Excelente projeto 👏 — ele mistura IoT, backend e visualização de dados, e o **Azure** oferece exatamente o ecossistema ideal pra isso.  
Vamos montar o cenário passo a passo, indicando as **ferramentas do Azure** mais adequadas em cada parte do fluxo:

**🌡️ 1. Arduino → Azure**

Você precisa enviar os dados do Arduino (sensor de temperatura) para a nuvem.

**🔹 Opção 1 – Azure IoT Hub (recomendada para IoT)**

* É o **serviço principal de entrada de dados IoT** do Azure.
* O Arduino (ou ESP32/ESP8266, se estiver usando Wi-Fi) envia os dados diretamente para o IoT Hub via MQTT, AMQP ou HTTPS.
* O IoT Hub garante segurança, autenticação e gerenciamento de dispositivos.
* Ele atua como o “ponteiro central” dos dispositivos conectados.

📦 Use as bibliotecas Azure IoT Hub SDK for C ou Arduino para facilitar a comunicação.

**☁️ 2. Recepção e Armazenamento no Azure**

Depois que o IoT Hub recebe os dados, você decide onde armazená-los.

**🔹 Azure Stream Analytics**

* Processa os dados em tempo real vindos do IoT Hub.
* Pode aplicar filtros, médias, alertas, etc.
* Redireciona a saída para vários destinos, como bancos de dados, storage ou dashboards.

**🔹 Azure Data Lake ou Azure Blob Storage**

* Ideal se você quiser **guardar o histórico bruto** das leituras.

**🔹 Azure Cosmos DB (NoSQL) ou Azure SQL Database**

* Ideal para armazenar os valores de temperatura com timestamps para depois consultar via site.
* Escolha o Cosmos DB se quiser escalabilidade e baixa latência global.

**📊 3. Visualização e Site de Estatísticas**

Aqui entram duas partes: o backend (API) e a interface.

**🔹 Azure App Service (Web App)**

* Hospeda seu **site ASP.NET/C#** facilmente, com integração direta ao GitHub ou Azure DevOps.
* Esse site pode consumir os dados do banco (Cosmos DB ou SQL) e exibir estatísticas.

**🔹 Power BI (integrado ao Azure)**

* Se quiser **dashboards automáticos**, pode conectar o Power BI diretamente ao banco de dados.
* Dá pra criar gráficos de temperatura, médias diárias, máximas/mínimas etc., sem precisar codificar o front-end.

**⚙️ 4. Fluxo Resumido**

Arduino (sensor DHT11/22, por ex.)

↓ envia via MQTT/HTTP

Azure IoT Hub

↓

Azure Stream Analytics (opcional, para processar)

↓

Cosmos DB ou Azure SQL Database

↓

Azure Web App (site com gráficos e estatísticas)

ou Power BI (visualização automática)

**💡 Ferramentas principais do Azure no seu caso**

| **Etapa** | **Ferramenta Azure** | **Função** |
| --- | --- | --- |
| Receber dados do Arduino | **IoT Hub** | Canal seguro de entrada |
| Processar dados em tempo real | **Stream Analytics** | Filtrar, agregar, detectar eventos |
| Armazenar leituras | **Cosmos DB** ou **SQL Database** | Banco de dados |
| Mostrar estatísticas | **App Service** (para seu site ASP.NET) ou **Power BI** | Visualização |
| (Opcional) Automatizar pipelines | **Logic Apps** ou **Functions** | Processos automáticos |

Se quiser, posso te ajudar com um **exemplo completo do fluxo**, mostrando:

* o código do Arduino para enviar os dados,
* o endpoint no Azure IoT Hub,
* e o modelo de API no App Service para exibir as estatísticas.

Quer que eu monte esse exemplo?

**Você disse:**

Eu gostaria de fazer eu mesmo. Primeiro me diga quais ferramentas do azure tem o menor custo e em seguida me forneça sites com tutoriais para tudo isso. Se quiser, voce mesmo pode escrever o tutorial de cada parte

**O ChatGPT disse:**

Ótimo — podemos sim estruturar isso para você executar sozinho. Antes de passarmos aos tutoriais, vamos ver **quais ferramentas do Microsoft Azure têm o menor custo** (ou mesmo opções gratuitas) para o seu cenário, e depois eu te indico links de tutoriais + posso escrever um tutorial adaptado ao seu projeto.

**1) Ferramentas com menor custo / gratuitas possíveis**

Para seu fluxo (Arduino → Azure → site de estatísticas), essas são boas escolhas de baixo custo:

| **Ferramenta** | **Custo / plano gratuito relevante** | **Comentários** |
| --- | --- | --- |
| Azure IoT Hub | Existe a edição **Free**: até **8.000 mensagens/dia** e até 500 identidades de dispositivo. [Microsoft Azure+3 Microsoft Azure+3 Microsoft Learn+3](https://azure.microsoft.com/en-us/pricing/details/iot-hub/?utm_source=chatgpt.com) | Se for um dispositivo ou poucos, com leituras simples, isso pode dar. |
| Azure App Service (para o site) | Existe a camada “Free” ou “Shared” para desenvolvimento/teste. [Microsoft Learn+2 Microsoft Azure+2](https://learn.microsoft.com/en-us/azure/app-service/overview-hosting-plans?utm_source=chatgpt.com) | Ótima para protótipo — se for pouco tráfego, pode manter em plano gratuito ou muito barato. |
| Banco de dados / armazenamento leve | A conta “Free” da Azure lista várias ofertas gratuitas para novos clientes ou sempre gratuitas. [Microsoft Azure](https://azure.microsoft.com/en-us/pricing/free-services?utm_source=chatgpt.com) | Você pode usar um banco pequeno, ou até tabela simples, para começar. |
| Processamento de stream (opcional) – Azure Stream Analytics | É cobrada por unidade de streaming; para baixo volume fica barato. [Microsoft Learn](https://learn.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction?utm_source=chatgpt.com) | Se você não precisar de processamento sofisticado em real-time, talvez pule essa etapa agora para economizar. |

**Conclusão:** Para começar você pode usar: IoT Hub Free → armazenamento simples (SQL ou NoSQL leve) → App Service Free/Shared. Depois, se crescer, escalamos. Isso minimiza custos iniciais.

**2) Links de tutoriais úteis para cada parte**

Aqui vão bons sites/tutoriais que você pode seguir para cada parte do fluxo:

* Para IoT Hub:
  + “Send device telemetry to Azure IoT Hub” (Microsoft Learn) – [Microsoft Learn](https://learn.microsoft.com/en-us/azure/iot/tutorial-send-telemetry-iot-hub?utm_source=chatgpt.com)
  + “Create and manage Azure IoT hubs” (Microsoft Learn) – [Microsoft Learn](https://learn.microsoft.com/en-us/azure/iot-hub/create-hub?utm_source=chatgpt.com)
  + “Introduction to Azure IoT” – [Microsoft Learn+1](https://learn.microsoft.com/en-us/azure/iot/iot-introduction?utm_source=chatgpt.com)
* Para App Service / web app:
  + “Getting started with Azure App Service” – [Microsoft Learn](https://learn.microsoft.com/en-us/azure/app-service/getting-started?utm_source=chatgpt.com)
  + “Quickstart: Deploy an ASP.NET web app – Azure App Service” – [Microsoft Learn](https://learn.microsoft.com/en-us/azure/app-service/quickstart-dotnetcore?utm_source=chatgpt.com)
  + “Step-by-step guide to creating web apps and using App Service Plans” – [Medium](https://medium.com/%40marvinconejo/step-by-step-guide-to-creating-web-apps-and-using-app-service-plans-in-azure-a9a1d597d24f?utm_source=chatgpt.com)
* Para Stream Analytics (se você quiser):
  + “Azure Stream Analytics documentation / quickstarts” – [Microsoft Learn+1](https://learn.microsoft.com/en-us/azure/stream-analytics/?utm_source=chatgpt.com)
* Para acompanhar gratuitamente os serviços:
  + “Explore Free Azure Services” – mostra o que é grátis ou para testar. [Microsoft Azure](https://azure.microsoft.com/en-us/pricing/free-services?utm_source=chatgpt.com)

**3) Tutorial adaptado ao seu projeto**

A seguir um **tutorial passo-a-passo adaptado** ao cenário: seu Arduino coleta temperatura → envia para Azure → site mostra estatísticas. Você adapta para C# + seu site ASP.NET.

**Parte A – Preparar o Azure**

**A1. Criar conta e configurar plano gratuito**

* Se ainda não tem, crie uma conta Azure e escolha “Try for free” para ganhar crédito ou usar camadas gratuitas.
* No portal Azure, crie um *Resource Group* (ex: “IoTProjectRG”).
* Ative o plano gratuito “Free services” para os serviços que usar. [Microsoft Azure](https://azure.microsoft.com/en-us/pricing/free-services?utm_source=chatgpt.com)

**A2. Criar IoT Hub**

* No portal Azure → “Create a resource” → “Internet of Things” → “IoT Hub”. [Microsoft Learn](https://learn.microsoft.com/en-us/azure/iot-hub/create-hub?utm_source=chatgpt.com)
* Escolha a edição *Free* (se disponível) ou o mínimo necessário.
* Após criação, registre seu dispositivo (Arduino) como dispositivo no IoT Hub para obter *connection string*. [Microsoft Learn+1](https://learn.microsoft.com/en-us/azure/iot/tutorial-send-telemetry-iot-hub?utm_source=chatgpt.com)

**A3. Preparar banco de dados ou armazenamento**

* No portal Azure, escolha um banco: pode ser Azure SQL Database (relacional) ou Azure Cosmos DB (NoSQL).
* Configure uma tabela/coleção para armazenar: timestamp, temperatura, talvez dispositivo id.

**A4. Criar App Service para o site**

* No portal Azure → “Create a resource” → “Web + Mobile” → “App Service”. [Microsoft Learn+1](https://learn.microsoft.com/en-us/azure/app-service/getting-started?utm_source=chatgpt.com)
* Escolha o plano Free ou Shared se for experimentar. [Microsoft Learn](https://learn.microsoft.com/en-us/azure/app-service/overview-hosting-plans?utm_source=chatgpt.com)
* Configure sua aplicação ASP.NET (C#) para ser hospedada ali.

**Parte B – Código do Arduino para envio de dados**

**B1. No Arduino (ou ESP8266/ESP32) você faz algo assim (esqueleto):**

#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

const char\* ssid = "SEU\_SSID";

const char\* password = "SUA\_SENHA";

const char\* endpoint = "https://SEU\_SITE.azurewebsites.net/api/temperatura"; // ou envio direto ao IoT Hub via MQTT

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("WiFi conectado");

}

void loop() {

float temperatura = // … mede usando seu sensor (DHT11, DHT22, etc.)

if (WiFi.status() == WL\_CONNECTED) {

HTTPClient http;

http.begin(endpoint);

http.addHeader("Content-Type", "application/json");

StaticJsonDocument<200> doc;

doc["deviceId"] = "Arduino01";

doc["timestamp"] = millis();

doc["temperatura"] = temperatura;

String body;

serializeJson(doc, body);

int httpResponseCode = http.POST(body);

if (httpResponseCode > 0) {

Serial.println("Dados enviados: " + body);

} else {

Serial.println("Erro no envio: " + String(httpResponseCode));

}

http.end();

}

delay(60000); // envia a cada 60 segundos, por exemplo

}

**B2. Alternativamente, envio direto ao IoT Hub via MQTT ou usando SDK Azure IoT para o tipo de dispositivo.**  
Veja o tutorial “Send device telemetry to Azure IoT Hub” para detalhes. [Microsoft Learn](https://learn.microsoft.com/en-us/azure/iot/tutorial-send-telemetry-iot-hub?utm_source=chatgpt.com)

**Parte C – Receber os dados no Azure / armazenar**

**C1. Se usar IoT Hub → seu backend (por exemplo uma Function ou Web API) lê mensagens quando elas chegam.**

* No IoT Hub, ative *Message Routing* ou um *Event Hub* / *Storage* como destino.
* Crie uma Web API em C# (ASP.NET Core) que consome as mensagens ou que seja chamada diretamente pelo Arduino (como no B1).
* Insira na lógica da API: ao receber sensor data, salve no banco (SQL ou Cosmos).

**C2. Se quiser processamento extra (opcional)**

* Use Azure Stream Analytics para agregações em tempo real (por ex: média temperatura por hora) se necessário. [Microsoft Learn](https://learn.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction?utm_source=chatgpt.com)

**Parte D – Construir o site de estatísticas**

**D1. No projeto ASP.NET / Web App você implementa:**

* Página ou API que consulta o banco para obter dados (ex: últimas 24 h, mínimas/máximas, média)
* Gráfico ou tabela (pode usar libraria JS como Chart.js) para mostrar essas estatísticas
* Hospede este site via App Service criado em A4

**D2. Deploy no Azure App Service**

* No Visual Studio ou via GitHub Actions, configure deploy para o App Service.
* Após deploy, acesse https://seusite.azurewebsites.net e verifique se está online.
* Se quiser autenticação ou domínio customizado, siga tutorial “Build a secure web app on Azure App Service” [Microsoft Learn](https://learn.microsoft.com/en-us/azure/app-service/scenario-secure-app-overview?utm_source=chatgpt.com)

**Parte E – Otimização de custo para seu caso**

* Use plano Free ou Shared nos serviços enquanto for protótipo.
* Mantenha frequência de envio moderada (ex: 1-5 minutos) para não ultrapassar gratuitamente o IoT Hub.
* Use banco leve, não incremente muitos recursos até que seja necessário.
* Monitore o uso e orçamento no portal Azure para evitar surpresas.

Perfeito 👍 Então vamos montar **um tutorial completo em português**, com **passos detalhados, capturas simuladas e comandos**, dividido em **etapas práticas**, para você criar **do zero** o fluxo:

**Arduino → Azure IoT Hub → Banco de Dados → Site ASP.NET (App Service)**

O foco será **baixo custo** (usando planos gratuitos) e **execução passo a passo**.

**🌐 Tutorial: Projeto Arduino + Azure (IoT Hub + Web App + Banco)**

**🧩 ETAPA 1 – Criar conta e configurar ambiente gratuito**

**🔹 Passo 1.1 — Criar conta Azure**

1. Vá para https://azure.microsoft.com/pt-br/free
2. Clique em **"Comece gratuitamente"**.
3. Cadastre-se com sua conta Microsoft (Hotmail/Outlook).
4. Você ganhará **US$ 200 de crédito por 30 dias** + **camadas gratuitas** (IoT Hub Free, App Service Free etc).

💡 Dica: não é necessário cartão para experimentar os serviços gratuitos básicos.

**🔹 Passo 1.2 — Criar um *Resource Group***

Um *Resource Group* é um contêiner lógico para manter todos os seus recursos juntos.

1. No portal https://portal.azure.com
2. Pesquise por **“Resource groups”** → **Create**
3. Nomeie como: IoTProjectRG
4. Escolha uma região (ex: *Brazil South*)
5. Clique em **Review + Create** → **Create**

**🛰️ ETAPA 2 – Criar e configurar o IoT Hub**

**🔹 Passo 2.1 — Criar o IoT Hub**

1. No portal → **Create a resource** → procure por **IoT Hub**
2. Clique em **Create**
3. Em *Subscription*, escolha sua assinatura
4. Em *Resource Group*, selecione IoTProjectRG
5. Nomeie o hub (ex: temp-iothub-demo)
6. Selecione **Free Tier (F1)** — até **8.000 mensagens/dia**
7. Região: *Brazil South*
8. Clique em **Review + Create** → **Create**

**🔹 Passo 2.2 — Registrar o dispositivo (Arduino)**

1. Após criado o IoT Hub, vá em:  
   **IoT Hub → Devices → + New Device**
2. Nomeie: arduino-temp-01
3. Mantenha as opções padrão e clique em **Save**
4. Clique no dispositivo criado e copie o **Primary Connection String**  
   (será usada no código Arduino)

**🌡️ ETAPA 3 – Programar o Arduino**

**🔹 Requisitos**

* Placa com Wi-Fi (ESP8266 ou ESP32)
* Sensor de temperatura (DHT11, DHT22 etc.)
* Bibliotecas:
  + WiFi.h
  + AzureIoTHub.h (se usar SDK da Microsoft)
  + ou HTTPClient.h se quiser enviar via REST

**🔹 Passo 3.1 — Código de envio para IoT Hub**

**Exemplo usando HTTP simples (mais fácil para começar):**

#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

#include "DHT.h"

#define DHTPIN 4

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

const char\* ssid = "SEU\_WIFI";

const char\* password = "SENHA\_WIFI";

const char\* endpoint = "https://SEU\_WEB\_APP.azurewebsites.net/api/temperatura"; // depois ajustamos

void setup() {

Serial.begin(115200);

dht.begin();

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("\nConectado ao WiFi!");

}

void loop() {

float temp = dht.readTemperature();

if (isnan(temp)) {

Serial.println("Erro ao ler temperatura");

return;

}

if (WiFi.status() == WL\_CONNECTED) {

HTTPClient http;

http.begin(endpoint);

http.addHeader("Content-Type", "application/json");

StaticJsonDocument<200> doc;

doc["deviceId"] = "arduino-temp-01";

doc["timestamp"] = millis();

doc["temperature"] = temp;

String json;

serializeJson(doc, json);

int code = http.POST(json);

Serial.printf("HTTP %d | Dados enviados: %s\n", code, json.c\_str());

http.end();

}

delay(60000); // 1 minuto

}

⚙️ Depois substituiremos o endpoint por um **Azure Function ou API REST no App Service**.

**🗄️ ETAPA 4 – Criar o Banco de Dados (para armazenar leituras)**

**🔹 Passo 4.1 — Criar um Azure SQL Database gratuito**

1. No portal → “Create a resource” → **SQL Database**
2. Nome: TempDB
3. Grupo: IoTProjectRG
4. Crie um **Server** (ex: tempserver01) → configure usuário e senha.
5. Selecione **Basic** ou **Free Trial** tier.
6. Clique **Review + Create → Create**

**🔹 Passo 4.2 — Criar tabela**

Após criado:

1. Acesse o SQL Server → clique em “Query Editor”
2. Execute:

CREATE TABLE Temperaturas (

Id INT IDENTITY(1,1) PRIMARY KEY,

DeviceId NVARCHAR(50),

Timestamp DATETIME,

Temperatura FLOAT

);

**🌍 ETAPA 5 – Criar o site (ASP.NET Web API)**

**🔹 Passo 5.1 — Criar projeto no Visual Studio**

1. **Novo Projeto** → “ASP.NET Core Web API”
2. Nome: TempApi
3. No WeatherForecastController.cs, substitua pelo seu controlador:

[ApiController]

[Route("api/[controller]")]

public class TemperaturaController : ControllerBase

{

private readonly IConfiguration \_config;

public TemperaturaController(IConfiguration config) => \_config = config;

[HttpPost]

public async Task<IActionResult> Post([FromBody] TemperaturaDto data)

{

using var conn = new SqlConnection(\_config.GetConnectionString("DefaultConnection"));

await conn.OpenAsync();

var cmd = new SqlCommand(

"INSERT INTO Temperaturas (DeviceId, Timestamp, Temperatura) VALUES (@d, GETDATE(), @t)", conn);

cmd.Parameters.AddWithValue("@d", data.DeviceId);

cmd.Parameters.AddWithValue("@t", data.Temperature);

await cmd.ExecuteNonQueryAsync();

return Ok("Registro inserido!");

}

}

public record TemperaturaDto(string DeviceId, double Temperature);

No appsettings.json, adicione:

"ConnectionStrings": {

"DefaultConnection": "Server=tcp:tempserver01.database.windows.net,1433;Initial Catalog=TempDB;User ID=seuUsuario;Password=suaSenha;Encrypt=True;"

}

**🔹 Passo 5.2 — Publicar no Azure**

1. Clique com o botão direito no projeto → **Publicar**
2. Escolha **Azure → App Service → Create New**
3. Nomeie (ex: temp-monitor-webapp)
4. Plano: **Free (F1)**
5. Clique em **Create**
6. Após publicado, o endpoint estará disponível, ex:  
   https://temp-monitor-webapp.azurewebsites.net/api/temperatura

🔧 Use esse endpoint no Arduino no campo endpoint.

**📊 ETAPA 6 – Exibir estatísticas (página web)**

No mesmo App Service (ou outro projeto ASP.NET MVC), crie uma página que:

* Consulta as últimas medições via SQL
* Exibe gráficos (ex: **Chart.js** ou **Plotly.js**)
* Mostra **média, mínima e máxima** da temperatura

**💰 ETAPA 7 – Monitorar e otimizar custos**

1. No portal Azure → **Cost Management + Billing**
2. Defina alertas de custo (ex: R$ 0,00 limite diário)
3. Monitore o IoT Hub e o App Service (uso de mensagens e CPU)
4. Ajuste frequência de envio do Arduino para evitar excesso de dados.

**🔗 Referências oficiais (em português)**

* 📘 Microsoft Learn: Introdução ao IoT Hub
* 📘 Quickstart: enviar telemetria de dispositivo
* 📘 App Service Free Tier
* 📘 Azure SQL Database Básico