro 201806505



Descrição do Problema e Solução



PROBLEMA

Criar um programa capaz de gerir um condomínio com habitações de diferentes tipos, informação dos condóminos e ainda serviços disponíveis a cada um destes condóminos



SOLUÇÃO

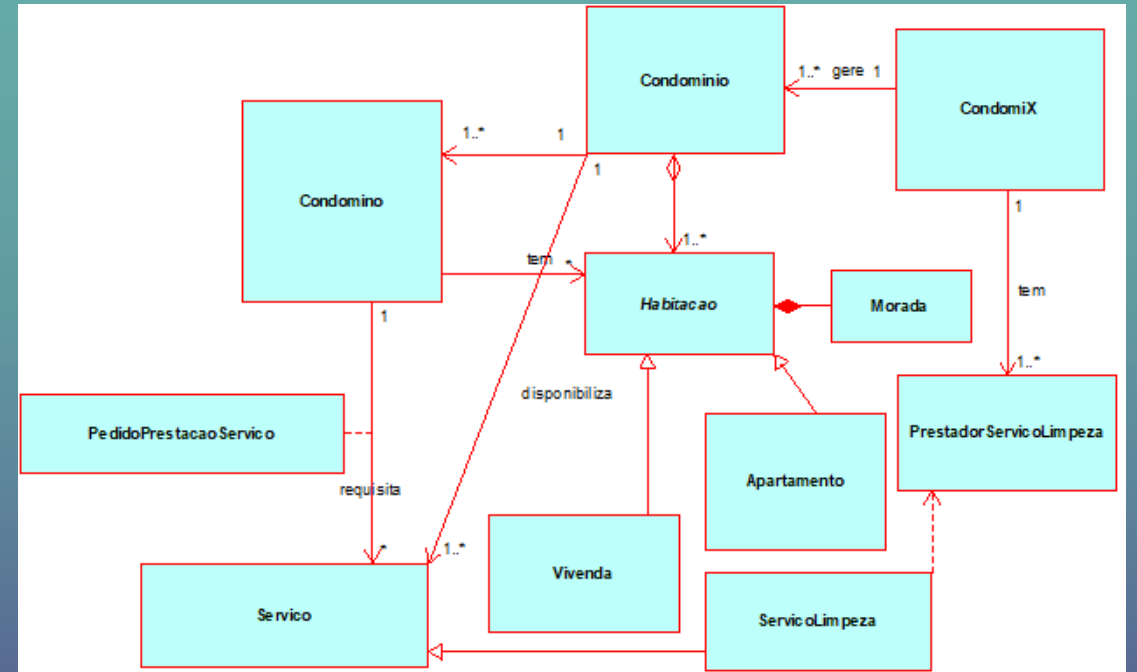
- Desenvolver uma aplicação que utilize:
 - Classes adequadas para representação das entidades envolvidas
 - Conceitos de Herança e Polimorfismo
 - Escrita e Leitura de Ficheiros
 - Exceções
 - Algoritmos de Pesquisa e Ordenação

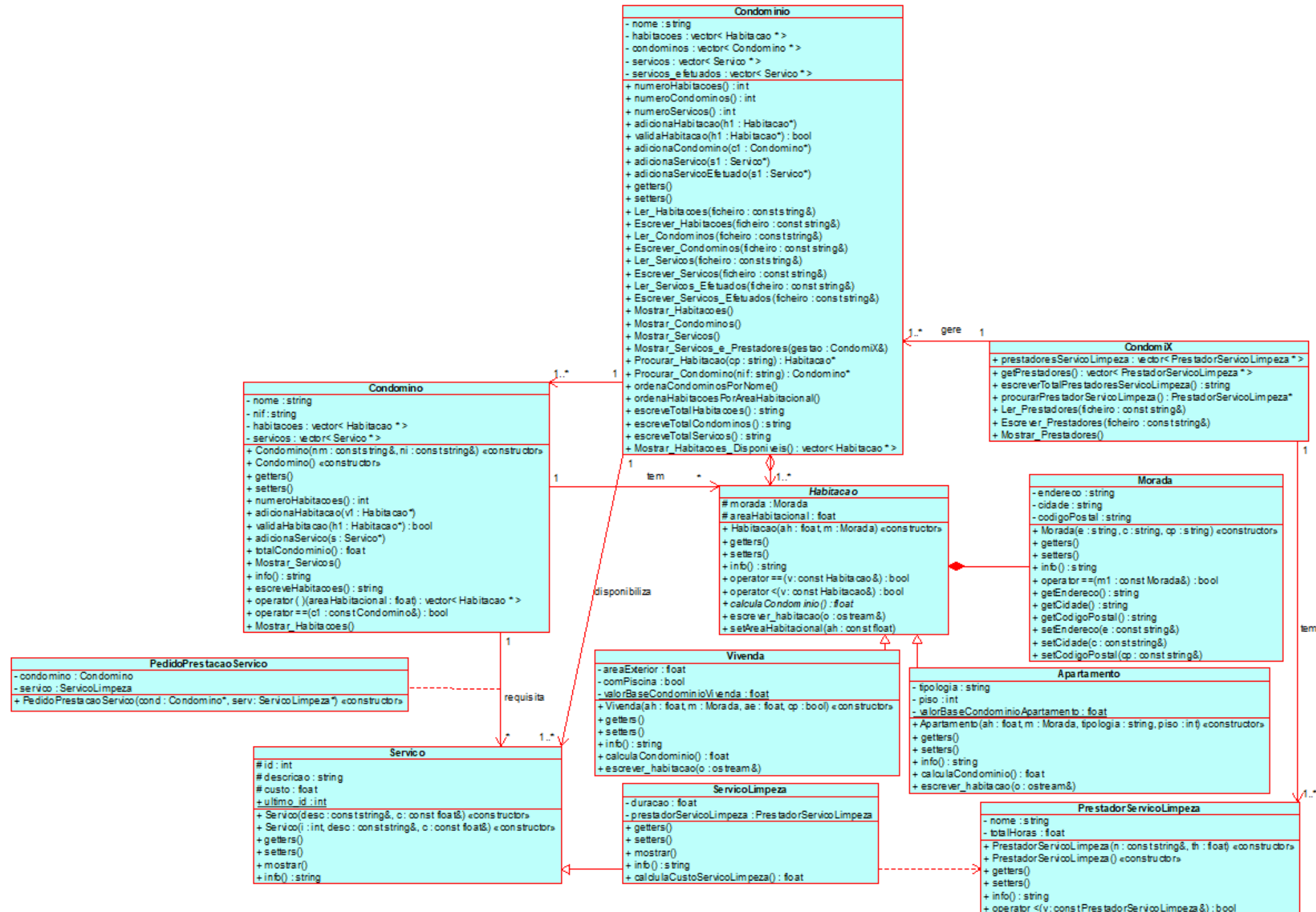


ALGORITMOS RELEVANTES

- Sequential Search
- Insertion Sort
- Sort (STL)
- Algoritmos de Escrita e Leitura de Ficheiros
- Etc.

Diagrama de Classes





Estrutura de Ficheiros (Ex: Habitacoes.txt)

Rua Agra Nova
Povoa de Varzim
4490-036
300
Vivenda
100
0

Rua da FEUP
Porto
4490-657
234
Apartamento
qualquer tipo
1

Rua Nova de Teste
Lisboa
4490-333
600
Apartamento
t4
3

Habitacoes.txt

```
void Condominio::Ler_Habitacoes(const string &ficheiro)
{
    habitacoes.clear();
    ifstream ler;
    string end ,cid ,cp, tipo_habitacao, separacao;
    float area_habitacional;
    try{
        ler.open(ficheiro);
    }
    catch (Erro_Ler(&ficheiro)){
        cout << "Erro a Ler Habitacoes\n";
        exit(1);
    }
    do{
        getline(ler, end);
        getline(ler, cid);
        getline(ler, cp);
        Morada *morada = new Morada(end,cid,cp);
        ler >> area_habitacional;
        ler.ignore(1000, '\n');
        getline(ler, tipo_habitacao);
        if(tipo_habitacao == "Vivenda")
        {
            float area_exterior;
            bool com_piscina;
            ler >> area_exterior;
            ler.ignore(1000, '\n');
            ler >> com_piscina;
            ler.ignore(1000, '\n');
            getline(ler, separacao);
            Habitacao *vivenda = new Vivenda(area_habitacional,*morada,
            area_exterior,com_piscina);
            habitacoes.push_back(vivenda);
        }
        else
        {
            string tipologia;
            int piso;
            getline(ler, tipologia);
            ler >> piso;
            ler.ignore(1000, '\n');
            getline(ler, separacao);
            Habitacao *apartamento = new Apartamento(area_habitacional,
            *morada, tipologia, piso);
            habitacoes.push_back(apartamento);
        }
    } while (!ler.eof());
    ler.close();
}
```

Leitura de Habitacoes.txt

```
void Condominio::Escrever_Habitacoes(const string
&ficheiro)
{
    ofstream escrever;
    vector<Habitacao *>::const_iterator it;
    try{
        escrever.open(ficheiro);
    }
    catch (Erro_Escrita(&ficheiro)){
        cout << "Erro a Escrever Habitacoes\n";
        exit(1);
    }
    for(it = habitacoes.begin(); it != habitacoes.end();
    it++)
    {
        if(it == habitacoes.begin())
            (*it)->escrever_habitacao(escrever);
        else
        {
            escrever << endl;
            (*it)->escrever_habitacao(escrever);
        }
    }
    escrever.close();
}
```

```
void Habitacao::escrever_habitacao(ostream &o) const
{
    o << morada.getEndereco() << endl <<
    morada.getCidade() << endl <<
    morada.getCodigoPostal() << endl;
    o << areaHabitacional << endl;
}
```

Escrita em Habitacoes.txt

Exemplo: Habitacao_Nao_Encontrada

[illegible]

TRATAMENTO DE EXCEÇÕES

```
class Erro_Escrita{
    string ficheiro;
public:
    Erro_Escrita(string f){ ficheiro = f;}
};

class Erro_Ler{
    string ficheiro;
public:
    Erro_Ler(string f){ ficheiro = f;}
};

class Habitacao_Nao_Encontrada{
    string cp;
public:
    Habitacao_Nao_Encontrada(string c) {cp = c;}
};

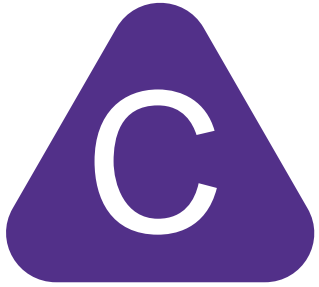
class Habitacao_Ja_Aquirida{
    string cp;
public:
    Habitacao_Ja_Aquirida(string c) {cp = c;}
};

class Condominio_Nao_Encontrado{
    string nif;
public:
    Condominio_Nao_Encontrado(string n) {nif = n;}
};

class CP_Ja_Existente{
    string cp;
public:
    CP_Ja_Existente(string c) {cp = c;}
};

class NIF_Ja_Existente{
    string nif;
public:
    NIF_Ja_Existente(string n) {nif = n;}
};
```

Funcionalidades Implementadas



Create



Read



Update



Delete

```
void Condominio::Ler_Habitacoes(const string &ficheiro)
```

```
void Condominio::Escrever_Habitacoes(const string &ficheiro)
```

```
void Condominio::Ler_Condominos(const string &ficheiro)
```

```
void Condominio::Escrever_Condominos(const string &ficheiro)
```

```
void Condominio::Ler_Servicos(const string &ficheiro)
```

```
void Condominio::Escrever_Servicos(const string &ficheiro)
```

```
void Condominio::Ler_Servicos_Efetutados(const string  
&ficheiro)
```

```
void Condominio::Escrever_Servicos_Efetutados(const string  
&ficheiro)
```

```
void Adicionar_Habitacao(Condominio &condominio);
```

```
void Editar_Habitacao(Condominio &condominio);
```

```
void Eliminar_Habitacao(Condominio &condominio);
```

```
void Adicionar_Condomino(Condominio &condominio);
```

```
void Editar_Condomino(Condominio &condominio);
```

```
void Eliminar_Condomino(Condominio &condominio);
```

```
void Adicionar_Servico(Condominio &condominio);
```

```
void Editar_Servico(Condominio &condominio);
```

```
void Eliminar_Servico(Condominio &condominio);
```

Completas

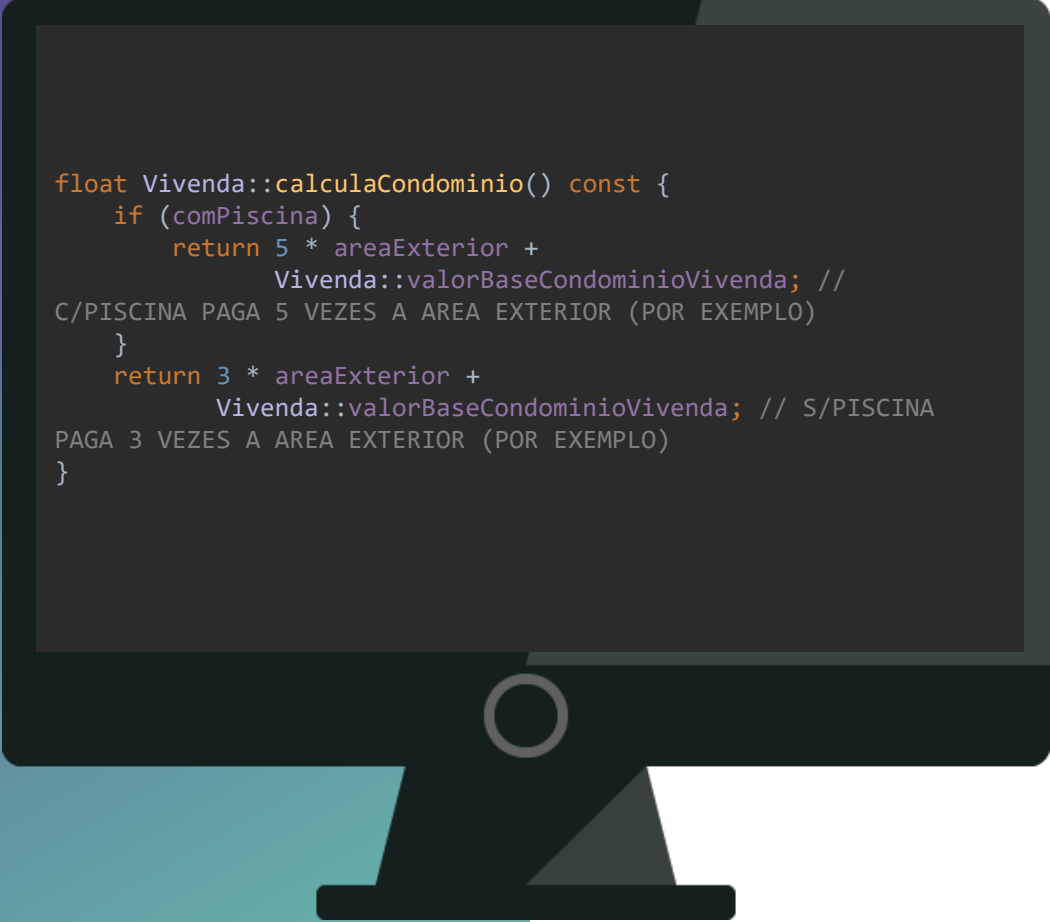
Listagem

```
void Visualizar_Todas_Habitacoes(Condominio &condominio);  
void Visualizar_Todos_Condominios (Condominio &condominio);  
void Visualizar_Servicos(Condominio &condominio);  
void Visualizar_Prestadores(Condominio &condominio,  
Condominio &gestao);
```

```
void Visualizar_Todas_Habitacoes (Condominio &condominio)
{
    condominio.Ler_Habitacoes("Habitacoes.txt");
    system("CLS");
    Logotipo();
    cout << "\n\n\t\t      Visualizacao de Todas as  

Habitacoes (Ordem Decrescente de Area Habitacional) \n\n";
    condominio.ordernaHabitacoesPorAreaHabitacional();
    cout << condominio.escreveTotalHabitacoes();
    char sair = Sair_Programa();
    if (sair == 'N' || sair == 'n')
        Menu_Principal();
}
```


Funcionalidade Destaque



```
float Vivenda::calculaCondominio() const {  
    if (comPiscina) {  
        return 5 * areaExterior +  
            Vivenda::valorBaseCondominioVivenda; //  
C/PISCINA PAGA 5 VEZES A AREA EXTERIOR (POR EXEMPLO)  
    }  
    return 3 * areaExterior +  
        Vivenda::valorBaseCondominioVivenda; // S/PISCINA  
PAGA 3 VEZES A AREA EXTERIOR (POR EXEMPLO)  
}
```

```
class Habitacao {  
    (...)  
public:  
    (...)  
    virtual float calculaCondominio()  
    const = 0;  
};
```

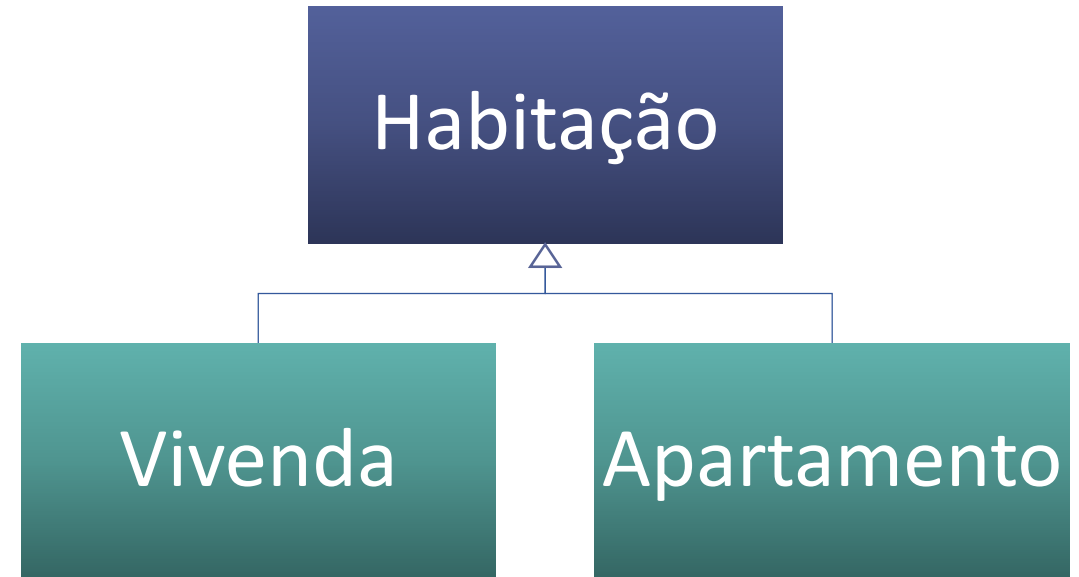
```
class Vivenda: public Habitacao {  
    (...)  
public:  
    (...)  
    float calculaCondominio() const;  
};  
  
class Apartamento: public Habitacao {  
    (...)  
public:  
    (...)  
    float calculaCondominio() const;  
};
```

Utilização de uma função virtual pura para facilitar o cálculo do valor mensal de condomínio, dependendo do tipo de habitação e das suas características

Principais Dificuldades

Casting

```
void Editar_Habitacao(Condominio &condominio)
{
    (...)
    for(int i = 0; i < condominio.numeroHabitacoes(); i++)
    {
        if(i == (id-1))
        {
            (...)
            cout << "\nTipo da Habitacao: \n";
            cout << "\n[1] Vivenda";
            cout << "\n[2] Apartamento";
            do {
                cout << "\n\nOpcao: ";
                cin >> tp;
                cin.ignore(1000, '\n');
            } while (tp != 1 && tp != 2);
            if (tp == 1) tipo = "Vivenda";
            if (tp == 2) tipo = "Apartamento";
            if(tipo == "Vivenda")
            {
                (...)
                ((Vivenda *) vetor[i])->setAreaExterior(areaExterior);
                ((Vivenda *) vetor[i])->setComPiscina(comPiscina);
            }
            else if (tipo == "Apartamento")
            {
                (...)
                ((Apartamento *) vetor[i])->setTipologia(tipologia);
                ((Apartamento *) vetor[i])->setPiso(piso);
            }
            break;
        }
    }
    (...)
}
```



Exemplos de Execução

Exemplo1: Pesquisa de uma Habitação pela Morada

```
CondominX
Menu Principal

Utilizador
[1] Visualizar Informacoes
[2] Adquirir Habitacao
[3] Remover Habitacao
[4] Requisitar Servico
[5] Terminar Servico

Administrador
[6] Gestao de Habitacoes
[7] Gestao de Condominos
[8] Gestao de Servicos
[9] Dados da Empresa

[0] Sair do Programa

Opcao: _
```

Menu Principal



```
CondominX
Menu de Visualizacao

[1] Visualizar Habitacoes
[2] Visualizar Condominos
[3] Visualizar Servicos
[4] Visualizar Prestadores
[5] Cancelar

Opcao: _
```

Menu de Visualização



```
CondominX
Menu de Visualizacao de Habitacoes

[1] Visualizar Todas as Habitacoes (Ordem Decrescente de Area Habitacional)
[2] Visualizar Habitacao pela Morada
[3] Cancelar

Opcao: _
```

Menu de Visualização de Habitações



```
CondominX
Visualizacao de uma Habitacao pela Morada

Codigo-postal da Habitacao a procurar: 4490-036
-----
Tipo de Habitacao: Vivenda
Endereco: Rua Agra Nova
Cidade: Povia de Varzim
Codigo Postal: 4490-036

Area Habitacional: 300
Area Exterior: 100
Piscina: Nao

Sair do Programa? (S/N)
R: _
```

Habitação Encontrada: 4490-036



```
CondominX
Visualizacao de uma Habitacao pela Morada

Codigo-postal da Habitacao a procurar: _
```

Menu de Procura de uma habitação pela Morada

Exemplos de Execução

Exemplo2: Visualizar os Serviços Existentes no Condomínio e os Seus Prestadores

```
CondominX

Menu Principal

Utilizador

[1] Visualizar Informacoes
[2] Adquirir Habitacao
[3] Remover Habitacao
[4] Requisitar Servico
[5] Terminar Servico

Administrador

[6] Gestao de Habitacoes
[7] Gestao de Condominos
[8] Gestao de Servicos
[9] Dados da Empresa

[0] Sair do Programa

Opcao: _
```

Menu Principal

```
CondominX

Dados da Empresa

-----
Numero de Habitacoes: 10      Numero de Condominos: 9      Numero de Servicos: 5
Numero de Habitacoes Vendidas: 7      Numero de Servicos Efetuados: 7      Valor Mensal Obtido: 18710
-----

Visualizar:
[1] Servicos Efetuados
[2] Servicos Existentes e os Seus Prestadores
[3] Cancelar

Opcao: _
```

Menu de Visualização dos Dados da Empresa

```
CondominX

Servicos Existentes e Seus Prestadores

[1] Servico de Limpeza
    Nome: Sr Antonio Horas: 470
    Nome: Sr Benedito Horas: 456493
    Nome: Sr Carlos Horas: 62
    Nome: Sr Dinis Horas: 58
    Nome: Sr Eugenio Horas: 65
[2] Servico de Canalizacao
[3] Servico de Pintura
[4] Servico de Seguranca
[5] Servico de Apoio

Sair do Programa? (S/N)

R: _
```

Serviços Existentes e os seus Prestadores