Envio de Dados do Hardware para Web:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Timers;
using Newtonsoft.Json;
using System.IO;

namespace SmartHomeControlPanel
{
    public partial class MainForm : Form
    {
        // Classe para representar o status dos dispositivos
        public class DeviceStatus
        {
            public string DeviceId { get; set; }
            public string DeviceName { get; set; }
            public string RoomName { get; set; }
                int sensorId = roomEntry.Key;


                var sensorLatestData = latestData.FirstOrDefault(d => d.SensorId
== sensorId);
                if (sensorLatestData != null)
```

```
                {
                    double roomConsumption =
CalculateRoomMonthlyConsumption(roomName, sensorLatestData);

    public bool IsOn { get; set; }
    public double PowerConsumption { get; set; } // em Watts
    public DateTime LastStatusChange { get; set; }
}

// Instância do processador de dados
private SmartHomeDataProcessor dataProcessor;

// Lista de dispositivos controlados
private List<DeviceStatus> devices = new List<DeviceStatus>();

// Cliente HTTP para enviar dados para a web
private HttpClient httpClient;

// URL da API web
private string apiUrl = "https://sua-api-casa-inteligente.com/api/devices";

// Timer para simular leituras de sensores
private System.Timers.Timer sensorTimer;

// Timer para enviar dados para a web
private System.Timers.Timer webSyncTimer;

// Caminho para armazenar logs locais
private string logFilePath = "smart_home_logs.txt";

// Controles da interface
```

```
        private Dictionary<string, Dictionary<string, Button>> deviceButtons = new
Dictionary<string, Dictionary<string, Button>>();
        private Dictionary<string, Label> roomTempLabels = new
Dictionary<string, Label>();
        private Dictionary<string, Label> roomHumidityLabels = new
Dictionary<string, Label>();
        private Dictionary<string, Label> roomMovementLabels = new
Dictionary<string, Label>();
        private TextBox logTextBox;
        private Button syncNowButton;
        private CheckBox autoSyncCheckBox;
        private Label lastSyncLabel;

        public MainForm()
        {
            InitializeComponent();

            // Inicializar o processador de dados
            dataProcessor = new SmartHomeDataProcessor();

            InitializeDevices();
            InitializeHttpClient();
            InitializeTimers();
            InitializeUI();
        }

        private void InitializeComponent()
        {
            this.Text = "Painel de Controle Casa Inteligente";
            this.Size = new Size(1000, 700);
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;
```

```
    this.MaximizeBox = false;
    this.BackColor = Color.WhiteSmoke;
}

private void InitializeDevices()
{
    // Obter mapeamento de sensores para cômodos
    var sensorToRoom = dataProcessor.GetSensorToRoomMapping();

    // Obter dispositivos por cômodo
    var roomDevices = dataProcessor.GetRoomDevices();

    // Inicializar dispositivos para cada cômodo
    foreach (var roomEntry in roomDevices)
    {
        string roomName = roomEntry.Key;
        var deviceList = roomEntry.Value;

        foreach (var device in deviceList)
        {
            string deviceName = device.Item1;
            double powerConsumption = device.Item2;

            // Criar ID único para o dispositivo
            string deviceId = roomName.ToLower().Replace(" ", "_") + "_" +
                              deviceName.ToLower().Replace(" ", "_");

            // Determinar estado inicial (geladeira sempre ligada)
            bool isOn = deviceName.Contains("Geladeira");
```

```
                devices.Add(new DeviceStatus
                {
                    DeviceId = deviceId,
                    DeviceName = deviceName,
                    RoomName = roomName,
                    IsOn = isOn,
                    PowerConsumption = powerConsumption,
                    LastStatusChange = DateTime.Now
                });
            }
        }
    }

    private void InitializeHttpClient()
    {
        httpClient = new HttpClient();
        httpClient.Timeout = TimeSpan.FromSeconds(10);
        httpClient.DefaultRequestHeaders.Add("User-Agent",
"SmartHomeControlPanel");
    }

    private void InitializeTimers()
    {
        // Timer para simular leituras de sensores (a cada 5 segundos)
        sensorTimer = new System.Timers.Timer(5000);
        sensorTimer.Elapsed += OnSensorTimerElapsed;
        sensorTimer.AutoReset = true;
        sensorTimer.Start();

        // Timer para sincronizar com a web (a cada 30 segundos)
        webSyncTimer = new System.Timers.Timer(30000);
```

```
    webSyncTimer.Elapsed += OnWebSyncTimerElapsed;
    webSyncTimer.AutoReset = true;
    webSyncTimer.Start();
}

private void InitializeUI()
{
    // Obter mapeamento de sensores para cômodos
    var sensorToRoom = dataProcessor.GetSensorToRoomMapping();

    // Criar TabControl para organizar as abas por cômodo
    TabControl roomTabControl = new TabControl();
    roomTabControl.Dock = DockStyle.Top;
    roomTabControl.Height = 400;
    this.Controls.Add(roomTabControl);

    // Criar abas para cada cômodo
    foreach (var roomName in sensorToRoom.Values.Distinct())
    {
        TabPage roomTab = new TabPage(roomName);
        roomTabControl.TabPages.Add(roomTab);

        // Inicializar dicionário para os botões deste cômodo
        deviceButtons[roomName] = new Dictionary<string, Button>();

        // Painel para os dispositivos
        Panel devicePanel = new Panel();
        devicePanel.Dock = DockStyle.Top;
        devicePanel.Height = 250;
        devicePanel.AutoScroll = true;
```

```
            roomTab.Controls.Add(devicePanel);

            // Adicionar controles para cada dispositivo no cômodo
            int y = 20;
            foreach (var device in devices.Where(d => d.RoomName == roomName))
            {
                Label deviceLabel = new Label();
                deviceLabel.Text = device.DeviceName;
                deviceLabel.AutoSize = true;
                deviceLabel.Location = new Point(20, y);
                devicePanel.Controls.Add(deviceLabel);

                Button toggleButton = new Button();
                toggleButton.Text = device.IsOn ? "Desligar" : "Ligar";
                toggleButton.BackColor = device.IsOn ? Color.LightGreen : Color.LightGray;
                toggleButton.Size = new Size(80, 30);
                toggleButton.Location = new Point(200, y - 5);
                toggleButton.Tag = device.DeviceId;
                toggleButton.Click += ToggleDevice_Click;
                devicePanel.Controls.Add(toggleButton);

                // Guardar referência ao botão
                deviceButtons[roomName][device.DeviceId] = toggleButton;

                Label powerLabel = new Label();
                powerLabel.Text = $"Consumo: {device.PowerConsumption}W";
                powerLabel.AutoSize = true;
                powerLabel.Location = new Point(300, y);
                devicePanel.Controls.Add(powerLabel);
```

```
            y += 40;
        }

        // Painel para os sensores
        Panel sensorPanel = new Panel();
        sensorPanel.Dock = DockStyle.Bottom;
        sensorPanel.Height = 100;
        sensorPanel.BorderStyle = BorderStyle.FixedSingle;
        roomTab.Controls.Add(sensorPanel);

        // Adicionar labels para os dados dos sensores
        Label sensorTitleLabel = new Label();
        sensorTitleLabel.Text = "Dados do Sensor:";
        sensorTitleLabel.Font = new Font(sensorTitleLabel.Font,
FontStyle.Bold);
        sensorTitleLabel.AutoSize = true;
        sensorTitleLabel.Location = new Point(20, 10);
        sensorPanel.Controls.Add(sensorTitleLabel);

        Label tempLabel = new Label();
        tempLabel.Text = "Temperatura: --°C";
        tempLabel.AutoSize = true;
        tempLabel.Location = new Point(20, 40);
        sensorPanel.Controls.Add(tempLabel);
        roomTempLabels[roomName] = tempLabel;

        Label humidityLabel = new Label();
        humidityLabel.Text = "Umidade: --%";
        humidityLabel.AutoSize = true;
        humidityLabel.Location = new Point(200, 40);
```

```
    sensorPanel.Controls.Add(humidityLabel);
    roomHumidityLabels[roomName] = humidityLabel;

    Label movementLabel = new Label();
    movementLabel.Text = "Movimento: Não";
    movementLabel.AutoSize = true;
    movementLabel.Location = new Point(380, 40);
    sensorPanel.Controls.Add(movementLabel);
    roomMovementLabels[roomName] = movementLabel;
}

// Painel para logs e sincronização
Panel logPanel = new Panel();
logPanel.Dock = DockStyle.Bottom;
logPanel.Height = 200;
this.Controls.Add(logPanel);

Label logTitleLabel = new Label();
logTitleLabel.Text = "Logs e Sincronização:";
logTitleLabel.Font = new Font(logTitleLabel.Font, FontStyle.Bold);
logTitleLabel.AutoSize = true;
logTitleLabel.Location = new Point(20, 10);
logPanel.Controls.Add(logTitleLabel);

logTextBox = new TextBox();
logTextBox.Multiline = true;
logTextBox.ScrollBars = ScrollBars.Vertical;
logTextBox.ReadOnly = true;
logTextBox.Size = new Size(700, 120);
logTextBox.Location = new Point(20, 40);
```

```
            logPanel.Controls.Add(logTextBox);

            syncNowButton = new Button();
            syncNowButton.Text = "Sincronizar Agora";
            syncNowButton.Size = new Size(150, 30);
            syncNowButton.Location = new Point(750, 40);
            syncNowButton.Click += SyncNow_Click;
            logPanel.Controls.Add(syncNowButton);

            autoSyncCheckBox = new CheckBox();
            autoSyncCheckBox.Text = "Sincronização Automática";
            autoSyncCheckBox.AutoSize = true;
            autoSyncCheckBox.Checked = true;
            autoSyncCheckBox.Location = new Point(750, 80);
            autoSyncCheckBox.CheckedChanged += AutoSync_CheckedChanged;
            logPanel.Controls.Add(autoSyncCheckBox);

            lastSyncLabel = new Label();
            lastSyncLabel.Text = "Última sincronização: Nunca";
            lastSyncLabel.AutoSize = true;
            lastSyncLabel.Location = new Point(750, 120);
            logPanel.Controls.Add(lastSyncLabel);

            // Inicializar a interface com os dados atuais
            UpdateSensorUI();
            LogMessage("Sistema inicializado. Aguardando dados dos
sensores...");

            // Mostrar estatísticas iniciais
            DisplayInitialStatistics();
        }
```

```
        private void DisplayInitialStatistics()
        {
            var stats = dataProcessor.GetHomeStatistics();

            LogMessage($"Estatísticas da Casa:");
            LogMessage($"Temperatura média:
{stats["AverageTemperature"]:F1}°C");
            LogMessage($"Umidade média: {stats["AverageHumidity"]:F1}%");
            LogMessage($"Consumo atual estimado:
{stats["CurrentConsumption"]:F2} kW");
            LogMessage($"Custo mensal estimado: R$ {stats["MonthlyCost"]:F2}");
            LogMessage($"Cômodo com maior consumo:
{stats["HighestConsumptionRoom"]}");
        }

        private void ToggleDevice_Click(object sender, EventArgs e)
        {
            Button button = (Button)sender;
            string deviceId = (string)button.Tag;

            // Encontrar o dispositivo
            var device = devices.FirstOrDefault(d => d.DeviceId == deviceId);
            if (device != null)
            {
                // Alternar estado
                device.IsOn = !device.IsOn;
                device.LastStatusChange = DateTime.Now;

                // Atualizar aparência do botão
                button.Text = device.IsOn ? "Desligar" : "Ligar";
                button.BackColor = device.IsOn ? Color.LightGreen : Color.LightGray;
```

```
            // Registrar ação
            LogMessage($"Dispositivo {device.DeviceName} em
{device.RoomName} foi {(device.IsOn ? "LIGADO" : "DESLIGADO")}.");


            // Enviar dados para
```