📄 Documentación de Desarrollo

Sistema de Gestión Inteligente para Help Desk IT

🎯 Objetivo:

Desarrollar un sistema que automatice y optimice la gestión de soporte técnico mediante la clasificación inteligente de tickets usando IA y ML, integración con chatbot para resolución de problemas básicos, ingreso manual de tickets y visualización de métricas en Power BI.

🧱 Arquitectura General

Capa	Tecnología
Frontend	HTML, CSS, JavaScript puro
Backend	.NET Core 9 (API RESTful)
Base de Datos	SQL Server
Dashboard	Power BI
IA/ML	Clasificación automática con ML.NET (NuGet)
Chatbot	API de ChatGPT para problemas comunes

🔁 Flujo General del Sistema

1. Usuario ingresa manualmente un ticket desde el frontend.
2. La API recibe y almacena el ticket en SQL Server.
3. Se clasifica automáticamente la prioridad usando un modelo preentrenado de ML.NET.
4. Si es un problema común, se responde automáticamente desde el chatbot (vía ChatGPT).
5. Técnicos visualizan los tickets en el dashboard de Power BI.
6. Admins pueden consultar todos los datos y estadísticas.

🔧 Backend – .NET Core API

Tecnología:

- .NET Core 9
- Paquetes NuGet:
    - Microsoft.ML
    - Microsoft.ML.TextAnalytics (opcional)
    - Microsoft.ML.DataView (opcional)

Endpoints principales:

- POST /tickets – Crear nuevo ticket (con clasificación incluida)
- GET /tickets – Listar todos los tickets
- GET /tickets/{id} – Consultar un ticket por ID
- PUT /tickets/{id} – Actualizar estado o prioridad manualmente
- POST /chatbot – Consultar respuesta con ChatGPT

🗄️ Base de Datos – SQL Server

Tablas sugeridas:

- Users (UserId, Name, Email, Role)
- Tickets (TicketId, Title, Description, Problem, Priority, Status, CreatedAt, AssignedTo, ClassifiedByML)
- ChatLogs (LogId, TicketId, Question, Answer, CreatedAt)

🧠 IA / Machine Learning – Clasificación con ML.NET (C#)

Propósito:

Asignar automáticamente prioridad Alta, Media o Baja a cada ticket.

Librería usada:

ML.NET con Microsoft.ML + modelo preentrenado con clasificación binaria que se adapta a prioridad múltiple.

Entradas del modelo:

- Title
- Description
- Problem

Se unen en un solo campo de texto para análisis (por ejemplo: "title + description + problem").

Ejemplo de uso:

csharp

CopyEdit

```
public class TicketInput
{
    public string Text { get; set; }
}

public class TicketPrediction
{
    [ColumnName("PredictedLabel")]
    public string PredictedPriority { get; set; }
}

public class PriorityClassifier
{
    private PredictionEngine<TicketInput, TicketPrediction> _engine;

    public PriorityClassifier()
```

```
    {
        var mlContext = new MLContext();
        var model = mlContext.Model.Load("PretrainedPriorityModel.zip", out _);
        _engine = mlContext.Model.CreatePredictionEngine<TicketInput,
TicketPrediction>(model);
    }

    public string ClassifyPriority(string title, string description, string problem)
    {
        var input = new TicketInput { Text = $"{title} {description} {problem}" };
        var prediction = _engine.Predict(input);
        return prediction.PredictedPriority;
    }
}
```

El modelo PretrainedPriorityModel.zip puede ser un modelo de sentimiento general adaptado a etiquetas tipo “Alta”, “Media”, “Baja”.

💬 **Chatbot con ChatGPT**

Propósito:

Responder automáticamente preguntas básicas antes de escalar a un técnico.

Implementación:

- API Key de OpenAI
- Base de conocimientos común en texto
- Lógica:
    - Usuario describe el problema
    - API hace consulta a ChatGPT
    - Si la respuesta es útil → Se muestra

- Si no, se crea el ticket

🎨 Frontend – HTML, CSS y JavaScript puro

Páginas:

- Formulario de ingreso de tickets
- Lista de tickets
- Detalles del ticket + respuesta del chatbot
- Login simple para técnicos y admins

Comunicación:

- fetch() para llamadas a la API RESTful
- Validaciones con JS puro

📊 Dashboard con Power BI

Conexión:

- Directa a SQL Server

Métricas clave:

- Tickets por prioridad
- Tickets resueltos automáticamente
- Tiempo promedio de resolución
- Actividad del chatbot

✅ Seguridad y Roles

- Autenticación con JWT
- Roles: Usuario, Técnico, Admin
- Protección de endpoints vía Authorize

📦 Deployment

Componente	Plataforma sugerida
Backend	Azure App Service / IIS
Frontend	Hosting estático o Azure Blob
Base de datos	Azure SQL / SQL Server local
Power BI	Workspace IT / Embedded opcional

📌 Consideraciones Finales

- Evaluar precisión del modelo y permitir corrección manual.
- Optimizar costos de API con OpenAI.
- Permitir reentrenamiento en el futuro si se desea mejorar la clasificación.
- Registrar logs de IA y chatbot para auditoría.