ESCALABILIDAD

**Evaluación de Necesidades**

**Objetivo**: Identificar los puntos críticos que requieren escalabilidad.  
**Acciones**:

* Analizar métricas históricas (ej: picos de pedidos en festivos).
* Determinar cuellos de botella (ej: base de datos, procesamiento de pagos).
* Estimar crecimiento esperado (ej: 30% más de usuarios en 6 meses).

**Herramientas PaaS**:

* **Google Analytics** + **AWS CloudWatch** para identificar patrones de uso.

|  |
| --- |
| Elección del Tipo de Escalabilidad |
| | **Tipo** | **Descripción** | **Uso Recomendado** | | --- | --- | --- | | **Escalabilidad Horizontal** | Añade más instancias de la aplicación (ej: servidores virtuales). | Ideal para manejar tráfico variable (ej: promociones virales). | | **Escalabilidad Vertical** | Aumenta recursos de una instancia (ej: CPU, RAM). | Útil para procesos intensivos (ej: generación de reportes contables complejos). | |

**Proveedores PaaS**:

* **AWS Elastic Beanstalk**: Configura reglas de autoescalado basadas en CPU o solicitudes por minuto.
* **Google App Engine**: Escala automáticamente a cero en horas de baja demanda para ahorrar costos.

**Configuración de Autoescalado en PaaS**

**Pasos**:

1. **Definir métricas de disparo**:
   * Ej: Si el uso de CPU supera el 70% por 5 minutos, añade 2 instancias.
2. **Límites de escalado**:
   * Establecer un máximo de instancias para evitar costos descontrolados (ej: 10 instancias).
3. **Integración con servicios externos**:
   * Asegurar que APIs (ej: pasarelas de pago) soporten escalamiento paralelo.

**Pruebas de Carga y Estrés**

**Objetivo**: Simular escenarios de alta demanda para validar la escalabilidad.  
**Herramientas PaaS**:

* **AWS Load Testing**: Simula 10,000 usuarios concurrentes realizando pedidos.
* **BlazeMeter** (integración con CI/CD): Pruebas automatizadas en cada actualización del código.

**Métricas a validar**:

* Tiempo de respuesta de la API (< 2 segundos).
* Capacidad de la base de datos para manejar transacciones simultáneas.