INTRODUÇÃO

O programa foi feito para auxiliar o sindico do prédio em calcular as despesas gastas pelo prédio no mês e mostrar para o sindico quanto cada morador devera pagar no mês.

Ele vai cadastrar o condominio, o morador, as despesas, permite exibir para o usuario as depesas de cada morador juntamente com os dados do mesmo, consultar arquivos calculados anteriormente.

OBJETIVOS

Gerais

Calcular as despesas para o usuario, exibir na tela o calculo e salvar o mesmo em arquivo.

Especifico

O programa esta dividido em partes, “Cadastro”, “Calculo”, “Consulta”, onde vamos mostrar cada uma agora:

Cadastro

Na parte de cadastro será possível cadastrar o condomínio com dados de nome, endereço (Rua, Bairro e Cidade), numero de prédios.

Haverá o cadastro do prédio, com nome do prédio, numero e numero de apartamentos.

No cadastro de morador será necessário informar o nome do morador, telefone, numero de dependentes, CPF, endereço (prédio, num_ap).

No cadastro de contas, o usuário vai cadastras as contas fixas mensais como agua e energia, um campo para outras despesas com um loop para poder cadastrar diversas outras despesas, data do vencimento das despesas.

Calculo

No calculo do programa, será somado todas as despesas e dividido pelo numero de moradores.

Exibir para o usuário (sindico) um relatório com um nome do morador, o numero do apartamento e o valor do condomínio a ser pago.

Consulta

Na consulta será permitido consultar as contas cadastradas, os moradores, e os prédios, será possível também consultar um relatório de algum mês anterior.

METODOLOGIA

REQUISITOS TÉCNICOS DO PRODUTO

Entre os requisitos estão segurança onde será criado uma área de login e senha para o usuário, desempenho do software, ele deve ser eficiente.

O programa devera ter a possibilidade do usuário cadastrar e consultar, e somente o administrador excluir.

Ele deve possuir um help para usuários leigos.

RESULTADOS ESPERADOS

Esperamos conseguir agilizar e organizar a forma de cobrança do condomínio, trazendo mais facilidade com os relatórios por morador e por mês. E com isso o Sindico do condomínio consegue fazer todo seu trabalho que poderia demorar dias em apenas alguns minutos ou horas. O sistema será simples de usar para que uma pessoa que pouco conhecimento em informática tenha capacidade de dominar com clareza o programa.

CRONOGRAMA

Atividade	Agosto	Setembro	Outubro	Novembro
Pré-Projeto	X			
Fase 1		X		
Fase 2			X	
Fase 3				X

2 FASE

CÓDIGOS FONTE

Estruturas

```
public struct tipo_morador
        {
            public int cod;
            public string nome;
            public string tel;
            public int num_dependentes;
            public string cpf;
            public string rua;
            public int num;
            public int complemento;
            public DateTime inicio_moradia;

        }
        public struct tipo_condominio
        {
            public string nome;
            public int num_apartamentos, cod;
        }
        public struct tipo_despesa
        {
            public int cod;
            public DateTime data_pagamento;
            public string descricao;
            public double valor;

        }
        public struct tipo_controle_pagamento
        {
            public int cod;
            public int cod_morador;
            public DateTime data_referencia;
            public DateTime data_pagamento;
            public bool pago;

        }
```

Foram criadas as estruturas que vão ser usadas no programa.

Arquivos

```
        static string localDados = @"C:/ProjetoIntegrador/Prog_Cond/";
        static string arquivoDadosCondominio = @"Condominio.txt";
        static string arquivoDadosMorador = @"Morador.txt";
        static string arquivoDadosDespesas = @"Despesas.txt";
        static string arquivoDadosPagamentos = @"Pagamento.txt";
```

Foram definidos em variáveis os caminhos para salvar os arquivos, isso facilita depois quando precisamos chamar os arquivos.

```
DirectoryInfo dirInfo = new DirectoryInfo(localDados);
            if (!dirInfo.Exists)
            {
```

```
                dirInfo.Create();
            }

            if (!File.Exists(localDados + arquivoDadosMorador))
            {
                File.Create(localDados + arquivoDadosMorador);
            }

            if (!File.Exists(localDados + arquivoDadosDespesas))
            {
                File.Create(localDados + arquivoDadosDespesas);
            }

            if (!File.Exists(localDados + arquivoDadosPagamentos))
            {
                File.Create(localDados + arquivoDadosPagamentos);
            }

            if (!File.Exists(localDados + arquivoDadosCondominio))
            {
                File.Create(localDados + arquivoDadosCondominio);
            }
```

Nesta parte do código usamos o comando para criar os arquivos caso eles não existam.

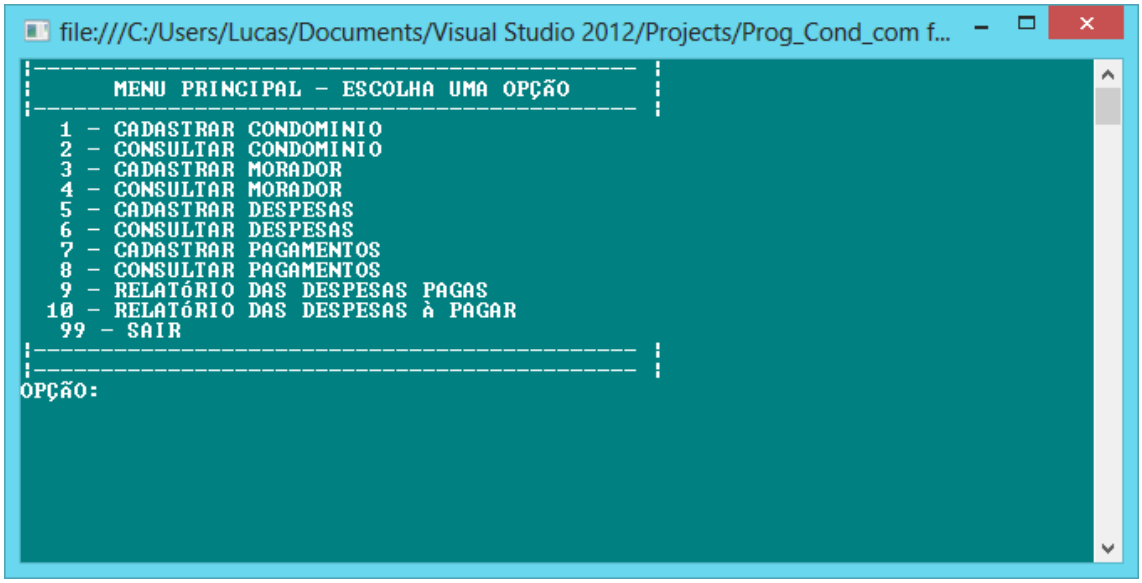
Menu

```
while (true)
            {
                try
                {
                    Console.Clear();
                    Console.WriteLine("|----------------------------------------
---- |");
                    Console.WriteLine("|      MENU PRINCIPAL - ESCOLHA UMA OPÇÃO
|");
                    Console.WriteLine("|----------------------------------------
---- |");
                    Console.WriteLine("   1 - CADASTRAR CONDOMINIO");
                    Console.WriteLine("   2 - CONSULTAR CONDOMINIO");
                    Console.WriteLine("   3 - CADASTRAR MORADOR");
                    Console.WriteLine("   4 - CONSULTAR MORADOR");
                    Console.WriteLine("   5 - CADASTRAR DESPESAS");
                    Console.WriteLine("   6 - CONSULTAR DESPESAS");
                    Console.WriteLine("   7 - CADASTRAR PAGAMENTOS");
                    Console.WriteLine("   8 - CONSULTAR PAGAMENTOS");
                    Console.WriteLine("   9 - RELATÓRIO DAS DESPESAS PAGAS");
                    Console.WriteLine("  10 - RELATÓRIO DAS DESPESAS À PAGAR");
                    Console.WriteLine("   99 - SAIR");
                    Console.WriteLine("|----------------------------------------
---- |");
                    Console.WriteLine("|----------------------------------------
---- |");
                    Console.Write("OPÇÃO:");
                    int opcaoEscolhida = int.Parse(Console.ReadLine());

                    if (opcaoEscolhida == 1)
                    {
                        cadastro_condominio();
                    }
```

```
                    else if (opcaoEscolhida == 2)
                    {
                        consulta_condominio();
                    }
                    else if (opcaoEscolhida == 3)
                    {
                        cadastro_morador();
                    }
                   else if (opcaoEscolhida == 4)
                    {
                        consulta_morador();
                    }
                    else if (opcaoEscolhida == 5)
                    {
                        cadastro_despesa();
                    }
                    else if (opcaoEscolhida == 6)
                    {
                        consulta_despesa();
                    }
                  /*  else if (opcaoEscolhida == 7)
                    {
                        relatorio_despesas_pagas();
                    }
                    else if (opcaoEscolhida == 8)
                    {
                        relatorio_despesas_a_pagar();
                    }
                  */
                    else if (opcaoEscolhida == 99)
                    {
                        break;
                    }
                    else
                    {
                        Console.WriteLine("OPÇÃO INVÁLIDA!");
```

```
                    }
                    Console.WriteLine("|----------------------------------------
---- |");
                    Console.WriteLine("|  PRECIONE ALGUMA TELCA PARA VOLTAR AO
MENU   |");
                    Console.WriteLine("|----------------------------------------
---- |");
                    Console.ReadLine();
                }
                catch (Exception ex)
                {
                    Console.WriteLine("|----------------------------------------
---- |");
                    Console.WriteLine("ERRO:");
                    Console.WriteLine(ex.Message);
                    Console.WriteLine("|----------------------------------------
---- |");
                    Console.WriteLine("|  PRECIONE ALGUMA TELCA PARA VOLTAR AO
MENU   |");
                    Console.WriteLine("|----------------------------------------
---- |");
                    Console.ReadLine();
                }
            }
```

Este é nosso menu, onde ele vai pedir uma opção para usuario e com isso chama uma função para cada item do menu. São opções de cadastro, consulta e relatorios. Um item do menu fecha o programa, ele roda sempre até que o usario use a opçao para fechar o programa. Com o while(true), sempre que cada função for finalizada ele volta para o menu principal.



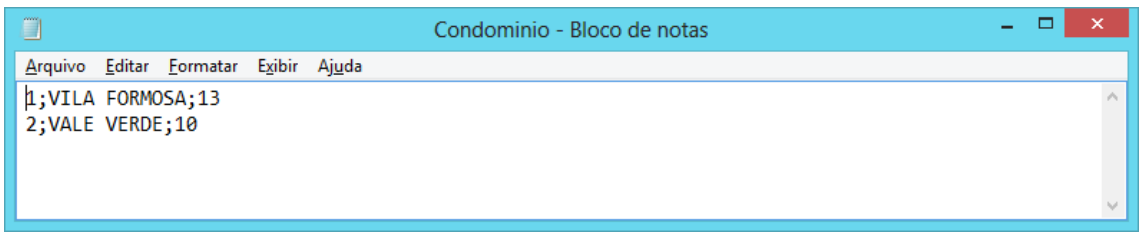
Funções

Função cadastro_condomino()

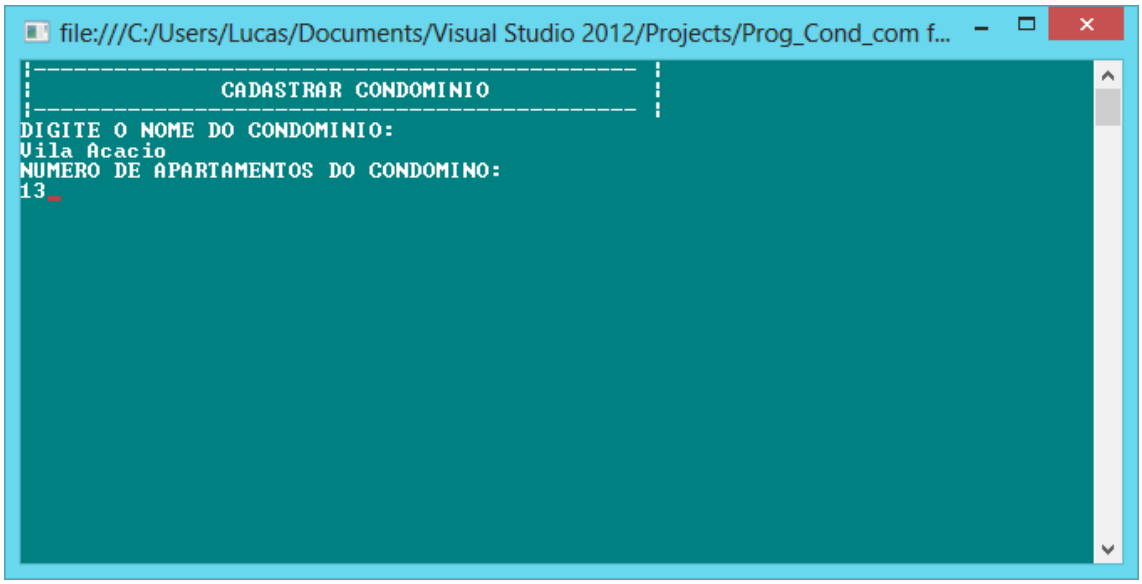
```
  static void cadastro_condominio()
        {

            tipo_condominio condominio;
            Console.WriteLine("|---------------------------------------------
|");
            Console.WriteLine("|              CADASTRAR CONDOMINIO
|");
            Console.WriteLine("|---------------------------------------------
|");
            Console.WriteLine("DIGITE O NOME DO CONDOMINIO:");
            condominio.nome = Console.ReadLine().ToUpper().Replace(';', ' ');
            Console.WriteLine("NUMERO DE APARTAMENTOS DO CONDOMINO:");
            condominio.num_apartamentos = int.Parse(Console.ReadLine());

            Console.Clear();


            StreamReader reader = new StreamReader(localDados +
arquivoDadosCondominio);


            int contaRegistro = 0;

            while (reader.ReadLine() != null)
```

```
            {
                contaRegistro++;
            }

            reader.Close();

            reader.Dispose();

            StreamWriter writer = File.AppendText(localDados +
arquivoDadosCondominio);

            condominio.cod = contaRegistro + 1;

            writer.WriteLine(condominio.cod + ";" + condominio.nome + ";" +
condominio.num_apartamentos );

            writer.Close();

            writer.Dispose();

            Console.WriteLine("|---------------------------------------------
|");
            Console.WriteLine("|              CODIGO DO CONDOMINIO: " +
condominio.cod + "              |");
            Console.WriteLine("|---------------------------------------------
|");
            Console.WriteLine("|                    SUCESSO
|");
            Console.WriteLine("|---------------------------------------------
|");
        }
```

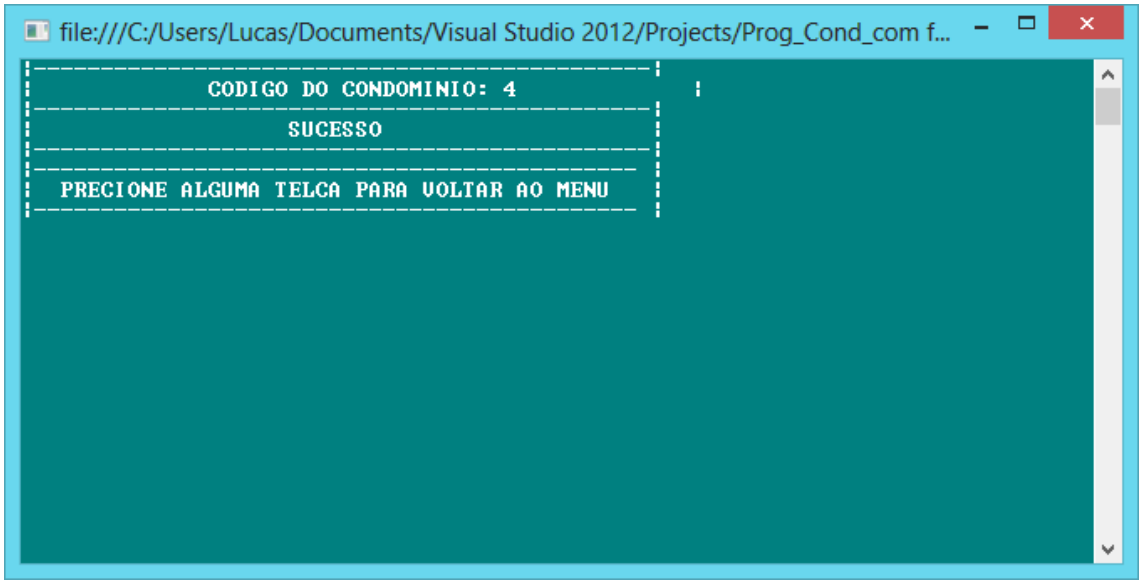
Esta é a função para cadastrar condominio, ela é a primeira opção do menu. Nela pedimos para que o usuario digite o nome e o numero de apartamentos no condominio. Depois salvamos ele no arquivo Condominio.txt separando por ponto e virgula cada item. A opçao de código do condominio é preenchida de forma automatica, para saber o novo código do condominio, o programa faz uma leitura nas linhas do arquivo Condominio.txt e de acordo com a quantidade de linhas ele gera o código. Por exemplo, se não houver nenhuma linha escrita, esse condominio vai levar o código ‘1’, e se já houver outros ‘4’ cadastrados, ele vai gerar o código ‘5’ e salvar os dados desse condominio na linha 5.



Logo abaixo é a tela de cadastro de condominio



Depois do usuario digitar os dados. Ele confirma e mostra o código para o usuario.



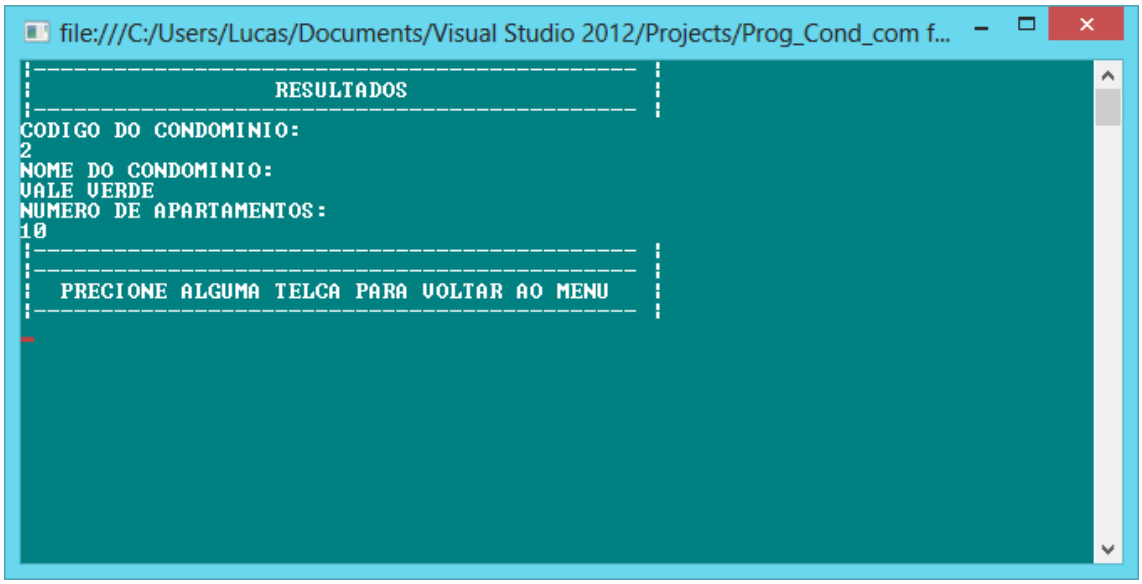
Função busca_condominio()

```
static void consulta_condominio()
        {

                Console.Clear();
                Console.WriteLine("|-----------------------------------------
|");
                Console.WriteLine("|                BUSCAR CONDOMINIO
|");
                Console.WriteLine("|-----------------------------------------
|");
                Console.WriteLine("   DIGITE O NOME DO CONDOMINIO:");
                string termoBusca = Console.ReadLine().Replace(';', '
').ToUpper();


                Console.Clear();
```

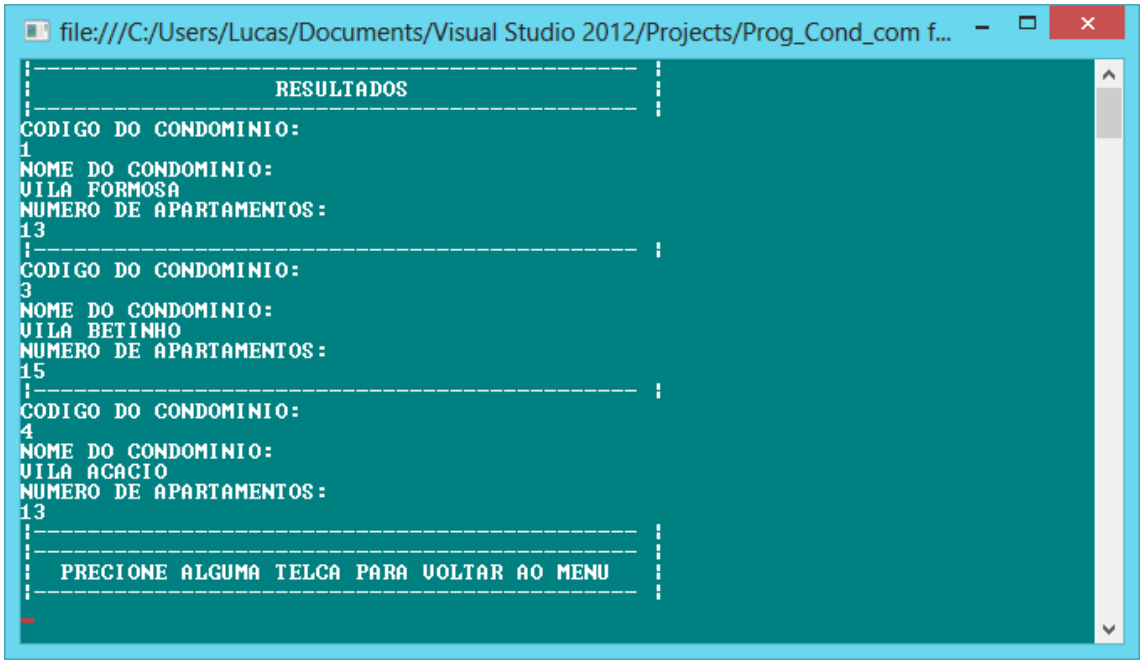
```
                Console.WriteLine("|-----------------------------------------------
|");
                Console.WriteLine("|                    RESULTADOS
|");
                Console.WriteLine("|-----------------------------------------------
|");

                StreamReader reader = new StreamReader(localDados +
arquivoDadosCondominio);

                string resultado;
                while ((resultado = reader.ReadLine()) != null)
                {

                    if (resultado.Contains(termoBusca))
                    {

                        string[] condominio_resultado = resultado.Split(';');


                        tipo_condominio condominio;
                        condominio.cod =
Convert.ToInt32(condominio_resultado[0]);
                        condominio.nome = condominio_resultado[1];
                        condominio.num_apartamentos=
int.Parse(condominio_resultado[2]);


                        Console.WriteLine("CODIGO DO CONDOMINIO:");
                        Console.WriteLine(condominio.cod);
                        Console.WriteLine("NOME DO CONDOMINIO:");
                        Console.WriteLine(condominio.nome);
                        Console.WriteLine("NUMERO DE APARTAMENTOS:");
                        Console.WriteLine(condominio.num_apartamentos);
                        Console.WriteLine("|-----------------------------------
-------- |");
                    }
                }

                reader.Close();

                reader.Dispose();
            }
```

Nesta função, pedimos ao usuário o nome do condomínio, buscamos no arquivo o termo que ele digitou, se o termo digitado estiver condigo em algum condomínio salvo ele vai exibir todos que contenha aquele termo. Ele faz a leitura da linha em que esta salvo cada condomínio, e usa o ponto e virgula para quebrar a linha em diversos vetores e salvando cada um em seu respectivo item para que possa mostrar de forma organizada para o usuário a busca.

Pesquisando no programa “Vale Verde”, vemos o cadastro feito anteriormente de forma organizada.



Se no termo de busca usarmos algo que pode conter em mais de um condomínio, ele mostra todos que contem aquele termo. Neste caso usamos o termo “Vila”;



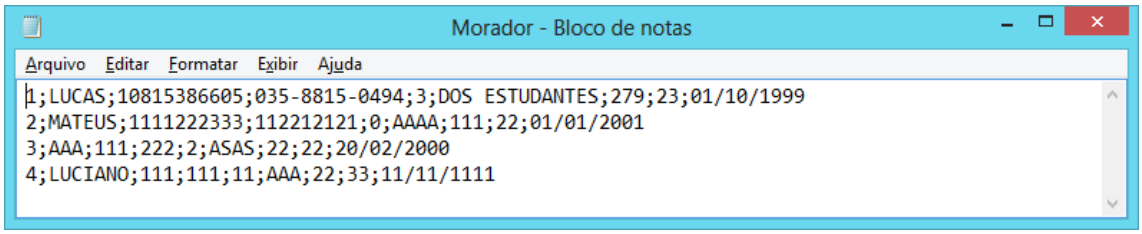
Função cadastro_morador()

```
static void cadastro_morador()
        {
            Console.Clear();


            tipo_morador morador;
            Console.WriteLine("|-----------------------------------------
|");
            Console.WriteLine("|              CADASTRAR MORADOR
|");
```

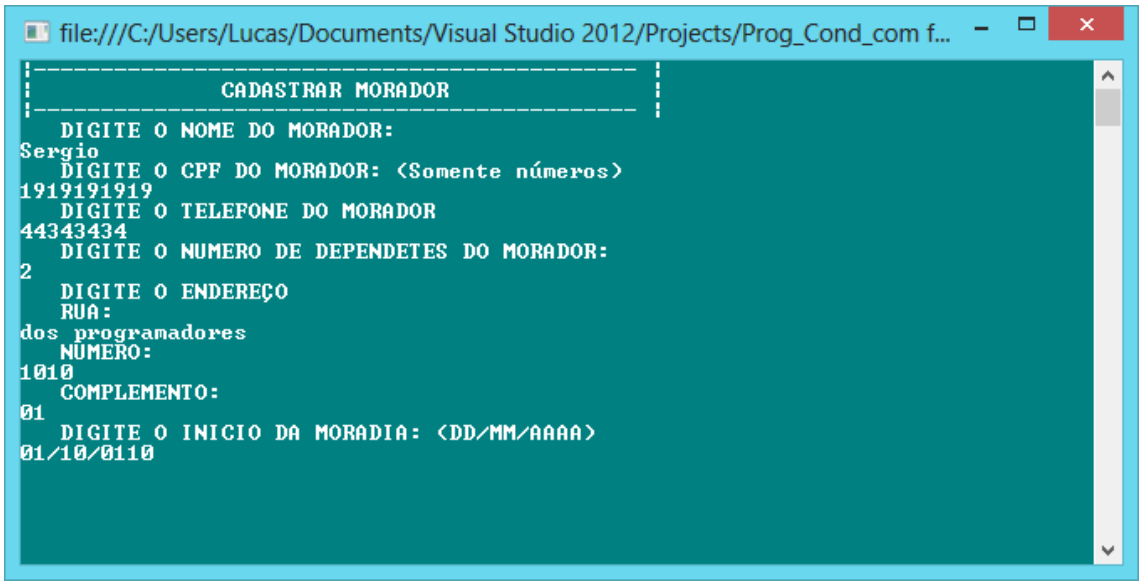
```
            Console.WriteLine("|-----------------------------------------------
|");
            Console.WriteLine("   DIGITE O NOME DO MORADOR:");
            morador.nome = Console.ReadLine().Replace(';', ' ').ToUpper();
            Console.WriteLine("   DIGITE O CPF DO MORADOR: (Somente números)");
            morador.cpf = Console.ReadLine();
            Console.WriteLine("   DIGITE O TELEFONE DO MORADOR");
            morador.tel = Console.ReadLine().Replace(';', ' ');
            Console.WriteLine("   DIGITE O NUMERO DE DEPENDETES DO MORADOR:");
            morador.num_dependentes =
Convert.ToInt16(Console.ReadLine().Replace(';', ' '));
            Console.WriteLine("   DIGITE O ENDEREÇO");
            Console.WriteLine("   RUA:");
            morador.rua = Console.ReadLine().Replace(';', ' ').ToUpper();
            Console.WriteLine("   NUMERO:");
            morador.num = Convert.ToInt16(Console.ReadLine().Replace(';', ' '));
            Console.WriteLine("   COMPLEMENTO:");
            morador.complemento = Convert.ToInt16(Console.ReadLine().Replace(';',
' '));
            Console.WriteLine("   DIGITE O INICIO DA MORADIA: (DD/MM/AAAA)");
            morador.inicio_moradia = DateTime.Parse(Console.ReadLine());


            Console.Clear();

            StreamReader reader = new StreamReader(localDados +
arquivoDadosMorador);


            int contaRegistro = 0;

            while (reader.ReadLine() != null)
            {
                contaRegistro++;
            }

            reader.Close();

            reader.Dispose();


            StreamWriter writer = File.AppendText(localDados +
arquivoDadosMorador);


                morador.cod = contaRegistro + 1;


                writer.WriteLine(morador.cod + ";" + morador.nome + ";" +
morador.cpf + ";" + morador.tel + ";" + morador.num_dependentes + ";" +
morador.rua + ";" + morador.num + ";" + morador.complemento+ ";" +
morador.inicio_moradia.ToShortDateString());

            writer.Close();

            writer.Dispose();

            Console.WriteLine("|-----------------------------------------------
|");
            Console.WriteLine("|             CODIGO DO MORADOR: " + morador.cod +
"              |");
```

```
            Console.WriteLine("|-----------------------------------------------
|");
            Console.WriteLine("|                    SUCESSO
|");
            Console.WriteLine("|-----------------------------------------------
|");
        }
```

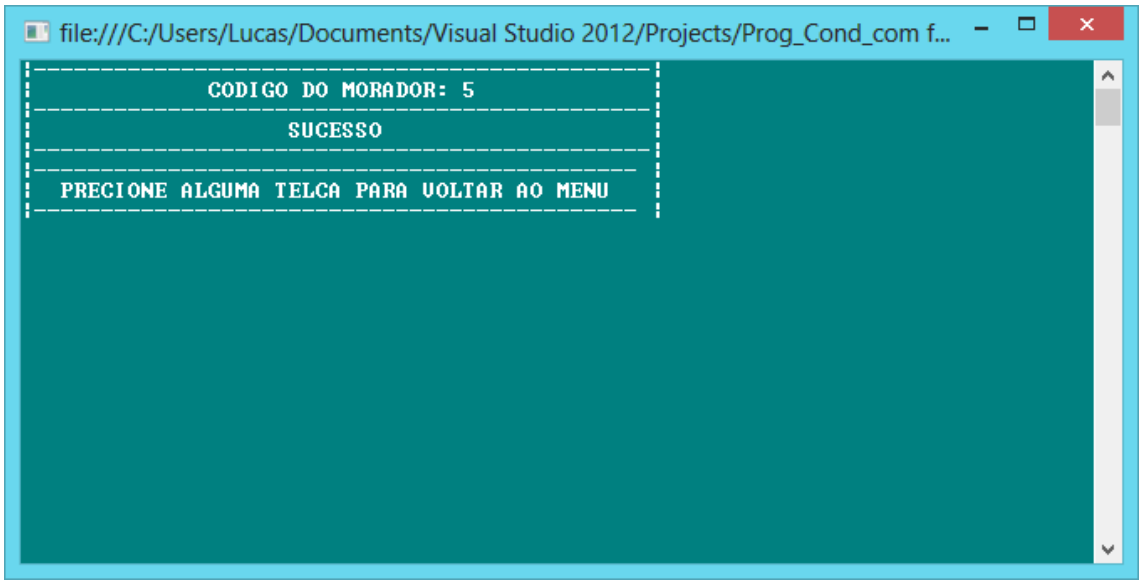
Da mesma forma que cadastramos condomínio podemos fazer o cadastro de morador, com alguns itens a mais, o usuário digita os itens e salvamos separados pelo ponto e virgula para a busca futuramente. Tudo fica salvo no arquivo Morador.txt separado pelo ponto e virgula.



Tela de cadastro de morador



Depois do usuario digitar os dados. Ele confirma e mostra o código para o usuario.

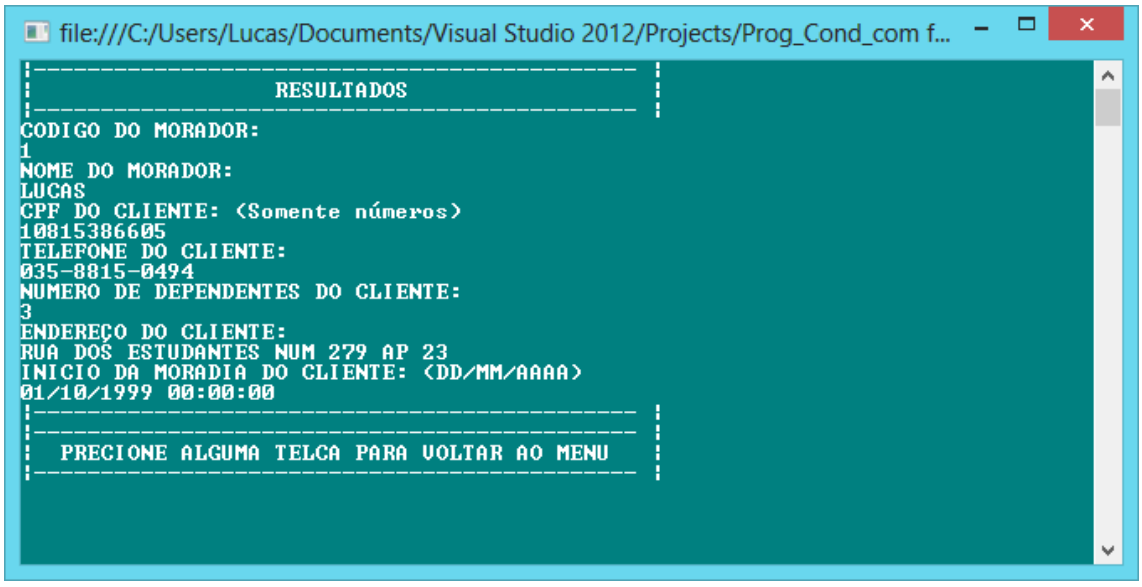


Função consulta_morador()

```
static void consulta_morador()
        {
                Console.Clear();
                Console.WriteLine("|---------------------------------------------
|");
                Console.WriteLine("|                BUSCAR MORADOR
|");
                Console.WriteLine("|---------------------------------------------
|");
                Console.WriteLine("   DIGITE O NOME DO MORADOR:");
                string termoBusca = Console.ReadLine().Replace(';', '
').ToUpper();


                Console.Clear();
                Console.WriteLine("|---------------------------------------------
|");
                Console.WriteLine("|                   RESULTADOS
|");
                Console.WriteLine("|---------------------------------------------
|");


                StreamReader reader = new StreamReader(localDados +
arquivoDadosMorador);

                string resultado;
                while ((resultado = reader.ReadLine()) != null)
                {

                    if (resultado.Contains(termoBusca))
                    {

                        string[] morador_resultado = resultado.Split(';');


                        tipo_morador morador;
                        morador.cod = Convert.ToInt32(morador_resultado[0]);
                        morador.nome = morador_resultado[1];
```

```
                        morador.cpf = morador_resultado[2];
                        morador.tel = morador_resultado[3];
                        morador.num_dependentes =
Convert.ToInt16(morador_resultado[4]);
                        morador.rua = morador_resultado[5];
                        morador.num = Convert.ToInt16(morador_resultado[6]);
                        morador.complemento =
Convert.ToInt16(morador_resultado[7]);
                        morador.inicio_moradia =
DateTime.Parse(morador_resultado[8]);

                        Console.WriteLine("CODIGO DO MORADOR:");
                        Console.WriteLine(morador.cod);
                        Console.WriteLine("NOME DO MORADOR:");
                        Console.WriteLine(morador.nome);
                        Console.WriteLine("CPF DO CLIENTE: (Somente números)");
                        Console.WriteLine(morador.cpf);
                        Console.WriteLine("TELEFONE DO CLIENTE:");
                        Console.WriteLine(morador.tel);
                        Console.WriteLine("NUMERO DE DEPENDENTES DO CLIENTE:");
                        Console.WriteLine(morador.num_dependentes);
                        Console.WriteLine("ENDEREÇO DO CLIENTE:");
                        Console.WriteLine("RUA "+ morador.rua + " NUM " +
morador.num + " AP " + morador.complemento);
                        Console.WriteLine("INICIO DA MORADIA DO CLIENTE:
(DD/MM/AAAA)");
                        Console.WriteLine(morador.inicio_moradia);
                        Console.WriteLine("|-------------------------------------
-------- |");
                    }
                }

                reader.Close();

                reader.Dispose();
            }
```

Telas da Consulta

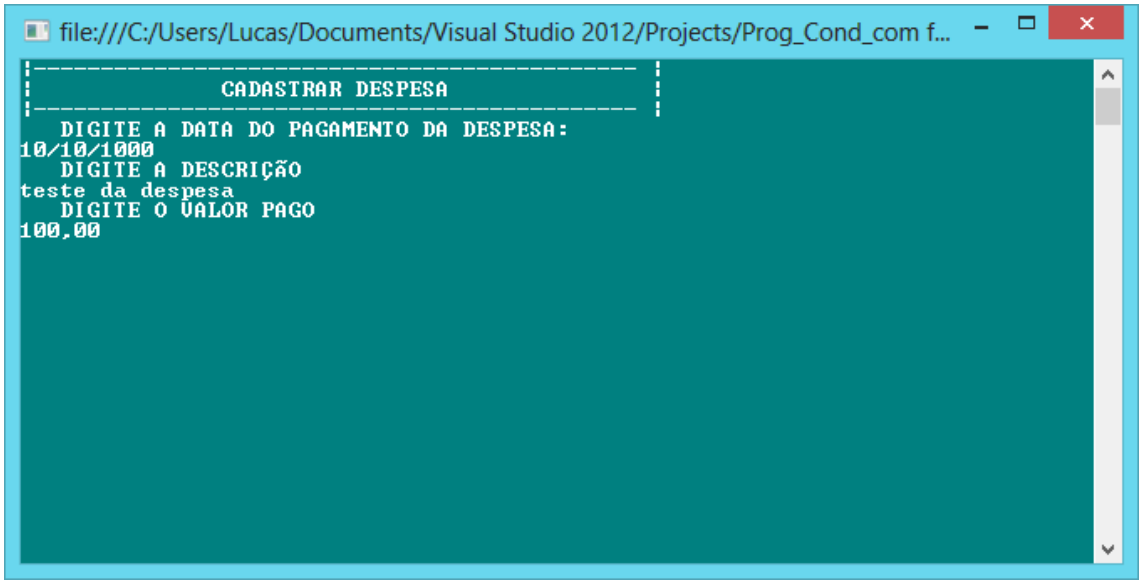


Função cadastro_despesa()

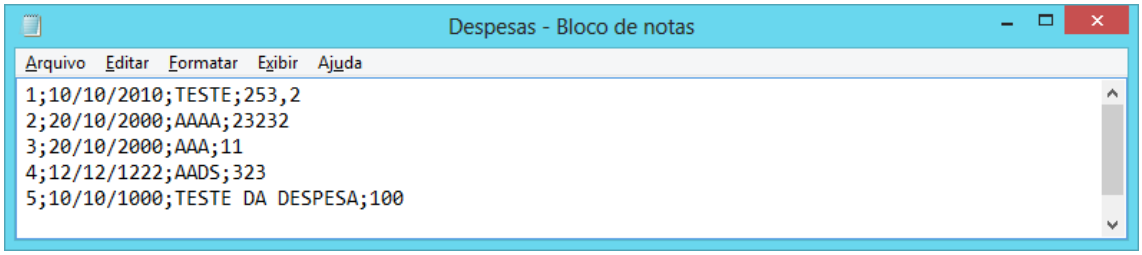
```
static void cadastro_despesa()
        {
                Console.Clear();


                tipo_despesa despesa;
                Console.WriteLine("|--------------------------------------------
|");
                Console.WriteLine("|              CADASTRAR DESPESA
|");
                Console.WriteLine("|--------------------------------------------
|");
                Console.WriteLine("   DIGITE A DATA DO PAGAMENTO DA DESPESA:");
                despesa.data_pagamento = DateTime.Parse(Console.ReadLine());
                Console.WriteLine("   DIGITE A DESCRIÇÃO");
                despesa.descricao = Console.ReadLine().Replace(';', '
').ToUpper();
                Console.WriteLine("   DIGITE O VALOR PAGO");
                despesa.valor = Convert.ToDouble(Console.ReadLine());
                Console.Clear();


                StreamReader reader = new StreamReader(localDados +
arquivoDadosDespesas);


                int contaRegistro = 0;

                while (reader.ReadLine() != null)
                {
                    contaRegistro++;
                }

                reader.Close();

                reader.Dispose();


                StreamWriter writer = File.AppendText(localDados +
arquivoDadosDespesas);


                despesa.cod = contaRegistro + 1;



                writer.WriteLine(despesa.cod + ";" +
despesa.data_pagamento.ToShortDateString() + ";" + despesa.descricao + ";" +
despesa.valor);

                writer.Close();

                writer.Dispose();

                Console.WriteLine("|--------------------------------------------
-|");
                Console.WriteLine("|             CODIGO DA DESPESA: " +
despesa.cod + "             |");
```

```
                Console.WriteLine("|-----------------------------------------
-|");
                Console.WriteLine("|                    SUCESSO
|");
                Console.WriteLine("|-----------------------------------------
-|");
            }
```

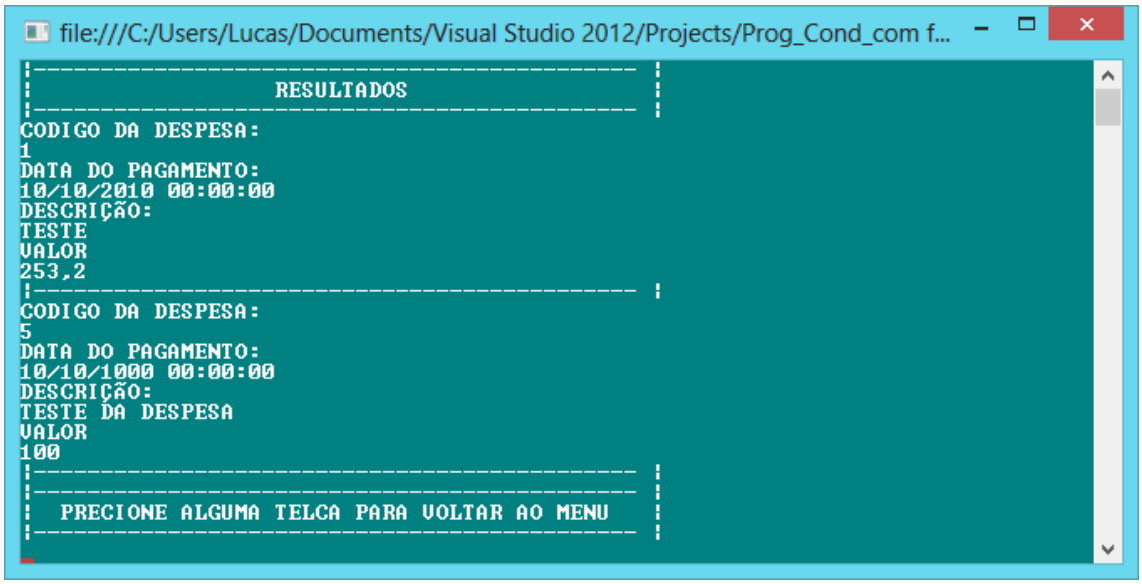
Telas do cadastro da despesas



Depois de salvo no arquivo



Função consulta_despesa()


file:///C:/Users/Lucas/Documents/Visual Studio 2012/Projects/Prog_Cond_com f...
RESULTADOS
CODIGO DA DESPESA:
1
DATA DO PAGAMENTO:
10/10/2010 00:00:00
DESCRIÇÃO:
TESTE
VALOR
253,2
CODIGO DA DESPESA:
5
DATA DO PAGAMENTO:
10/10/1000 00:00:00
DESCRIÇÃO:
TESTE DA DESPESA
VALOR
100
PRECIONE ALGUMA TELCA PARA VOLTAR AO MENU

REFERÊNCIAS BIBLIOGRÁFICAS

DEITEL ,H. M.; DEITEL, P.J. ; LISTFIELD J.; NIETO T.R.; YAEGER C.; ZLATKINA M.. C# - Como programar. Tradução. João Eduardo Nóbrega Tortello; revisão técnica Alvaro Antunes

São Paulo: Pearson Education, 2003

Engenharia de Software – Disponível em: <http://www.governancamunicipal.sp.gov.br/conteudo/arquivos/Analise%20de%20requisitos.pdf >– Acessado em: 17 de setembro de 2013.