FECAP

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

Lucy Mari Tabuti

PROJETO INTEGRADOR – ENTREGA 18/05/2025

FLEXHOUSE – DASHBOARD

GRUPO 42

19010077 - Caio Araujo

25027720 - Gustavo Bitencourt Silva

25027029 - Jorge Alves Marinho

25027317 - Matheus Breda Andreo

São Paulo

1 / 2025

Algoritmo FlexHouse

```
// Importação de bibliotecas necessárias para o funcionamento do Windows Forms, manipulação de arquivos e dados
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

// Namespace define um agrupamento lógico de classes. Aqui estamos criando o projeto FlexHouse
namespace FlexHouse
{
  // A classe DashBoard_Form herda da classe Form, ou seja, é uma janela do aplicativo
  public partial class DashBoard_Form : Form
  {
    // Caminho do arquivo CSV que armazena os dados da casa inteligente
    string caminhoCSV_Base = "BD_Casa_Inteligente.csv";
```

```
// Variáveis auxiliares
string linhaSeparada, id_Sensor_String;
List<string> listaFiltrado;
List<string> listaOrdenada;
List<string> listaGastoOrdenada;

// Variável pública para representar o ID do sensor atual
public static int id_sensor_int = 1;

// Variáveis para cálculo de consumo
DateTime dataRequisicao;
TimeSpan horasAtivo;
double horas, potenciaWatts, consumoWH;

// Formatos de data aceitos no CSV
string[] formatos = {
    "d/M/yy",
    "dd/MM/yyyy",
    "d/M/yy",
    "dd/M/yy"
};

// Construtor: executado automaticamente ao iniciar a janela
public DashBoard_Form()
{
    InitializeComponent(); // Inicializa os componentes gráficos do formulário
```

```
            loadForm(new home_Pag()); // Carrega a página inicial

            // Loop para calcular consumo de cada sensor (de 1 a 5)
            for (id_sensor_int = 1; id_sensor_int <= 5; id_sensor_int++)
            {
               Escolha(); // Define o nome e a potência do sensor atual

               listaGastoOrdenada = new List<string>();
               listaGastoOrdenada.Add("Data;Sensor;Hora_Ativo;ConsumoWH"); // Cabeçalho do CSV de saída

               // Loop por todos os dias de janeiro a julho
               for (int mes = 1; mes <= 7; mes++)
               {
                  int diasMes = DateTime.DaysInMonth(2025, mes);

                  for (int dia = 1; dia <= diasMes; dia++)
                  {
                     dataRequisicao = new DateTime(2025, mes, dia);
                     FiltroListaCSV(dataRequisicao); // Filtra e ordena os dados do dia
                     CalculoWattsH(potenciaWatts);   // Calcula consumo com base nas horas ativas

                     // Monta e adiciona uma linha ao relatório
                     string registro = $"{dataRequisicao:dd/MM/yyyy};{id_Sensor_String};{horasAtivo};{consumoWH:F2}";
                     listaGastoOrdenada.Add(registro);
                  }
```

```
        }

        // Salva os dados filtrados e processados em um novo arquivo CSV
        string nomeArquivo = $"Ordenado_{id_Sensor_String}_2025.csv";
        File.WriteAllLines(nomeArquivo, listaGastoOrdenada);
    }
}

// Define o nome e potência de acordo com o ID do sensor
public void Escolha()
{
    switch (id_sensor_int)
    {
        case 1:
            id_Sensor_String = "Quarto 01";
            potenciaWatts = 1500;
            break;
        case 2:
            id_Sensor_String = "Quarto 02";
            potenciaWatts = 1500;
            break;
        case 3:
            id_Sensor_String = "Sala";
            potenciaWatts = 50;
            break;
        case 4:
```

```
            id_Sensor_String = "Cozinha";
            potenciaWatts = 3000;
            break;
        case 5:
            id_Sensor_String = "Piscina";
            potenciaWatts = 7000;
            break;
    }
}

// Anima o menu lateral expandindo ou recolhendo
bool menuExpand = false;
private void MenuTransition_Tick(object sender, EventArgs e)
{
    if (!menuExpand)
    {
        MenuContainer.Height += 10;
        if (MenuContainer.Height >= 330)
        {
            MenuTransition.Stop();
            menuExpand = true;
        }
    }
    else
    {
        MenuContainer.Height -= 10;
```

```
        if (MenuContainer.Height <= 55)
        {
            MenuTransition.Stop();
            menuExpand = false;
        }
    }
}

// Carrega o formulário da página principal
private void bnt_Home_Click(object sender, EventArgs e)
{
    MenuTransition.Start();
    loadForm(new home_Pag());
}

// Método genérico que carrega qualquer formulário na área principal
public void loadForm(object form)
{
    if (this.Panel_main.Controls.Count > 0)
        this.Panel_main.Controls.RemoveAt(0); // Remove formulário atual

    Form f = form as Form; // Converte o objeto para formulário
    f.TopLevel = false;
    f.Dock = DockStyle.Fill; // Preenche o painel
    this.Panel_main.Controls.Add(f);
    this.Panel_main.Tag = f;
```

```
            f.Show(); // Exibe o novo formulário
        }

        // Botões para navegar entre ambientes
        private void btn_Qrt1_Click(object sender, EventArgs e) => loadForm(new AmbientePag(1, "Quarto 01", "1,5"));
        private void btn_Qrt2_Click(object sender, EventArgs e) => loadForm(new AmbientePag(1, "Quarto 02", "1,5"));
        private void bnt_sala_Click(object sender, EventArgs e) => loadForm(new AmbientePag(1, "Sala", "0,05"));
        private void btn_sala_Click(object sender, EventArgs e) => loadForm(new AmbientePag(1, "Cozinha", "3"));
        private void btn_Piscina_Click(object sender, EventArgs e) => loadForm(new AmbientePag(1, "Piscina", "7"));
        private void button2_Click(object sender, EventArgs e) => loadForm(new RankingForms());
        private void button1_Click(object sender, EventArgs e) => loadForm(new Gasto(1, "Quarto 01"));
        private void exit_bnt_Click(object sender, EventArgs e) => Application.Exit(); // Fecha o app

        // Filtra os dados do CSV de acordo com a data e o sensor
        public void FiltroListaCSV(DateTime dataRequisicao)
        {
            string[] linhas = File.ReadAllLines(caminhoCSV_Base);
            listaOrdenada = new List<string> { linhas[0] }; // Cabeçalho

            for (int i = 1; i < linhas.Length; i++)
            {
                string linhaAtual = linhas[i].Replace(" ", ";");
                string[] coluna = linhaAtual.Split(';');
```

```
            if (coluna[2] == id_sensor_int.ToString())
            {
                if (DateTime.TryParseExact(coluna[0], formatos, CultureInfo.InvariantCulture,
DateTimeStyles.None, out DateTime data))
                {
                    if (TimeSpan.TryParse(coluna[1], out TimeSpan hora) && data.Date ==
dataRequisicao.Date)
                    {
                        string novaLinha = data.ToString("dd/MM/yyyy") + ";" + hora.ToString(@"hh\:mm") +
";" + string.Join(";", coluna.Skip(2));
                        listaOrdenada.Add(novaLinha);
                    }
                }
            }
        }

        // Ordena por data e hora
        listaOrdenada.Sort((linha1, linha2) =>
        {
            var col1 = linha1.Split(';');
            var col2 = linha2.Split(';');
            int comp = string.Compare(col1[0], col2[0]);
            return comp == 0 ? string.Compare(col1[1], col2[1]) : comp;
        });

        // Cálculo do tempo ativo do sensor
```

```
horasAtivo = TimeSpan.Zero;
DateTime horaMaior = DateTime.Parse("00:00");
DateTime horaMenor = DateTime.Parse("00:00");

for (int i = listaOrdenada.Count - 1; i >= 0; i--)
{
   string[] col = listaOrdenada[i].Split(';');
   if (col.Length < 6) continue;

   if (col[5] == "1" && TimeSpan.TryParse(col[1], out TimeSpan horaAtual))
   {
      if (horaAtual > horaMaior.TimeOfDay)
         horaMaior = DateTime.ParseExact(col[1], "HH:mm", CultureInfo.InvariantCulture);
   }

   if (col[5] == "0" && (horaMaior > horaMenor || i == 0))
   {
      horaMenor = DateTime.ParseExact(col[1], "HH:mm", CultureInfo.InvariantCulture);
      horasAtivo += horaMaior - horaMenor;
      horaMaior = DateTime.Parse("00:00");
   }
}

// Se o sensor não foi desligado, considera 1 hora ativa
if (horasAtivo <= TimeSpan.Zero)
   horasAtivo = TimeSpan.FromHours(1);
```

```
        }

        // Calcula o consumo de energia com base na potência e tempo ativo
        public void CalculoWattsH(double potenciaWH)
        {
            horas = horasAtivo.TotalHours;
            consumoWH = potenciaWH * horas;
        }
    }
}
```