Examen práctico de Analista de Desarrollo

Gerencia de Innovación

Nombre aspirante:

Teléfono:

**Contestar las siguientes preguntas:**

1. **Bases de datos:**
   1. **¿Qué hace un left join?**
      1. Devuelve todos los registros de la tabla izquierda y los coincidentes de la derecha; los no coincidentes muestran NULL.
   2. **¿Qué hace un inner join?**
      1. Devuelve solo los registros que tienen coincidencia en ambas tablas.
   3. **¿Cuál es la diferencia entre el cast y convert?**
      1. Ambos convierten tipos de datos; CAST es estándar SQL, CONVERT es específico de SQL Server y permite formato adicional.
   4. **¿Para qué sirve un índice?**
      1. Optimiza las búsquedas y mejora el rendimiento de las consultas al acelerar la lectura.
   5. **¿Para qué sirve un Store Procedure?**
      1. Encapsula lógica SQL reutilizable en el servidor para ejecutar operaciones de forma controlada y eficiente.
   6. **¿Cuál es la diferencia entre un índice Clusteriado y no clusteriado?**
      1. Clustered ordena físicamente los datos de la tabla; No Clustered crea una estructura separada con punteros a los datos.
   7. **¿Para qué sirve una tabla temporal y como se declara?**
      1. Guarda datos temporales en una sesión, así: CREATE TABLE #temp (columna1 tipo, columna2 tipo);
   8. **¿Para qué sirve una tabla doble temporal y como se declara?**
      1. Es global y accesible por múltiples sesiones: CREATE TABLE ##temp (columna1 tipo, columna2 tipo);
   9. **¿Para qué sirve la función set dateformat dmy?**
      1. Define el formato de fecha como Día-Mes-Año para interpretar correctamente las fechas.
   10. **Escriba la sintaxis de un cursor:**

DECLARE cursor\_name CURSOR FOR

SELECT columnas FROM tabla;

OPEN cursor\_name;

FETCH NEXT FROM cursor\_name INTO @variable;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- operaciones

FETCH NEXT FROM cursor\_name INTO @variable;

END

CLOSE cursor\_name;

DEALLOCATE cursor\_name;

* 1. **¿Diferencia entre un truncate table y un drop table?**
     1. TRUNCATE borra todos los registros sin eliminar la tabla; DROP elimina la tabla completa del esquema.
  2. **¿Indique una herramienta para ver el performance de un gestor de base de datos y que encontramos?**
     1. SQL Server Profiler o Performance Monitor; permiten analizar consultas, tiempos de ejecución, bloqueos e índices utilizados.

2 Backend:

**2.1 Explica cómo implementarías una estrategia de cacheo a nivel de aplicación sin utilizar servicios externos (Se puede apoyar de una base de datos relacional como Postgresql o sql server al ser un escenario común donde no se cuenta con la disponibilidad de usar servicios de alguna base de datos dictionary in memory como Redis):**

Crear tabla cache con campos clave, valor, fecha\_expiracion. Consultar primero la tabla; si el dato expira, regenerar y actualizar. Implementar capa en memoria (Dictionary estático) con TTL controlado.

**2.2 Explica el concepto de versionado de APIs y cómo lo implementarías.**

Permite coexistencia de múltiples versiones. Se implementa con prefijo en URL (/api/v1/), en encabezado (Accept-Version) o mediante rutas controladas en el backend.

**2.3 ¿Qué son los endpoints idempotentes y por qué son importantes?**

Operaciones que producen el mismo resultado sin importar cuántas veces se ejecuten. Son importantes para consistencia y tolerancia a reintentos.

**2.4 ¿Qué es la inyección SQL y cómo prevenirla?**

Ataque que inserta código SQL malicioso. Se previene usando consultas parametrizadas, ORM, validación de entrada y mínimo privilegio en la base de datos.

**2.5 Explica cómo implementar autenticación y autorización en una aplicación backend (incluyendo como manejarías el almacenamiento seguro de contraseñas) para evaluar su conocimiento de encriptación simétrica y asimétrica.**

Usar JWT o OAuth2; contraseñas almacenadas con hash seguro, como, bcrypt o Argon2 y salt.

Cifrado simétrico (AES) para datos sensibles; asimétrico (RSA) para intercambio seguro de claves y tokens.

**2.6 ¿Cómo identificarías y resolverías un cuello de botella en una aplicación backend?**

Monitorear métricas (CPU, RAM, I/O, latencia, logs, APM).

Optimizar consultas, usar cache, asincronía, balanceo de carga o mejorar concurrencia según el origen del problema.

**2.7 Explica las diferencias entre concurrencia y paralelismo y cómo afectan el diseño de aplicaciones backend.**

La concurrencia es cuando múltiple tareas van progresando alternadamente, de manera async. El paralelismo es cuando las tareas se ejecutan de manera simultánea en varios núcleos. Ambos afectan el diseño, en la parte del manejo de recursos compartidos y eficiencia del backend.

**2.8 ¿Cómo manejarías la sincronización de hilos en un entorno multihilo para evitar condiciones de carrera?**

Usar locks, semáforos, mutex o monitores para proteger secciones críticas. Evitar deadlocks usando jerarquía y mínimo tiempo de bloqueo.

**2.9 ¿Cuáles son los problemas asociados con el uso de números de punto flotante en aplicaciones financieras? Proporcione ejemplos específicos.**

La perdida de precisión, por ejemplo, 0.1+0.2 no es igual a 0.3. Se tiene que usar DECIMAL o NUMERIC, para cálculos exactos y evitar errores al momento de tener que redondear números.

**2.10 ¿Cómo afecta el uso excesivo de bloques try-catch y la implementación de recursión profunda al rendimiento de una aplicación backend? En tu respuesta, analiza el impacto en el uso de la CPU y la memoria RAM, y explica los conceptos de ciencias de la computación que sustentan estos efectos.**

El uso excesivo de los try y catch aumentan el overhead y degradan el CPU. La recursión profunda consume pila, teniendo presente stack overflow y RAM. Afectan el rendimiento por mayor uso de memoria y los ciclos de CPU, lo mejor es iteración y manejo centralizado de los errores.

1. Desarrollo web y mobile
   1. **¿Cuáles son las principales diferencias entre el Renderizado del Lado del Servidor (SSR) y el Renderizado del Lado del Cliente (CSR)? Analiza sus ventajas y desventajas en términos de rendimiento, SEO y experiencia de usuario.**

El renderizado del lado del servidor (SSR) genera el HTML completo en el servidor, lo que mejora el SEO y reduce el tiempo de carga inicial, aunque aumenta la carga en el servidor. El renderizado del lado del cliente (CSR) depende del navegador para construir la interfaz, ofreciendo una experiencia más dinámica pero con peor rendimiento inicial y menor indexación.

* 1. **Explica qué es el Critical Rendering Path en el contexto del desarrollo frontend y describe las técnicas avanzadas que utilizarías para optimizarlo, reduciendo el tiempo de carga percibido por el usuario.**

El Critical Rendering Path es la secuencia que sigue el navegador para convertir el HTML, CSS y JS en píxeles visibles. Se optimiza minificando recursos, aplicando defer o async en scripts, usando CSS crítico inline, lazy loading de imágenes y priorizando el contenido visible.

* 1. **Describe cómo implementarías un middleware personalizado en cualquier gestor de estados (Redux o Zustand) para manejar efectos secundarios complejos, como llamadas a APIs externas y gestión de autenticación, asegurando que el flujo de datos permanezca predecible y mantenible.**
     1. Un middleware personalizado en Redux o Zustand intercepta las acciones antes de llegar al estado. Permite manejar efectos secundarios como llamadas a APIs o autenticación, ejecutando la lógica asíncrona y luego despachando acciones de éxito o error para mantener un flujo predecible.
  2. **Explica las diferencias entre REST y GraphQL en el contexto del desarrollo frontend y describe cómo implementarías una solución híbrida que aproveche las ventajas de ambos enfoques.**
     1. REST usa múltiples endpoints con estructuras fijas, mientras que GraphQL utiliza uno solo que permite solicitar exactamente los datos necesarios. Una solución híbrida combina REST para operaciones simples y GraphQL para consultas complejas mediante un gateway unificado.
  3. **Explica los conceptos de Progressive Enhancement y Graceful Degradation en el desarrollo frontend y cómo los aplicarías para asegurar una experiencia de usuario consistente en diferentes navegadores y dispositivos.**
     1. Progressive Enhancement parte de una base funcional y añade mejoras en navegadores modernos, mientras que Graceful Degradation asegura que una app avanzada siga funcionando en entornos limitados. Ambas garantizan una experiencia consistente en distintos dispositivos.
  4. **Explica la importancia de la accesibilidad en el desarrollo y describe las mejores prácticas para asegurar que una aplicación web o mobile sea accesible para todos los usuarios, incluyendo aquellos con discapacidades o poca educación digital (usuarios de escasos recursos con dispositivos demasiado antiguos y con los que les cuesta interactuar).**
     1. La accesibilidad es esencial para incluir a todos los usuarios. Se logra usando etiquetas semánticas, texto alternativo, buen contraste, roles ARIA, navegación por teclado y diseño adaptable, facilitando el uso incluso en dispositivos antiguos o por personas con discapacidad.
  5. **Describe las Core Web Vitals (LCP, FID y CLS) y explica cómo optimizarías una aplicación frontend para cumplir con estos indicadores de rendimiento.**
     1. Se optimizan mejorando la carga de imágenes y fuentes, reduciendo scripts bloqueantes y evitando saltos de diseño. Cumplir con ellas mejora el rendimiento y la percepción del usuario.
  6. **En el contexto de desarrollar un chat frontend que interactúa con una inteligencia artificial (IA) a través de una API de un modelo de lenguaje grande (LLM), ¿qué tecnología utilizarías para manejar la comunicación en tiempo real: WebSockets o Server-Sent Events (SSE)? Detalla los beneficios de la tecnología seleccionada y explica cómo gestionarías la conexión, el envío y la recepción de mensajes**
     1. La mejor opción es Server-Sent Events (SSE), ya que permite streaming continuo del modelo hacia el cliente con menor consumo que WebSockets. Se gestiona abriendo una conexión persistente y actualizando la interfaz conforme llegan los mensajes.

**4. Inteligencia Artificial**

**4.1 ¿Cuáles son los principales algoritmos de aprendizaje superficial (shallow learning) y en qué casos usarías cada uno?**  
Los principales son regresión lineal, regresión logística, KNN, SVM, árboles de decisión y Naive Bayes. Se usan según el tipo de problema: regresión para valores continuos, clasificación para etiquetas, y SVM o KNN cuando los límites no son lineales.

**4.2 ¿Qué es la regresión lineal y cuándo la emplearías?**  
Es un modelo que estima relaciones entre variables independientes y una dependiente continua; se usa para predicciones numéricas como ventas o precios.

**4.3 ¿Cómo funciona un árbol de decisión y cuál es la diferencia entre criterio Gini y entropía?**  
Divide los datos por condiciones en nodos. Gini mide impureza y entropía mide incertidumbre; ambos evalúan la calidad de las divisiones.

**4.4 ¿En qué se diferencian un SVM con kernel lineal y uno con kernel RBF?**  
El SVM lineal separa datos con un plano recto, el RBF crea límites no lineales mapeando los datos a un espacio de mayor dimensión.

**4.5 ¿Qué efecto tiene limitar la profundidad de un árbol de decisión?**  
Reduce el sobreajuste, mejora la generalización y disminuye la varianza, aunque puede aumentar el sesgo.

**4.6 ¿En qué se diferencian bagging y boosting, y cómo reduce cada uno el error del ensemble?**  
Bagging entrena modelos independientes y promedia resultados (reduce varianza). Boosting los entrena secuencialmente corrigiendo errores (reduce sesgo).

**4.7 ¿Cómo elegirías el número de clusters en KMeans usando el método del codo y la silueta?**  
Con el método del codo se elige donde la inercia deja de bajar bruscamente; la silueta mide qué tan bien están separados los clusters.

**4.8 Define la matriz de confusión y explica cómo se relacionan precisión, recall y la probabilidad posterior de Bayes.**  
La matriz compara predicciones con valores reales. Precisión mide exactitud positiva, recall mide cobertura, y ambas reflejan probabilidades posteriores según Bayes.

**4.9 ¿En qué consiste un modelo generativo estocástico como Naive Bayes y cómo difiere de un enfoque discriminativo?**  
Modela la distribución conjunta de clases y características. A diferencia del discriminativo, aprende cómo se generan los datos y no solo la frontera entre clases.

**4.10 ¿Cuál es el propósito de la regularización L1 vs L2 y cómo afecta a los coeficientes del modelo?**

L1 genera coeficientes cero (selección de variables) y L2 los reduce de forma uniforme (estabilidad). Ambas evitan sobreajuste penalizando pesos grandes.

**4.11 ¿Qué arquitecturas de redes (CNN, RNN, Transformer) se usan típicamente en visión, sonido y NLP, y qué ventajas ofrece cada una?**  
CNN en visión (patrones espaciales), RNN en audio o texto (secuencias) y Transformers en NLP (atención paralela y contexto global).

**4.12 ¿Cuál es el propósito de una función de activación y por qué ReLU es tan popular?**  
Introduce no linealidad en la red. ReLU es popular por su eficiencia y por evitar el problema del gradiente desvanecido.

**4.13 ¿Cómo actúa el dropout para prevenir el sobreajuste en redes profundas?**  
Desactiva aleatoriamente neuronas durante el entrenamiento, evitando dependencias entre ellas y mejorando la generalización.

**4.14 ¿En qué se diferencian las convoluciones 1D, 2D y 3D y sus aplicaciones típicas?**  
1D para secuencias, 2D para imágenes y 3D para video o volúmenes. Cada una analiza patrones locales en diferentes dimensiones.

**4.15 Explica la atención escalada en Transformers y por qué su coste es O(n²).**  
Compara cada token con todos los demás para calcular relevancia; su coste es O(n²) porque realiza comparaciones entre todos los pares.

**4.16 ¿Cómo se estructuran y relacionan forward y backpropagation durante el entrenamiento de una red neuronal? ¿Qué es el descenso de gradiente y por qué es fundamental?**  
Forward calcula las salidas aplicando pesos y activaciones; backpropagation propaga errores y ajusta pesos con descenso de gradiente, que minimiza la función de pérdida moviéndose en dirección contraria al gradiente.

**4.17 ¿Qué papel juegan las convoluciones en visión por computador y cómo se comparan con Vision Transformers? ¿Ventajas de usar AWS Rekognition frente a modelos propios?**  
Las convoluciones extraen rasgos locales; los Vision Transformers usan atención global. AWS Rekognition es rápido y escalable, pero los modelos propios ofrecen control y personalización.

**4.18 ¿Qué técnicas ofrece OpenCV para mejorar imágenes degradadas y cómo se comparan con redes neuronales?**  
Usa filtros Gaussianos, bilaterales y CLAHE para mejorar contraste y reducir ruido. Las redes neuronales logran mejores resultados pero requieren más cómputo.

**4.19 ¿Qué es la calibración de cámara en OpenCV, por qué es importante y cómo se realiza?**  
Estima parámetros internos y externos usando un patrón de ajedrez. Es clave en visión estereoscópica y reconstrucción 3D precisa.

**4.20 ¿Cómo estimar el estilo artístico de una imagen y qué técnicas se usan en style transfer?**  
Se emplean Gram Matrix y redes VGG para capturar textura y estilo. Tienen alto costo y poco realismo; modelos modernos como AdaIN o Diffusion mejoran eficiencia y resultados.

**4.21 ¿Qué retos presenta la visión por computador en dispositivos móviles o embebidos y cómo optimizar OpenCV?**  
Limitaciones de energía, CPU y RAM. Se optimiza con modelos comprimidos (quantization, pruning) y uso de aceleración por GPU o NEON.

**4.22 ¿Cómo influye el diseño de prompts (zero-shot, few-shot, chain-of-thought, tool-aware) en LLMs?**  
Determina su capacidad de razonamiento y precisión. Zero-shot da instrucciones directas, few-shot provee ejemplos, chain-of-thought guía pasos y tool-aware permite interactuar con herramientas externas.

**4.23 ¿Qué define a un agente en IA y en qué se diferencia de un modelo de ML?**  
Un agente actúa autónomamente con memoria y metas; un modelo de ML solo predice sin capacidad de decisión continua.

**4.24 ¿Qué impacto tiene el uso de etiquetas XML en los prompts sobre el comportamiento del modelo?**  
Estructuran el contenido y delimitan roles, mejorando interpretación y coherencia de las respuestas.

**4.25 Diseña un prompt largo “sandwich” (goal–persona–context–goal) para un LLM.**  
Goal: crear plan de desarrollo. Persona: arquitecto senior. Context: proyecto web con Flask y React. Goal final: generar entregables y responsables.

**4.26 ¿Cómo implementar autenticación y control de acceso en un servidor MCP?**  
Usar JWT para autenticación y control por roles. Cifrar contraseñas con bcrypt y datos sensibles con AES y RSA.

**4.27 ¿Qué técnicas de razonamiento puedes incorporar en un agente para que justifique sus decisiones?**  
Chain of Thought, Tree of Thoughts y State of Thought, que estructuran el razonamiento paso a paso o jerárquicamente.

**4.28 ¿Cuáles son los componentes básicos de un agente?**  
Perfil, memoria, razonamiento, herramientas, planificación y evaluación.

**4.29 ¿Cómo funciona un agente autónomo frente a uno asistente o proxy?**  
El autónomo actúa sin intervención humana; el asistente depende de instrucciones o aprobación del usuario.

**4.30 ¿Cómo diseñar un sistema multiagente colaborativo?**  
Cada agente cumple una función (planificación, búsqueda, memoria) y se comunican por colas o APIs. Se coordinan mediante un orquestador o bus de eventos.

**4.31 ¿Qué patrones de coordinación existen entre agentes y cuándo usarlos?**  
Broadcast para difusión general, pipeline para tareas secuenciales y blackboard para colaboración mediante memoria compartida.

**4.32 ¿Cómo integras feedback humano para mejorar decisiones de agentes?**  
Mediante corrección manual y ajuste de recompensas, reforzando comportamientos deseados.

**4.33 ¿Cómo evalúas el rendimiento y efectividad de un sistema multiagente?**  
Con KPIs, trazabilidad, tiempos de respuesta y precisión en las decisiones.

**4.34 Imagina un sistema de IA para emergencias en una ciudad inteligente. ¿Cómo decidirías su arquitectura?**  
Usaría agentes especializados coordinados jerárquicamente, comunicación en tiempo real y priorización por tipo de incidente.

**4.35 ¿Qué papel juegan los “agentic behavior trees” y los “policy planners”?**  
Los behavior trees estructuran conductas jerárquicas; los policy planners generan planes óptimos ante diferentes estados y objetivos.

**4.36 ¿Qué criterios considerarías al elegir un framework para agentes inteligentes y cuál elegirías?**  
Escalabilidad, soporte multimodal e integración. Preferiría LangGraph o Agno por su flexibilidad y colaboración entre agentes.

**4.37 ¿Qué ventajas ofrece el uso de structured outputs en la interacción entre agentes o herramientas?**  
Permiten respuestas en JSON o Pydantic validadas automáticamente, reduciendo errores y mejorando interoperabilidad.

**4.38 ¿Cómo implementar un agente con refuerzo usando ART?**  
Registrar interacciones, asignar recompensas, guardar trayectorias y actualizar el modelo iterativamente según desempeño.

**4.39 ¿Cómo garantizar reproducibilidad y auditoría de decisiones de un agente?**  
Registrando cada interacción con APIs, entradas, salidas, contexto y timestamp en logs persistentes.

**4.40 ¿Qué es un tokenizer y por qué es importante?**  
Divide texto en tokens; afecta cómo el modelo procesa entradas, el límite de longitud y el costo computacional.

**4.41 ¿Qué estrategias se usan para reducir tokens sin perder semántica?**  
Resumir texto, eliminar redundancias, usar sinónimos cortos y simplificar frases manteniendo el significado.

**4.42 ¿Cuáles son los beneficios de una memoria persistente en agentes y qué desafíos presenta?**  
Mantiene contexto, personaliza respuestas y se adapta con el tiempo. Los desafíos son coherencia y datos dinámicos; se soluciona con memorias vectoriales como Zep AI.

**4.43 ¿Qué ventajas aporta el Model Context Protocol (MCP) frente a function calling?**  
Estandariza comunicación entre modelos y herramientas, permite contexto persistente y mejor interoperabilidad.

**4.44 ¿Cómo diseñarías un agente que lea documentos y extraiga imágenes relevantes?**  
Usaría OCR y embeddings visuales para identificar imágenes y devolver nombre de archivo con su contexto textual.

**4.45 ¿Qué técnicas de caching y persistencia usarías para evitar reprocesar archivos vectorizados?**  
Hashing de contenido y base persistente de embeddings; si el hash ya existe, se reutiliza.

**4.46 ¿Cómo estructurar y almacenar documentos para un sistema RAG?**  
Fragmentarlos en chunks con metadatos, vectorizarlos y almacenarlos en una base semántica como FAISS o Milvus.

**4.47 ¿Cómo combinar búsqueda semántica con reranking para mejorar resultados de RAG?**  
Primero aplicar similitud vectorial y luego reranking con un modelo cross-encoder para mayor precisión.

**4.48 ¿Qué es un Deep Research Agent y en qué se diferencia de otros agentes?**  
Analiza múltiples fuentes, razona y sintetiza información, no solo busca textos.

**4.49 ¿Qué es un Computer Use Agent y en qué se diferencia de otros agentes?**  
Interactúa directamente con interfaces gráficas (clics, inputs, ventanas), no solo con texto o APIs.

**4.50 ¿Qué ventajas tiene usar una librería como Instructor frente a una llamada directa a la API de un LLM?**  
Garantiza respuestas estructuradas, tipadas y validadas, reduciendo errores y simplificando la integración.

**4.51 Explica qué es computación evolutiva.**  
Es una técnica basada en selección natural y genética que optimiza soluciones mediante mutación, cruce y evolución iterativa de individuos.

1. **Prueba Practica React, .Net y Flutter (Opcional: Kotlin o Swift)**

**2.1 UNICAMENTE REACT y .NET (MVC o MINIMAL APIS)**

Para esta **Prueba Práctica de React, .Net y Flutter** (con la opción de emplear Kotlin o Swift en lugar de Flutter) se te encarga crear un sistema integral para La Cazuela Chapina, negocio que combina dos pilares de la cocina guatemalteca: los tamales tradicionales y las bebidas de maíz y cacao.

Los tamales se venden por unidad, media docena o docena, y cada pieza puede personalizarse según cuatro atributos: tipo de masa (maíz amarillo, maíz blanco o arroz), relleno (recado rojo de cerdo, negro de pollo, chipilín vegetariano o mezcla estilo chuchito), envoltura (hoja de plátano o tusa de maíz) y nivel de picante (sin chile, suave o chapín).

Las bebidas artesanales se ofrecen en vasos de 12 oz y jarros de 1 litro. El cliente elige entre atol de elote, atole shuco, pinol o cacao batido; decide si endulza con panela, miel o prefiere sin azúcar; y, si lo desea, añade malvaviscos, canela o ralladura de cacao como topping.

Para aumentar el valor del ticket hay tres combos predefinidos. El Combo Familiar “Fiesta Patronal” incluye una docena surtida de tamales y dos jarros familiares; el Combo Eventos “Madrugada del 24” suma tres docenas, cuatro jarros y un termo de barro conmemorativo; y un Combo Estacional varía según la época – fiambre en noviembre, quema del Diablo en diciembre y recetas de Cuaresma en Semana Santa –, por lo que la aplicación debe permitir crear o ajustar combos sin redeploy.

Detrás del mostrador se controla inventario de materias primas (masa, hojas, proteínas, granos, endulzantes, especias), empaques y combustible, registrando entradas, salidas, costos y mermas. Un panel de inicio presenta indicadores clave: ventas diarias y mensuales, tamales más vendidos, bebidas preferidas por franja horaria, proporción de picante vs no picante, utilidades por línea y desperdicio de materia prima.

La versión móvil replica la funcionalidad, añade la gestión de sucursales con sus reportes y permite registrar ventas fuera de línea. Cada venta o fin de cocción emite una notificación push. Se valoran módulos opcionales como telemetría de vaporeras, cronometraje de lotes, gestión de proveedores, mapa de mesas y logística de envíos. Para la interfaz web se otorgan puntos extra si grabas un análisis con React Scan y explicas las mejoras sugeridas, sin necesidad de implementarlas.

### Guía ordenada de requisitos

1. Tecnologías objetivo  
   1.1 Frontend web en React  
   1.2 API en .Net  
   1.3 App móvil en Flutter (opcionalmente Kotlin o Swift) con el mismo backend
2. Catálogo de productos  
   2.1 Tamales: unidad, 6 uds, 12 uds  
   2.2 Atributos de tamal: masa, relleno, envoltura, picante  
   2.3 Bebidas: vaso 12 oz, jarro 1 L  
   2.4 Atributos de bebida: tipo, endulzante, topping
3. Combos  
   3.1 Combo Familiar “Fiesta Patronal”  
   3.2 Combo Eventos “Madrugada del 24”  
   3.3 Combo Estacional editable en producción
4. Inventario  
   4.1 Materias primas (masa, hojas, proteínas, granos, endulzantes, especias)  
   4.2 Empaques y combustible  
   4.3 Entradas, salidas, mermas, costos promedio  
   4.4 Bloqueo de venta por punto crítico
5. Dashboard mínimo  
   5.1 Ventas diarias y mensuales  
   5.2 Tamales más vendidos  
   5.3 Bebidas preferidas por horario  
   5.4 Proporción picante vs no picante  
   5.5 Utilidades por línea  
   5.6 Desperdicio de materias primas
6. Integración LLM obligatoria  
   6.1 API OpenRouter (Modelos gratuitos)  
   6.2 Implementación libre, se calificara la creatividad  
   6.3 Puntos extra: voz a texto, MCP de demanda
7. Funciones opcionales de alto impacto  
   7.1 Cronometraje de lotes  
   7.2 Gestión de proveedores  
   7.3 Plano de mesas  
   7.4 Logística de envíos a domicilio
8. Web – puntos extra  
   8.1 Video corto con React Scan   
   8.2 Análisis de resultados  
   8.3 Propuesta de mejoras (conceptual) no es necesario implementar
9. exclusivo  
   9.1 Gestión y reportes de sucursales  
   9.2 Registro de ventas en dispositivo  
   9.3 Notificaciones push por venta y fin de cocción

**IMPORTANTE: se estará calificando diseño, seguridad, rendimiento.**

**FECHA MÁXIMA DE ENTREGA: JUEVES 16 DE OCTUBRE A LA 11 PM**

Enviar un video explicando el proyecto (no hay necesidad de desplegar el proyecto), documentación y adjuntar el o los repositorios de su proyecto a los siguientes correos

[esantos@genesisempresarial.com](mailto:esantos@genesisempresarial.com)

[cfernandez@genesisempresarial.com](mailto:cfernandez@genesisempresarial.com)

[tdeleon@genesisempresarial.com](mailto:tdeleon@genesisempresarial.com)

aagonzalez@genesisempresarial.com

**DUDAS WHATSAPP: 33568980 o 30816121**