

ESTRUCTURA DEL FORMULARIO DE DIAGNÓSTICO PARA UNIDIGIHUB LATAM

Índice

Propósito del formulario.....	3
Principios de diseño.....	3
Contexto y alcance.....	4
Procesamiento de respuestas.....	4
Arquitectura técnica del formulario.....	4
Sección 1: Datos demográficos y socioeconómicos.....	5
Objetivos de la sección.....	5
Estructura de preguntas y campos.....	5
Metodologías clave para cumplir los 4 objetivos.....	6
Ejemplo de implementación de la pregunta 8.....	7
Sección 2: Problemáticas locales.....	8
Objetivos de la sección.....	8
Estructura de preguntas y campos.....	8
Metodologías clave para cumplir los 4 objetivos.....	9
Ejemplo de Implementación Técnica.....	10
Integración con herramientas de Google.....	10
Flujo de datos.....	10
Sección 3: Intereses profesionales.....	11
Objetivos de la sección.....	11
Estructura de preguntas y campos.....	11
Metodologías para la clasificación en 3 niveles.....	12
Metodología clave para cumplir los 4 objetivos.....	13
Integración con herramientas de Google.....	14
Sección 4: Habilidades técnicas.....	14
Objetivos de la sección.....	14
Estructura de preguntas y campos.....	15
Metodologías clave para cumplir los 4 objetivos.....	16
Integración con herramientas de Google.....	17
Sección 5: Habilidades blandas.....	17
Objetivos de la sección.....	17
Estructura de preguntas y campos.....	17
Metodologías clave para cumplir los 4 objetivos.....	19
Integración con herramientas de Google.....	20
Sección 6: Evaluación técnica adaptativa.....	20
Estrategia de implementación.....	20

Componentes clave.....	21
Herramientas de Google.....	21
Sección 7: Accesibilidad y preferencias.....	21
Estrategia de implementación.....	22
Herramientas de Google.....	22

Propósito del formulario

El Formulario de diagnóstico automatizado de UniDigiHub LATAM es una herramienta tecnológica clave para personalizar la experiencia educativa de los beneficiarios. Su objetivo principal es:

1. Evaluar habilidades técnicas, intereses profesionales y contexto socioeconómico.
2. Clasificar a los estudiantes en tres niveles de aprendizaje: UniExplorador (básico), UniCreador (intermedio) y UniVisionario (avanzado).
3. Asignar clusters sectoriales (AgriTech, FinTech, HealthTech, Energías Renovables) según problemáticas locales identificadas.
4. Activar rutas de aprendizaje personalizadas mediante IA generativa, garantizando relevancia y aplicabilidad en sus comunidades.

Principios de diseño

El formulario se estructura bajo los siguientes lineamientos técnicos y pedagógicos:

1. Dinamismo adaptativo:
 - a. Preguntas que ajustan su complejidad en tiempo real usando la Teoría de Respuesta al Ítem (TRI).
 - b. Contenido variable según sector de interés (ej: preguntas técnicas específicas para AgriTech).
2. Inclusión y accesibilidad:
 - a. Soporte multilingüe (español, portugués, inglés, lenguas indígenas) mediante Google Cloud Translation API.
 - b. Adaptaciones para discapacidades visuales, auditivas y motoras (ej: lectores de pantalla, alto contraste).
 - c. Funcionalidad offline con sincronización posterior a través de Google Cloud Storage.
3. Procesamiento inteligente:
 - a. Uso de Google Natural Language API para análisis semántico de respuestas abiertas (identificación de entidades, tono y problemáticas clave).
 - b. Integración con modelos de TensorFlow (clasificación de niveles) y XGBoost (asignación sectorial).
4. Escalabilidad y seguridad:
 - a. Almacenamiento en BigQuery para datasets estructurados y entrenamiento continuo de modelos.
 - b. Encriptación de datos con Google KMS (AES-256 en reposo, TLS 1.3 en tránsito).
 - c. Cumplimiento de normativas GDPR (UE) y LGPD (Brasil) para manejo de datos personales.

Contexto y alcance

UniDigiHub LATAM opera en comunidades rurales y semiurbanas de América Latina, donde el 8% de la población habla una lengua indígena (CEPAL, 2023). Para garantizar inclusión cultural y lingüística, el formulario de diagnóstico integra lenguas indígenas prioritarias en cada país, respetando dialectos y variantes regionales.

Lenguas Indígenas Incorporadas		
País	Lenguas Indígenas Incluidas	% Población Hablante
México	Náhuatl, Maya Yucateco, Mixteco, Zapoteco	6.1%
Colombia	Wayuunaiki, Nasa Yuwe, Emberá	1.5%
Chile	Mapudungun, Aymara, Rapa Nui	0.8%
Brasil	Guaraní, Tikuna, Yanomami	0.2%
Argentina	Quechua, Guaraní, Mapudungun	0.3%

Procesamiento de respuestas

- Modelos de NLP multilingües: Entrenamiento de BERT customizado en lenguas indígenas con datasets etiquetados por expertos.
- Detección de lengua automática: Si el usuario escribe en una lengua indígena no seleccionada, el sistema identifica la variante más cercana.
- Funcionalidades de accesibilidad
- Audio integrado: Grabaciones de preguntas en voz de hablantes nativos (ej: una mujer maya leyendo opciones en maya yucateco).
- Tecnología: Google Text-to-Speech con voces personalizadas para tonalidades indígenas.

Arquitectura técnica del formulario

plaintextCopy

```
1. Frontend (React.js + i18next):  
  - Interfaz dinámica con componentes modulares.  
  - Adaptación a dispositivos móviles y desktop.  
  
2. Backend (FastAPI + Google Cloud Functions):  
  - API REST para gestión de preguntas y respuestas.  
  - Integración con Google Cloud Translation API (traducción en tiempo real).  
  - Conexión a Firebase Auth para autenticación de usuarios.  
  
3. Procesamiento de Datos:  
  - Análisis de texto libre: Google Natural Language API → embeddings con Sentence-BERT.  
  - Clasificación de niveles: Modelo de red neuronal (TensorFlow).  
  - Clustering sectorial: Algoritmo HDBSCAN + XGBoost.  
  
4. Almacenamiento:  
  - Respuestas estructuradas: BigQuery (tablas segmentadas por país/cluster).  
  - Datos no estructurados: MongoDB (logs, respuestas de texto libre).  
  - Caché de resultados: Redis (para reducir latencia en consultas recurrentes).
```








Sección 1: Datos demográficos y socioeconómicos

Objetivos de la sección

1. Validar residencia en comunidades rurales/semiurbanas para priorizar beneficiarios.
2. Evaluar contexto socioeconómico y determinar elegibilidad para becas.
3. Identificar barreras de acceso a tecnología y educación.
4. Segmentar datos para análisis posterior (ej: correlación entre nivel educativo y habilidades técnicas).

Estructura de preguntas y campos

Pregunta	Tipo de entrada	Opciones/Detalles	Metodología aplicada
1. País	Dropdown con íconos	México, Colombia, Chile, Brasil, Argentina. Nombres mostrados en español y lenguas indígenas (ej: Mēxihco en náhuatl).	Validación geográfica con Google Maps API para filtrar comunidades rurales.
2. Departamento/Estado	Autocompletado geográfico	Sugerencias basadas en ubicación GPS o IP. Integrado con Google Maps Places API.	Datos cruzados con registros oficiales en BigQuery para priorizar zonas objetivo.
3. Municipio/Comunidad	Campo de texto + validación	Base de datos de comunidades priorizadas por UniDigiHub LATAM. Si no existe, se activa validación manual.	Almacenamiento en Firestore para futura inclusión en la base de datos.
4. Edad	Selector numérico	Rango: 15-90 años. Validación en tiempo real para excluir edades fuera del programa.	Algoritmo de detección de outliers (ej: edad <15 muestra alerta).
5. Género	Dropdown inclusivo	Opciones: - Femenino - Masculino - No binario - Prefiero no decir - Términos culturales (ej: Muxe en zapoteco).	Integración con glosarios co-creados con comunidades indígenas.

6. Nivel educativo	Dropdown con ejemplos	Opciones: - Primaria incompleta - Primaria completa - Secundaria - Técnico - Universitario - Posgrado. Ícono de diploma al seleccionar "Universitario".	Correlación con rutas de aprendizaje (ej: "Primaria incompleta" → nivel UniExplorador).
7. Situación laboral	Selector múltiple	Opciones adaptadas al contexto rural: - Agricultura de subsistencia - Empleo informal - Estudiante - Desempleado - Trabajo remoto.	Uso de XGBoost para predecir necesidad de beca según ocupación y región.
8. Acceso a tecnología	Checkbox con imágenes	Opciones: -  Teléfono móvil (sin internet) -  Teléfono con internet -  Computadora/Tablet -  Internet estable en casa -  Ninguno. Texto traducido a lenguas indígenas (ej: náhuatl).	Lógica condicional: - Si selecciona "  Ninguno" → activa descarga offline vía SMS/Radio. - Si elige "  sin internet" → prioriza cursos por WhatsApp.

Metodologías clave para cumplir los 4 objetivos

1. Validación de Residencia (Objetivo 1)

- Geofencing: Comparación de coordenadas GPS con polígonos de comunidades rurales definidos en Google Earth Engine.
- Ejemplo: Si el usuario está en una zona urbana de Monterrey, se muestra un mensaje: "Este programa prioriza comunidades rurales. ¿Resides en una?".

2. Evaluación socioeconómica (Objetivo 2)

- Modelo de elegibilidad:

python

Copy

```
def calcular_elegibilidad(respuestas):
    puntaje = 0
    # Nivel educativo bajo (+2 puntos)
    if respuestas['nivel_educativo'] in ['Primaria incompleta', 'Primaria completa']:
        puntaje += 2
    # Sin acceso a tecnología (+3 puntos)
    if respuestas['acceso_tecnologia'] == '✗ Ninguno':
        puntaje += 3
    # Zona de alta marginación (+2 puntos)
    if respuestas['municipio'] in lista_marginados:
        puntaje += 2
    return puntaje >= 5 # Aprueba beca si puntaje ≥5
```

3. Detección de Barreras Tecnológicas (Objetivo 3)

a. Priorización de recursos:

- i. Sin internet → Materiales en PDF descargables vía SMS usando Twilio API.
- ii. Solo teléfono móvil → Cursos por WhatsApp Business con chatbots en lenguas indígenas.

4. Segmentación para Análisis (Objetivo 4)

a. Dashboard en Looker Studio:

- i. Correlación entre nivel educativo y habilidades técnicas.
- ii. Mapa de calor de acceso a tecnología por región.
- iii. Tasa de becas asignadas vs. índice de pobreza local.

Ejemplo de implementación de la pregunta 8

javascript

Copy

```
// Lógica condicional para acceso a tecnología
function actualizarRecursos() {
    const acceso = document.querySelector('input[name="acceso_tecnologia"]:checked').value;
    switch (acceso) {
        case '✗ Ninguno':
            mostrarOpcionOffline(); // SMS/Radio
            break;
        case '📶 sin internet':
            activarWhatsAppCourse(); // Cursos por WhatsApp
            break;
        case '📶 sí':
            sugerirPlataformaOnline(); // Acceso completo
            break;
    }
}

// Integración con Twilio para enviar SMS con materiales
function mostrarOpcionOffline() {
    const telefono = prompt("Ingrese su número para recibir materiales por SMS:");
    fetch('https://api.twilio.com/sms', {
        method: 'POST',
        body: {
            to: telefono,
            body: 'Descarga materiales: [LINK]'
        }
    });
}
```

Sección 2: Problemáticas locales

Objetivos de la sección

1. Identificar problemas comunitarios clave que puedan abordarse con tecnologías emergentes.
2. Priorizar problemáticas según su impacto social y viabilidad técnica.
3. Vincular problemas con sectores estratégicos (AgriTech, HealthTech, etc.) para asignación de clusters.
4. Detectar urgencias que requieran intervención inmediata (ej: sequía extrema, falta de acceso a salud).

Estructura de preguntas y campos

Pregunta	Tipo de Entrada	Opciones/Detalles	Metodología Aplicada
1. Problema principal	Campo de texto guiado	Ejemplos: - "Sequía en cultivos" - "Falta de acceso a servicios de salud" - "Cortes frecuentes de energía". Sugerencias en tiempo real basadas en ubicación (ej: en Oaxaca, sugiere "sequía").	NLP con Google Natural Language API para extraer entidades (ej: "sequía" → AgriTech).
2. Relación con sectores	Selector múltiple	Clusters: - Agricultura y tecnología - Finanzas digitales - Salud comunitaria - Energía limpia. Imágenes: Íconos representativos (ej: panel solar para energías renovables).	Validación cruzada entre texto y selección manual.
3. Impacto del problema	Escala Likert + texto	Escala: 1 (Bajo impacto) – 5 (Crítico). Campo adicional: "¿Cómo afecta este problema a su comunidad?"	Análisis de sentimiento para detectar urgencia (ej: puntaje 4-5 activa alertas).
4. Soluciones intentadas	Checkbox + texto libre	Opciones: - Tecnología básica (ej: apps móviles) - Métodos tradicionales - Ninguna. Texto libre: "Describe soluciones	Clasificación con BERT multilingüe para identificar patrones (ej: "no funcionó el riego por goteo").

		fallidas/exitosas".	
5. Recursos disponibles	Selector múltiple	Opciones: - Acceso a internet - Tierra cultivable - Mano de obra - Ninguno. Imágenes: Íconos intuitivos (ej: 🌾 para tierra cultivable).	Geomatching con BigQuery para cruzar recursos con problemas locales.

Metodologías clave para cumplir los 4 objetivos

1. Identificación de problemas clave (Objetivo 1)

a. Procesamiento de Lenguaje Natural (NLP):

- Entidades y Keywords: Usa Google Natural Language API para detectar términos como "sequía", "enfermedades", o "falta de energía".
- Clustering semántico: Agrupa problemas similares usando HDBSCAN (ej: "falta de lluvia" y "cultivos secos" → cluster "sequía").

python

Copy

```
from sklearn.feature_extraction.text import TfidfVectorizer
import hdbscan

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(textos_problemas)
clusterer = hdbscan.HDBSCAN(min_cluster_size=10)
clusters = clusterer.fit_predict(X)
```

2. Priorización de problemáticas (Objetivo 2)

a. Modelo de priorización AHP:

i. Factores considerados:

- Impacto social (escala del usuario).
- Recursos disponibles (ej: sin internet → priorizar soluciones offline).
- Viabilidad técnica (según datos históricos de proyectos exitosos).

b. Salida: Lista ordenada de problemas (ej: "Sequía" > "Falta de energía" > "Acceso a salud").

3. Vinculación con sectores (Objetivo 3)

a. Modelo de asignación automática:

i. Reglas basadas en entidades:

- "Sequía" → AgriTech.
- "Enfermedades crónicas" → HealthTech.
- "Cortes de energía" → Energías Renovables.

- ii. Validación cruzada: Si el usuario menciona "sequía" pero elige el sector HealthTech, el sistema solicita confirmación.
- 4. Detección de urgencias (Objetivo 4)
 - a. Alertas en tiempo real:
 - i. Condiciones para alerta:
 - ii. Impacto ≥ 4 + sentimiento negativo detectado por NLP.
 - iii. Menció de términos como "emergencia" o "crisis".
 - b. Acción: Notificación al equipo de UniDigiHub LATAM para activar rutas aceleradas.

Ejemplo de Implementación Técnica

```
python                                                                    Copy
# Detección de urgencia con NLP
def detectar_urgencia(texto):
    client = language_v1.LanguageServiceClient()
    document = {"content": texto, "type_": "PLAIN_TEXT"}
    response = client.analyze_sentiment(request={'document': document})
    sentimiento = response.document_sentiment.score # Rango: -1 a 1
    if sentimiento < -0.5:
        return "Urgente"
    return "Normal"

# Consulta a BigQuery para problemas comunes en la región
query = f"""
SELECT problema, frecuencia
FROM `unidigihub.problemas_regionales`
WHERE municipio = '{municipio_usuario}'
ORDER BY frecuencia DESC
LIMIT 5
"""
```

Integración con herramientas de Google

1. Google Natural Language API: Extracción de entidades y análisis de sentimiento.
2. BigQuery + Looker Studio: Dashboard interactivo con mapas de calor de problemas por región.
3. Google Cloud Translation API: Traducción de respuestas en lenguas indígenas a español para procesamiento.
4. Firebase Firestore: Almacenamiento estructurado de problemas y recursos para consulta en tiempo real.

Flujo de datos








```
plaintext                                                                    Copy
1. Usuario describe problema → NLP identifica entidades → Asignación a cluster (ej: AgriTech).
2. Sistema prioriza problemas → AHP + datos históricos → Lista ordenada.
3. Urgencias detectadas → Alertas al equipo → Activación de rutas aceleradas.
4. Datos guardados en BigQuery → Análisis regional para ajustar programas educativos.
```

Sección 3: Intereses profesionales

Objetivos de la sección

1. Identificar el sector de interés principal (AgriTech, FinTech, HealthTech, Energías Renovables).
2. Determinar el nivel de aprendizaje (UniExplorador, UniCreador, UniVisionario) según experiencia y metas.
3. Vincular intereses con proyectos relevantes para su nivel.
4. Garantizar coherencia con la clasificación en 3 niveles en todas las secciones del formulario.

Estructura de preguntas y campos

Pregunta	Tipo de Entrada	Opciones/Detalles	Metodología Aplicada
1. Sector de interés	Dropdown con íconos	 AgriTech,  FinTech,  HealthTech,  Energías Renovables.	Asignación inicial al cluster, sin impacto directo en el nivel.
2. Experiencia previa	Selector de nivel + texto	Opciones: -  UniExplorador (Ninguna/baja experiencia) -  UniCreador (Experiencia básica en proyectos) -  UniVisionario (Experiencia avanzada con resultados). Campo de texto: "Describa su experiencia (ej: curso básico de IoT)".	Clave para la clasificación: - Respuestas de texto se analizan con NLP para validar el nivel autodeclarado.
3. Áreas de interés	Checkbox jerárquico	Opciones por nivel: - UniExplorador: Introducción a IoT, Conceptos básicos de blockchain. - UniCreador: Diseño de apps AgriTech, Análisis de datos en salud. - UniVisionario: Optimización de redes neuronales, Sistemas autónomos de	Las áreas se filtran según el nivel asignado en la pregunta 2.

		energía.	
4. Complejidad deseada	Slider con ejemplos	Rango: - Básico (Ej: "Aprender a usar sensores") - Avanzado (Ej: "Desarrollar un MVP escalable").	Si el usuario elige "Avanzado" pero es UniExplorador, el sistema sugiere ajustar el nivel.
5. Proyecto deseado	Campo de texto guiado	Ejemplos por nivel: - UniExplorador: "Crear un huerto con sensores básicos". - UniCreador: "Automatizar riego con Arduino". - UniVisionario: "Modelar una red inteligente de energía solar".	Comparación con proyectos históricos en BigQuery para validar viabilidad según nivel.

Metodologías para la clasificación en 3 niveles

1. Modelo de clasificación automatizada (TensorFlow)
 - a. Características de entrada: Experiencia previa (texto), áreas de interés, complejidad deseada.
 - b. Salida: Probabilidad de pertenecer a cada nivel (UniExplorador: 0-40%, UniCreador: 41-75%, UniVisionario: 76-100%).
 - c. Validación cruzada con NLP
2. Procesamiento de texto libre (experiencia y proyecto deseado):
 - a. Entidades técnicas: Uso de Google Natural Language API para detectar herramientas (ej: "Python", "TensorFlow").
 - b. Nivel inferido:
 - i. UniExplorador: Términos como "curso", "básico", "introducción".
 - ii. UniVisionario: Términos como "implementé", "escalable", "optimicé".
3. Asignación dinámica de contenido
 - a. Lógica condicional:

javascript

Copy

```
if (nivel === "UniExplorador") {
  mostrarCursos(["Introducción a AgriTech", "Uso de drones básicos"]);
} else if (nivel === "Unicreador") {
  mostrarCursos(["Desarrollo de apps IoT", "Análisis de datos con Python"]);
} else {
  mostrarCursos(["Machine Learning aplicado", "Blockchain para finanzas"]);
}
```

Metodología clave para cumplir los 4 objetivos

1. Identificar el sector de interés principal
 - a. Metodología:
 - i. Modelo XGBoost con Geomatching:
 - ii. Entrena un modelo de clasificación con datos históricos (sector, ubicación, problemáticas locales).
 - b. Variables de entrada:
 - i. Ubicación geográfica (GPS/municipio).
 - ii. Intereses declarados (ej: "agricultura sostenible").
 - iii. Problemas locales comunes (extraídos de BigQuery).
 - c. Salida: Sector asignado (AgriTech, FinTech, HealthTech, Energías Renovables).
2. Determinar el nivel de aprendizaje
 - a. Metodología:
 - i. Red Neuronal con NLP y Validación Cruzada:
 - ii. Procesamiento de Texto Libre: Usa Google Natural Language API para extraer entidades técnicas (ej: "Python", "IoT") de la experiencia descrita.
 - b. Clasificación con TensorFlow:
 - i. Entrada:
 1. Entidades técnicas detectadas.
 2. Complejidad autodeclarada (slider).
 3. Resultados de evaluación técnica (si aplica).
 - ii. Salida: Probabilidades para los 3 niveles (UniExplorador, UniCreador, UniVisionario).
 - iii. Validación cruzada: Compara el nivel autodeclarado con el predicho por el modelo. Si hay discrepancia $\geq 20\%$, se solicita revisión manual.
3. Vincular intereses con proyectos relevantes
 - a. Metodología:
 - i. Sistema de recomendación híbrido:
 1. Filtrado colaborativo: Sugiere proyectos populares en el mismo sector y nivel (ej: "Proyectos de UniCreadores en AgriTech").
 2. Filtrado basado en contenido: Usa Sentence-BERT para comparar la descripción del proyecto del usuario con una base de datos de proyectos históricos.
 - b. Geomatching automatizado:
 - i. Consulta en BigQuery los problemas frecuentes en la región del usuario.
 - ii. Prioriza proyectos que aborden esos problemas (ej: "Sistema de riego IoT" en zonas con sequía).
4. Garantizar consistencia en la clasificación
 - a. Metodología:
 - b. Centralización del nivel en firebase:
 - i. Almacena el nivel asignado en Firestore con el formato:

json

Copy

```
{
  "user_id": "123",
  "nivel": "UniCreador",
  "sector": "AgriTech",
  "fecha_clasificacion": "2025-03-15"
}
```

- c. Todas las secciones posteriores (ej: evaluación técnica, rutas de aprendizaje) consultan este dato para adaptar su contenido.
 - d. Validación transversal: Si un usuario clasificado como UniExplorador responde correctamente preguntas avanzadas en la Sección 4, el sistema sugiere recalibrar su nivel.
5. Dashboard de monitoreo (Looker Studio):
- a. Métricas clave:
 - i. % de coincidencia entre nivel autodeclarado y nivel predicho.
 - ii. Tasa de proyectos completados por nivel.
 - iii. Distribución geográfica de niveles.

Integración con herramientas de Google

1. Google AutoML Tables: Entrenamiento del modelo de clasificación con datos históricos etiquetados.
2. BigQuery: Almacenamiento de perfiles para análisis de tendencias (ej: "80% de UniVisionarios eligen HealthTech").
3. Looker Studio: Dashboard de seguimiento con métricas por nivel (ej: progreso promedio, tasa de finalización).

Sección 4: Habilidades técnicas

Objetivos de la sección

1. Evaluar competencias técnicas en áreas clave para los 4 clusters (AgriTech, FinTech, HealthTech, Energías Renovables).
2. Clasificar al usuario en uno de los 3 niveles: UniExplorador (básico), UniCreador (intermedio), UniVisionario (avanzado).
3. Identificar brechas de habilidades para personalizar rutas de aprendizaje.
4. Asignar recursos educativos adecuados al nivel y sector del usuario.

Estructura de preguntas y campos

Pregunta	Tipo de Entrada	Opciones/Detalles	Metodología
1. Autoevaluación general	Escala Likert (1-5)	Instrucción: "Califique su manejo de herramientas tecnológicas en su sector de interés". - 1: Nulo - 3: Intermedio - 5: Experto. Ejemplo visual: Emojis (😞 → 😊).	Validación cruzada con preguntas técnicas específicas.
2. Herramientas utilizadas	Checkbox dinámico	Opciones por cluster: - AgriTech: Sensores IoT, drones, software de análisis agrícola (ej: FarmBot). - FinTech: Plataformas blockchain (ej: Ethereum), APIs de pagos (ej: Stripe). - HealthTech: Sistemas de telemedicina (ej: Zoom for Healthcare), wearables. - Energías: Software de simulación (ej: PVsyst), herramientas de gestión de redes. Imágenes: Logotipos de herramientas populares.	Asignación de peso según complejidad (ej: "TensorFlow" = +2 puntos para UniVisionario).
3. Ejercicio práctico	Preguntas adaptativas	Ejemplos por nivel: - UniExplorador: "¿Qué es un sensor IoT?" (opción múltiple). - UniCreador: "Describa cómo configuraría un sistema de riego con Arduino" (texto breve). - UniVisionario: "Proponga un algoritmo para optimizar el consumo energético en una red solar" (texto libre). Nota: Las preguntas se activan según autoevaluación inicial.	NLP con BERT multilingüe para analizar respuestas abiertas y asignar nivel.

4. Certificaciones	Selector + verificación	Opciones: - Cursos en línea (ej: Coursera, edX). - Certificaciones técnicas (ej: AWS, Cisco). - Ninguna. Si selecciona una opción: Campo para subir certificado (PDF/IMG) o enlace.	Reconocimiento de texto (Google Vision AI) para validar certificaciones.
5. Proyectos realizados	Campo de texto guiado	Guía: "Describa un proyecto técnico que haya ejecutado (ej: app móvil, sistema de monitoreo)". Sugerencias: - UniExplorador: "Usé una plantilla para crear una app básica". - UniVisionario: "Desarrollé una red blockchain para transacciones locales".	Extracción de entidades técnicas (Google NLP) para identificar habilidades (ej: "Python", "TensorFlow").

Metodologías clave para cumplir los 4 objetivos

1. Evaluación de competencias (Objetivo 1)

a. Modelo de puntuación compuesta:

```
python
def calcular_puntaje(respuestas):
    # Puntos por herramientas usadas
    herramientas = {"FarmBot": 1, "TensorFlow": 3, "Blockchain": 2}
    puntaje = sum([herramientas[herramienta] for herramienta in respuestas['herramientas']])

    # Puntos por certificaciones
    if respuestas['certificaciones']:
        puntaje += 2

    # Puntos por ejercicio práctico (NLP)
    complejidad = modelo_bert.predict(respuestas['ejercicio'])[0]
    puntaje += complejidad * 2

    return puntaje
```

b. Clasificación:

- i. 0-10 puntos → UniExplorador.
- ii. 11-20 puntos → UniCreador.
- iii. 21+ puntos → UniVisionario.

2. Clasificación en niveles (Objetivo 2)

a. Lógica adaptativa:

- i. Si el usuario selecciona "TensorFlow" en herramientas pero su ejercicio práctico es básico, el sistema sugiere UniCreador y recomienda cursos intermedios.
- ii. Regla de congruencia: Si hay discrepancia $\geq 30\%$ entre autoevaluación y ejercicio práctico, se activa una revisión manual.

3. Identificación de brechas (Objetivo 3)

a. Mapa de brechas:

- i. Técnica: Comparar herramientas usadas vs. herramientas requeridas en el sector (ej: AgriTech requiere IoT, falta en respuestas → brecha detectada).
- ii. Visualización: Dashboard en Looker Studio con gráficos de radar (habilidades actuales vs. ideales).

4. Asignación de recursos (Objetivo 4)

a. Recomendación personalizada:

- i. UniExplorador: Cursos introductorios en YouTube/Coursera + kits de IoT básicos.
- ii. UniCreador: Bootcamps prácticos (ej: "Desarrollo de apps AgriTech con Arduino").
- iii. UniVisionario: Mentorías con expertos + acceso a laboratorios avanzados.

Integración con herramientas de Google

1. Google Cloud Natural Language: Extracción de entidades técnicas y análisis de complejidad en respuestas abiertas.
2. Google Vision AI: Validación de certificaciones subidas (OCR para detectar nombres de instituciones).
3. Firebase Firestore: Almacenamiento de perfiles técnicos para acceso en tiempo real.
4. BigQuery + Looker Studio: Reportes de distribución de habilidades por región y sector.

Sección 5: Habilidades blandas

Objetivos de la sección

1. Evaluar competencias blandas clave para el trabajo en equipo y liderazgo.
2. Clasificar al usuario en los niveles UniExplorador, UniCreador o UniVisionario según su perfil socioemocional.
3. Identificar áreas de mejora para fortalecer habilidades como comunicación, resolución de conflictos y adaptabilidad.
4. Asignar recursos educativos y actividades autogestionadas o colaborativas según el nivel y cluster sectorial.

Estructura de preguntas y campos

Pregunta	Tipo de Entrada	Opciones/Detalles	Metodología
----------	-----------------	-------------------	-------------

1. Autoevaluación	Escala Likert (1-5)	<p>Habilidades:</p> <ul style="list-style-type: none"> - Trabajo en equipo - Comunicación efectiva - Adaptabilidad - Resolución de conflictos - Liderazgo. <p>Ejemplo: "¿Cómo califica su capacidad para trabajar en equipo?" (1: Muy baja – 5: Excelente).</p>	Validación cruzada con preguntas situacionales para evitar sesgos de autopercepción.
2. Situaciones hipotéticas	Selección de respuestas	<p>Escenarios por cluster:</p> <ul style="list-style-type: none"> - AgriTech: "Su equipo discute métodos de riego. ¿Cómo lidera la decisión?" - FinTech: "Un compañero comete un error en datos financieros. ¿Cómo lo aborda?" <p>Opciones:</p> <ul style="list-style-type: none"> - "Impongo mi solución" (UniExplorador) - "Busco consenso" (UniCreador) - "Analizo causas y propongo mejora sistémica" (UniVisionario). 	Ponderación por nivel: Cada opción suma puntos específicos al perfil.
3. Experiencias pasadas	Campo de texto guiado	<p>Guía: "Describa un conflicto que resolvió en su comunidad o trabajo".</p> <p>Ejemplos:</p> <ul style="list-style-type: none"> - UniExplorador: "Medié entre dos vecinos". - UniVisionario: "Organicé un grupo para limpiar el río local". 	NLP con BERT multilingüe para detectar habilidades implícitas (ej: "organización" → liderazgo).
4. Preferencia de roles	Selector visual	<p>Imágenes:</p> <ul style="list-style-type: none"> - 👤 Colaborador (UniExplorador) - 🎯 Coordinador (UniCreador) - 🌟 Líder (UniVisionario). <p>Pregunta: "¿Qué rol suele asumir en proyectos grupales?"</p>	Correlación con nivel: Roles avanzados sugieren UniVisionario.

5. Recursos preferidos	Checkbox dinámico	Opciones: - Cursos en línea - Foros de discusión - Proyectos comunitarios - Guías prácticas descargables. Nota: Las opciones se filtran por nivel (ej: UniExplorador ve cursos básicos).	Recomendación basada en nivel y cluster sectorial.
------------------------	-------------------	---	--

Metodologías clave para cumplir los 4 objetivos

1. Evaluación de competencias blandas (Objetivo 1)

a. Modelo de Puntuación Compuesta:

```
python Copy
def calcular_puntaje(respuestas):
    # Puntos por situaciones hipotéticas (ej: opción UniVisionario = +3 puntos)
    puntos_situaciones = sum([opcion.puntos for opcion in respuestas['situaciones']])

    # Puntos por autoevaluación (Likert promedio)
    puntos_autoevaluacion = sum(respuestas['likert']) / len(respuestas['likert'])

    # Puntos por NLP en experiencias pasadas (detectar "organización" → +2)
    habilidades_nlp = analizar_texto(respuestas['experiencias'])
    return puntos_situaciones + puntos_autoevaluacion + habilidades_nlp
```

2. Clasificación en niveles (Objetivo 2)

a. Umbrales:

- UniExplorador: 0-10 puntos (respuestas pasivas, autoevaluación ≤ 3).
- UniCreador: 11-20 puntos (busca consenso, autoevaluación 3-4).
- UniVisionario: 21+ puntos (liderazgo sistémico, autoevaluación ≥ 4.5).

3. Identificación de brechas (Objetivo 3)

a. Mapa de brechas:

- Técnica: Comparar autoevaluación con desempeño en escenarios (ej: alto autocalificado en liderazgo pero elección pasiva → brecha detectada).
- Visualización: Gráficos de radar en Google Data Studio para mostrar fortalezas y debilidades.

4. Asignación de recursos (Objetivo 4)

a. Recomendaciones autogestionadas:

- UniExplorador:

1. Cursos en línea básicos (ej: "Introducción al trabajo en equipo" en YouTube).
2. Guías descargables para resolver conflictos cotidianos.
- ii. UniCreador:
 1. Foros de discusión moderados por pares (ej: Grupos de WhatsApp por cluster).
 2. Talleres virtuales con metodologías ágiles (ej: "Design Thinking para AgriTech").
- iii. UniVisionario:
 1. Acceso a proyectos comunitarios certificados (ej: Liderar una iniciativa de energía solar en su localidad).
 2. Cursos avanzados en Coursera/edX (ej: "Liderazgo en Innovación Sostenible").

Integración con herramientas de Google

1. Google Natural Language API: Detección de habilidades en respuestas de texto (ej: "mediación" → resolución de conflictos).
2. Firebase Firestore: Almacenamiento de preferencias de recursos y niveles.
3. Google Sheets + Apps Script: Automatización de envío de recursos por correo/SMS.
4. Google Looker Studio: Dashboard de progreso en habilidades blandas por región y sector.

Sección 6: Evaluación técnica adaptativa

Objetivo: Ajustar dinámicamente la dificultad de las preguntas técnicas según las respuestas del usuario, utilizando la Teoría de Respuesta al Ítem (TRI) para precisar su nivel real de competencia.

Estrategia de implementación

1. Modelo IRT: Se puede implementar un modelo IRT (por ejemplo, un modelo de dos parámetros o Rasch) que estime la habilidad del usuario y la dificultad de cada ítem. Con cada respuesta, el modelo recalcula la probabilidad de acierto y ajusta la selección de la siguiente pregunta para maximizar la información sobre la habilidad del examinado.
2. Flujo adaptativo:
 - a. Registro de respuestas: Cada respuesta se almacena (por ejemplo, en Firestore o BigQuery) para análisis en tiempo real.
 - b. Cálculo de probabilidad: Se puede usar una función logística para modelar la probabilidad de respuesta correcta, actualizando la estimación del parámetro de habilidad del usuario en cada iteración.

- c. Selección de ítems: Con base en la información de la respuesta, se selecciona el siguiente ítem cuyo parámetro de dificultad se encuentre cercano al nivel de habilidad estimado del usuario.

Componentes clave

1. Mecanismo de adaptación:
 - a. TRI aplicada:
 - i. Cada pregunta tiene parámetros predefinidos (dificultad, discriminación, azar).
 - ii. El sistema estima la habilidad del usuario en tiempo real. Ejemplo:
 - iii. Si responde correctamente una pregunta de dificultad media, la siguiente será más compleja.
 - iv. Si falla, se reduce la complejidad para evitar frustración.
 - b. Tipos de Preguntas:
 - a. Preguntas técnicas por sector:
 - i. AgriTech: "Calcule la eficiencia de riego con datos de humedad del suelo".
 - ii. HealthTech: "Interprete un conjunto de datos de pacientes con diabetes".
 - b. Formato adaptativo:
 - i. Opción múltiple para UniExplorador.
 - ii. Texto libre/código para UniVisionario.
3. Integración con IA:
 - a. TensorFlow + TRI:
 - i. Entrena modelos para predecir la habilidad del usuario y asignar preguntas óptimas.
 - b. Validación cruzada:
 - i. Compara resultados con la autoevaluación de la Sección 4 para detectar inconsistencias.

Herramientas de Google

1. Google Cloud Functions: Para ejecutar la lógica de adaptación en tiempo real en el backend.
2. BigQuery/Firestore: Para almacenar y consultar los datos de respuestas y parámetros del modelo.
3. Bibliotecas en Python: Utilizar bibliotecas estadísticas (como scipy o statsmodels) o incluso TensorFlow para ajustar modelos probabilísticos basados en TRI.
4. Firebase Realtime Database: Almacenamiento de respuestas y ajuste dinámico de la secuencia de preguntas.

Sección 7: Accesibilidad y preferencias

Objetivo: Garantizar la inclusión y adaptación a las necesidades de todos los usuarios.

Estrategia de implementación

1. Soporte multilingüe:
 - a. Utilizar la Google Cloud Translation API para traducir dinámicamente el contenido del formulario a múltiples idiomas (por ejemplo, español, portugués, inglés y lenguas indígenas). Esto permite que el contenido se adapte a la lengua del usuario, aumentando la inclusión.
2. Adaptaciones para discapacidades:
 - a. Lectores de pantalla y Text-to-Speech: Integrar la Google Cloud Text-to-Speech API para generar audio de las preguntas, de modo que los usuarios con discapacidad visual puedan escuchar el contenido.
 - b. Contraste y adaptaciones visuales: Asegurarse de que la interfaz del formulario (por ejemplo, desarrollada en un framework web compatible con Python como Flask o Django) cumpla con estándares de accesibilidad (WCAG), implementando opciones de alto contraste y navegación mediante teclado.
3. Preferencias de recursos:
 - a. Permitir que el usuario seleccione preferencias (por ejemplo, modo visual, auditivo o mixto) y ajustar la interfaz y el contenido en función de estas opciones. Esto puede incluir la adaptación de formatos (texto ampliado, interfaces simplificadas, etc.) y el uso de la Google Cloud Storage para la sincronización offline y posterior actualización.

Herramientas de Google

1. Google Accessibility Scanner: Para testear y optimizar la interfaz.
2. Firebase Remote Config: Ajustar preferencias de usuario sin actualizar la app.
3. Google Workspace: Integración con Forms para plantillas accesibles.