Claro, Gustavo. Aquí tienes la propuesta completa que te ofrecí inicialmente, organizada paso a paso para construir un modelo funcional del proyecto **GestUnifServ**, con integración en Microsoft Teams, backend modular, y automatización de reportes. Esta versión está adaptada para que puedas seguirla como una guía técnica clara y reproducible:

🧱 Arquitectura General del Proyecto

[Empleado → MS Teams Bot] → [Azure Bot Framework] → [API Gateway] → [Route DB] ↓ [Risk Evaluation Engine] ↓ [Analyst Review Dashboard] ↓ [Web Scraper + NLP Summary] ↓ [PDF Generator + Email Sender]



🛠️ Componentes Técnicos Clave

1. Bot en Microsoft Teams

* Construido con **Azure Bot Framework SDK** (Python o Node.js)
* Integrado en Teams vía canal de Azure
* Usa **Adaptive Cards** para capturar datos de ruta y mostrar resultados

2. Backend API

* Framework: **FastAPI** (ligero, rápido, compatible con async)
* Endpoints para:
  + Registro de rutas
  + Evaluación de riesgo
  + Revisión por analista
  + Generación de PDF
* Autenticación con **Azure Active Directory**

3. Base de Datos

* Motor: **PostgreSQL**
* Tablas:
* routes: rutas registradas
* cities: niveles de riesgo por ciudad
* decisions: revisiones de analistas
* users: empleados y roles

4. Motor de Evaluación de Riesgo

* Módulo Python que:
* Recibe lista de ciudades
* Calcula riesgo total y promedio
* Devuelve estructura JSON con detalles

5. Dashboard para Analistas

* Interfaz web básica (Flask o FastAPI + Jinja2)
* Permite revisar rutas, aprobar o rechazar
* Acceso restringido por rol

6. Scraping + NLP

* Scrapy o BeautifulSoup para extraer noticias de seguridad vial
* spaCy o HuggingFace para resumir contenido
* Cache por ruta para evitar duplicación

7. Generación de PDF + Envío

* HTML → PDF con WeasyPrint o pdfkit
* Envío por correo vía SMTP o Microsoft Graph API

📦 Estrategia de Desarrollo

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

🧭 Primeros pasos ejecutados

* Carpeta GestUnifServ creada en D:\Github
* Estructura base con src, data, notebooks, tests, docs
* Archivo README.md con descripción del proyecto
* Git inicializado y configurado con identidad
* Repositorio vinculado a GitHub

¿Quieres que avancemos ahora con el módulo risk\_evaluator.py o prefieres definir el flujo del bot en Teams? También puedo ayudarte a crear el archivo requirements.txt con las dependencias iniciales.