Simulação de Sensores

PIR HC-SR501 + DHT11

Eric Butzloff Gudera 25001129
Gabrielly Cristina Dos Reis 25000906
Lindsay Cristine Oliveira Souza - 25000762

Código:

```
//main.dart
import 'dart:async';
import 'dart:math';

void main() async {
  print('=== SISTEMA DE MONITORAMENTO PACKBAG ===');
  print('Filiais: Aguai e Casa Branca');
  print('Sensores: PIR HC-SR501 + DHT11\n');

  SimuladorService simuladorService = SimuladorService();

  await simuladorService.inicializarSensores();

  Dashboard dashboard = Dashboard(simuladorService);

  await dashboard.executarMonitoramento(30);
}

//leitura_sensor.dart
class LeituraSensor {
  // atributos da classe - sensores PIR e DHT11
  double temperatura;
  double umidade;
  bool movimentoDetectado;
  bool lampada;
  String horaTemperatura;
  String localFilial;

  //construtor da classe
  LeituraSensor({
    required this.temperatura,
    required this.umidade,
    required this.movimentoDetectado,
    required this.lampada,
```

```
    required this.horaTemperatura,
    required this.localFilial,
  });

  // factory method igual o do professor
  factory LeituraSensor.fromJson(Map<String, dynamic> json) {
    return LeituraSensor(
      temperatura: (json['temperatura'] as num).toDouble(),
      umidade: (json['umidade'] as num).toDouble(),
      movimentoDetectado: (json['movimentoDetectado'] as bool),
      lampada: (json['lampada'] as bool),
      horaTemperatura: (json['horaTemperatura'] as String),
      localFilial: (json['localFilial'] as String),
    );
  }

  // metodo para converter para Map
  Map<String, dynamic> toJson() {
    return {
      'temperatura': temperatura,
      'umidade': umidade,
      'movimentoDetectado': movimentoDetectado,
      'lampada': lampada,
      'horaTemperatura': horaTemperatura,
      'localFilial': localFilial,
    };
  }

  @override
  String toString() {
    String statusMovimento = movimentoDetectado ? "DETECTADO" : "SEM MOVIMENTO";
    String statusLampada = lampada ? "LIGADA" : "DESLIGADA";

    return '[$localFilial] Temp: ${temperatura.toStringAsFixed(1)}C | '
        'Umidade: ${umidade.toStringAsFixed(1)}% | '
        'PIR: $statusMovimento | '
        'Lampada: $statusLampada | '
        '$horaTemperatura';
  }
}

//simulador_service.dart
class SimuladorService {
  // List para guardar historico das leituras
  List<LeituraSensor> historicoLeituras = [];

  // Map com as configuracoes dos sensores
  Map<String, dynamic> configuracoesSensores = {
```

```
  //DHT11
  'temperaturaMin': 18.0,
  'temperaturaMax': 32.0,
  'umidadeMin': 35.0,
  'umidadeMax': 85.0,
  //PIR HC-SR501
  'probabilidadeMovimento': 0.3,
  'tempoLampada': 8,
  'intervaloAtualizacao': 3,
  'filiais': ['Aguai', 'Casa Branca'],
 };

 Random random = Random();
 Map<String, DateTime> ultimoMovimento = {};

 Future<void> inicializarSensores() async {
  print('Inicializando sensores Packbag...');
  print('- PIR HC-SR501 (Sensor de movimento)');
  print('- DHT11 (Temperatura e Umidade)');
  print('- LED (Lampadas da empresa)\n');

  // inicializa para cada filial
  for (String filial in configuracoesSensores['filiais']) {
   ultimoMovimento[filial] = DateTime.now().subtract(Duration(minutes: 5));
  }

  await Future.delayed(Duration(seconds: 1));
 }

 Future<LeituraSensor?> gerarNovaLeitura() async {
  try {
   // escolhe filial aleatoria
   List<String> filiais = List.from(configuracoesSensores['filiais']);
   String filialAtual = filiais[random.nextInt(filiais.length)];

   // gera valores do DHT11
   double temperatura = configuracoesSensores['temperaturaMin'] +
                 random.nextDouble() *
                 (configuracoesSensores['temperaturaMax'] -
configuracoesSensores['temperaturaMin']);

   double umidade = configuracoesSensores['umidadeMin'] +
                random.nextDouble() *
                (configuracoesSensores['umidadeMax'] -
configuracoesSensores['umidadeMin']);

   // simula o PIR HC-SR501
```

```
    bool movimentoDetectado = random.nextDouble() <
configuracoesSensores['probabilidadeMovimento'];

    // controla a lampada baseado no movimento
    bool lampada = false;
    DateTime agora = DateTime.now();

    if (movimentoDetectado) {
      lampada = true;
      ultimoMovimento[filialAtual] = agora;
    } else {
      DateTime ultimoMov = ultimoMovimento[filialAtual]!;
      int segundosDesdeUltimoMovimento = agora.difference(ultimoMov).inSeconds;
      lampada = segundosDesdeUltimoMovimento <
configuracoesSensores['tempoLampada'];
    }

    // gera timestamp atual
    String horaAtual = DateTime.now().toString().substring(0, 19);

    // cria nova leitura usando padrao do professor
    Map<String, dynamic> dadosJson = {
      'temperatura': temperatura,
      'umidade': umidade,
      'movimentoDetectado': movimentoDetectado,
      'lampada': lampada,
      'horaTemperatura': horaAtual,
      'localFilial': filialAtual,
    };

    LeituraSensor novaLeitura = LeituraSensor.fromJson(dadosJson);

    // adiciona ao historico (List)
    historicoLeituras.add(novaLeitura);

    // mantem apenas as ultimas 10 leituras
    if (historicoLeituras.length > 10) {
      historicoLeituras.removeAt(0);
    }

    return novaLeitura;

  } catch (e) {
    print('Erro ao gerar leitura: $e');
    return null;
  }
 }
```

```
  // metodo para obter estatisticas (utiliza List e Map)
  Map<String, dynamic> obterEstatisticas() {
    if (historicoLeituras.isEmpty) {
      return {'erro': 'Nenhuma leitura disponível'};
    }

    List<double> temperaturas = historicoLeituras.map((l) => l.temperatura).toList();
    List<double> umidades = historicoLeituras.map((l) => l.umidade).toList();

    // estatisticas por filial
    Map<String, int> movimentosPorFilial = {};
    Map<String, int> lampadasLigadas = {};

    for (String filial in configuracoesSensores['filiais']) {
      movimentosPorFilial[filial] = historicoLeituras
          .where((l) => l.localFilial == filial && l.movimentoDetectado)
          .length;
      lampadasLigadas[filial] = historicoLeituras
          .where((l) => l.localFilial == filial && l.lampada)
          .length;
    }

    return {
      'totalLeituras': historicoLeituras.length,
      'temperaturaMedia': temperaturas.reduce((a, b) => a + b) / temperaturas.length,
      'temperaturaMax': temperaturas.reduce((a, b) => a > b ? a : b),
      'temperaturaMin': temperaturas.reduce((a, b) => a < b ? a : b),
      'umidadeMedia': umidades.reduce((a, b) => a + b) / umidades.length,
      'totalMovimentos': historicoLeituras.where((l) => l.movimentoDetectado).length,
      'totalLampadasLigadas': historicoLeituras.where((l) => l.lampada).length,
      'movimentosPorFilial': movimentosPorFilial,
      'lampadasPorFilial': lampadasLigadas,
    };
  }
}

//dashboard.dart
class Dashboard {
  final SimuladorService simuladorService;

  Dashboard(this.simuladorService);

  Future<void> executarMonitoramento(int duracaoSegundos) async {
    print('Iniciando monitoramento por $duracaoSegundos segundos...\n');

    Timer.periodic(Duration(seconds: 3), (timer) async {
      if (timer.tick * 3 >= duracaoSegundos) {
        timer.cancel();
```

```
        print('\n=== MONITORAMENTO FINALIZADO ===');
        exibirRelatorioFinal();
        return;
      }

      await atualizarDashboard();
    });

    await Future.delayed(Duration(seconds: duracaoSegundos));
  }

  Future<void> atualizarDashboard() async {
    print('\n' + '='*50);
    print('DASHBOARD PACKBAG - ${DateTime.now().toString().substring(0, 19)}');
    print('='*50);

    // gera nova leitura
    LeituraSensor? leitura = await simuladorService.gerarNovaLeitura();

    if (leitura != null) {
      print('\nLEITURA ATUAL:');
      print('  $leitura');

      // exibe status dos sensores
      exibirStatusSensores(leitura);

      // exibe alertas
      exibirAlertas(leitura);

      // exibe estatisticas
      exibirEstatisticas();
    } else {
      print('Erro ao obter leitura dos sensores');
    }

    print('\n' + '-'*50);
  }

  void exibirStatusSensores(LeituraSensor leitura) {
    print('\nSTATUS DOS SENSORES:');
    print('  DHT11 - Temperatura: ${leitura.temperatura.toStringAsFixed(1)}C | Umidade:
${leitura.umidade.toStringAsFixed(1)}%');
    print('  PIR HC-SR501 - ${leitura.movimentoDetectado ? "Movimento detectado" : "Sem
movimento"}');
    print('  LED/Lampada - ${leitura.lampada ? "Ligada" : "Desligada"}');
  }

  void exibirAlertas(LeituraSensor leitura) {
```

```
    print('\nALERTAS:');

    if (leitura.temperatura > 28.0) {
      print('  ATENCAO: Temperatura alta em ${leitura.localFilial}:
${leitura.temperatura.toStringAsFixed(1)}C');
    }

    if (leitura.temperatura < 20.0) {
      print('  ATENCAO: Temperatura baixa em ${leitura.localFilial}:
${leitura.temperatura.toStringAsFixed(1)}C');
    }

    if (leitura.umidade > 75.0) {
      print('  ATENCAO: Umidade alta em ${leitura.localFilial}:
${leitura.umidade.toStringAsFixed(1)}%');
    }

    if (leitura.umidade < 40.0) {
      print('  ATENCAO: Umidade baixa em ${leitura.localFilial}:
${leitura.umidade.toStringAsFixed(1)}%');
    }

    if (leitura.movimentoDetectado) {
      print('  INFO: Movimento detectado em ${leitura.localFilial} - Lampada ligada
automaticamente');
    }

    if (!leitura.lampada && !leitura.movimentoDetectado) {
      print('  INFO: Modo economia em ${leitura.localFilial} - Lampada desligada');
    }
  }

  void exibirEstatisticas() {
    Map<String, dynamic> stats = simuladorService.obterEstatisticas();

    if (stats.containsKey('erro')) {
      return;
    }

    print('\nESTATISTICAS (${stats['totalLeituras']} leituras):');
    print('  Temperatura media: ${(stats['temperaturaMedia'] as
double).toStringAsFixed(1)}C');
    print('  Umidade media: ${(stats['umidadeMedia'] as double).toStringAsFixed(1)}%');
    print('  Total movimentos: ${stats['totalMovimentos']}');
    print('  Total lampadas ligadas: ${stats['totalLampadasLigadas']} vezes');

    // dados por filial
    Map<String, int> movFilial = stats['movimentosPorFilial'];
```

```
    Map<String, int> lampadaFilial = stats['lampadasPorFilial'];

    print('  Por filial:');
    for (String filial in movFilial.keys) {
      print('    $filial: ${movFilial[filial]} movimentos, ${lampadaFilial[filial]} ativacoes de
lampada');
    }
  }

  void exibirRelatorioFinal() {
    print('\nRELATORIO FINAL PACKBAG:');
    print('='*40);

    Map<String, dynamic> stats = simuladorService.obterEstatisticas();

    if (stats.containsKey('erro')) {
      print('Nenhum dado coletado.');
      return;
    }

    print('EMPRESA: Packbag');
    print('FILIAIS: Aguai e Casa Branca');
    print('TOTAL LEITURAS: ${stats['totalLeituras']}');
    print('');

    print('ANALISE TERMICA (DHT11):');
    print('  Temperatura media: ${(stats['temperaturaMedia'] as
double).toStringAsFixed(1)}C');
    print('  Temperatura maxima: ${(stats['temperaturaMax'] as double).toStringAsFixed(1)}C');
    print('  Temperatura minima: ${(stats['temperaturaMin'] as double).toStringAsFixed(1)}C');
    print('  Umidade media: ${(stats['umidadeMedia'] as double).toStringAsFixed(1)}%');

    print('');
    print('ANALISE DE MOVIMENTO (PIR HC-SR501):');
    print('  Total movimentos detectados: ${stats['totalMovimentos']}');
    print('  Total ativacoes de lampada: ${stats['totalLampadasLigadas']}');

    Map<String, int> movFilial = stats['movimentosPorFilial'];
    Map<String, int> lampadaFilial = stats['lampadasPorFilial'];

    print('');
    print('ANALISE POR FILIAL:');
    for (String filial in movFilial.keys) {
      print('  $filial:');
      print('    - Movimentos: ${movFilial[filial]}');
      print('    - Lampada ligada: ${lampadaFilial[filial]} vezes');
    }
```

```
    print('');
    print('HISTORICO DAS ULTIMAS LEITURAS:');
    int contador = 1;
    for (LeituraSensor leitura in simuladorService.historicoLeituras) {
      print('  $contador. $leitura');
      contador++;
    }

    print('\nFim do monitoramento');
  }
}
```

//Salvar em arquivo .dart e executar com: `dart run arquivo.dart`