PRUEBA TÉCNICA — LÍDER TÉCNICO DE DESARROLLO (ASISYA)

Duración recomendada: **3 días**
Entrega en GitHub + 1 presentación ejecutiva (PDF o Notion).

1. OBJETIVO GENERAL

El candidato debe demostrar criterio técnico, capacidad arquitectónica, liderazgo de desarrollo, buenas prácticas, DevOps, diseño de estándares y visión técnica, usando tecnologías alineadas con ASISYA:

- **Backend:** .NET 7 / .NET 8
- **Frontend:** React
- **Contenedores:** Docker
- **Infraestructura:** AWS (diseño, no despliegue real obligatorio)
- **CI/CD:** GitHub Actions
- **Arquitectura:** microservicios, DDD, clean architecture
- **Calidad:** testing, documentación, code review

2. ALCANCE DE LA PRUEBA

La prueba tiene 4 entregables obligatorios:

1. Diseño arquitectónico completo
2. Implementación parcial de un microservicio crítico (.NET)
3. Documentación técnica y estándares de desarrollo
4. Revisión técnica de un código defectuoso (code review)

3. DESAFÍO TÉCNICO REAL DE ASISYA

Caso: Sistema de Asistencias Vehiculares – Módulo "Asignación Inteligente de Proveedores"

Debes diseñar la solución para este escenario:

Cuando un usuario solicita una asistencia (grúa, cerrajería, batería), el sistema debe:

1. Recibir la solicitud desde la app móvil
2. Validar ubicación del cliente
3. Buscar proveedores disponibles
4. Calcular el proveedor óptimo (tiempo ETA, distancia, disponibilidad, rating)
5. Enviar notificación al proveedor

6. Registrar el caso en la base de datos
7. Permitir seguimiento en tiempo real por el usuario

4. ENTREGABLES

4.1. ENTREGABLE 1 — Arquitectura Completa (obligatorio)

Diseñar una arquitectura **moderna, escalable y segura** con:

4.1.1. Diagrama principal (C4 nivel 1–3)

Incluye:

- API Gateway
- Microservicios
- Bases de datos
- Message broker
- Autenticación (JWT/OAuth2)
- CDN
- S3
- RDS Postgres
- Cache Redis
- Observabilidad
- Load balancers

4.1.2. Diseño de microservicios

Definir al menos:

- AssistanceRequestService
- ProviderOptimizerService (algoritmo de selección)
- NotificationsService
- LocationService

Debes definir:

- responsabilidades
- entidades (DDD)
- bounded contexts
- eventos

- contratos

4.1.3. Estrategia de escalabilidad

Incluye:

- autoscaling
- colas (SQS)
- pub/sub
- procesamiento asincrónico
- rate limits
- fallback y retries

4.1.4. Estrategia de seguridad

- IAM Roles
- JWT
- WAF
- Secrets Manager
- Auditoría
- OWASP

4.2. ENTREGABLE 2 — Microservicio crítico implementado (.NET)

Implementar **solo un microservicio** (no todo el sistema):
ProviderOptimizerService

Debe incluir:

✓ Arquitectura limpia (Clean Architecture o Hexagonal)

✓ Endpoints:

- POST /optimize
- GET /providers/available

✓ Lógica:

Simular selección óptima de proveedor.

✓ Persistencia:

- Postgres o SQL Server
- Migration scripts

✓ **Docker:**

- Dockerfile
- docker-compose

✓ **Tests:**

- Unit tests
- Al menos 1 test de integración

4.3. ENTREGABLE 3 — Estándares técnicos del squad (documento)

Documento obligatorio con:

✓ **Convenciones de código para:**

- .NET
- React
- Docker

✓ **Políticas para:**

- branch strategy
- code reviews
- CI/CD
- definición de done
- manejo de secretos
- arquitectura base del squad
- control de deuda técnica

4.4. ENTREGABLE 4 — Code Review real

Se entrega este snippet defectuoso (lo puedes pegar en el repositorio) y el candidato debe:

- Encontrar errores
- Identificar vulnerabilidades
- Sugerir refactor
- Proponer mejores prácticas

Snippet (C#):

```
[HttpPost("assign")]
public async Task<IActionResult> AssignProvider(Request request)
{
    var providers = _db.Providers.ToList();
    var selected = providers.FirstOrDefault(); // escoger el primero nomas
    if(selected == null) return BadRequest("No providers");

    selected.IsBusy = true;
    _db.SaveChanges();

    return Ok(selected);
}
```

Debe detectar:

- falta de asíncronía real
- ausencia de transacciones
- error de concurrencia
- lógica incorrecta
- no hay validaciones
- no cumple SOLID
- no hay DTOs
- no hay política de selección

Y proponer solución.

5. CI/CD OBLIGATORIO

Pipeline en GitHub Actions con:

- build
- tests
- lint
- docker build
- (opcional) push a ECR (solo mostrar la configuración)

6. ENTREGABLE EXTRA (OPCIONAL)

Frontend React simple con:

- pantalla de login (JWT)
- consulta del estado de proveedores
- tabla con ETA simulado

(Opcional pero suma puntos).

asisya
SOLUCIONES PARA TU DÍA A DÍA