**EXAMEN DESARROLLADOR FULL STACK WEB**

* 1. ¿Qué beneficios aporta el uso de metodologías ágiles como SCRUM o Kanban en el desarrollo de software?
* **Mayor flexibilidad y adaptación**: Permiten ajustar los requisitos y prioridades a medida que evoluciona el proyecto.
* **Entrega continua de valor**: Se generan entregables funcionales en iteraciones cortas, lo que permite obtener retroalimentación temprana.
* **Mejor comunicación y colaboración**: Fomentan la interacción frecuente entre el equipo de desarrollo, clientes y otras partes interesadas.
* **Mayor transparencia**: Todos los miembros del equipo tienen visibilidad sobre el estado del proyecto a través de herramientas como tableros Kanban o reuniones diarias.
* **Reducción de riesgos**: Los problemas se identifican y corrigen más rápido gracias a revisiones constantes y retrospectivas.
* **Aumento en la productividad y eficiencia**: Al enfocarse en tareas prioritarias y evitar el trabajo innecesario, se optimizan los tiempos de desarrollo.
* **Mejora en la satisfacción del cliente**: La entrega frecuente de funcionalidades permite validar el producto con los usuarios y ajustar según sus necesidades.
  1. Describe las fases típicas de desarrollo en un proyecto basado en metodologías ágiles.

Estas son las **5 fases principales**:

1. **Planificación**
   * Se definen los objetivos del proyecto y las funcionalidades necesarias.
   * Se priorizan tareas en un **backlog** (lista de requerimientos).
   * Se establecen tiempos y asignación de responsabilidades dentro del equipo.
2. **Desarrollo**
   * Se programan las funcionalidades seleccionadas para la iteración o sprint.
   * Se realizan revisiones de código para mantener calidad y buenas prácticas.
   * Se trabaja en colaboración entre desarrolladores, testers y diseñadores.
3. **Pruebas y revisión**
   * Se ejecutan pruebas unitarias, funcionales y de integración.
   * Se valida que las nuevas funciones cumplan los requerimientos.
   * Se recibe retroalimentación del equipo o cliente para realizar ajustes.
4. **Entrega**
   * Se libera una versión funcional del software con las mejoras implementadas.
   * Puede ser una versión parcial para revisión o una entrega final para producción.
5. **Retroalimentación y mejora**
   * Se analiza el desempeño del equipo y del producto en una retrospectiva.
   * Se identifican áreas de mejora en el proceso de desarrollo.
   * Se aplican los aprendizajes en la siguiente iteración.

Este ciclo se repite en cada iteración hasta completar el desarrollo del producto.

**2.1.** Diseña una API RESTful en .NET 6 que tenga un endpoint GET y POST para gestionar un recurso llamado "Producto". Incluye un ejemplo de cómo utilizarías DAPPER para consultar productos desde SQL Server.

**2.2.** Explica las diferencias entre una **VIEW**, un **PROCEDURE** y una **FUNCTION** en SQL Server. Proporciona un ejemplo de uso para cada uno.

* **VIEW (Vista)**
* Es una consulta guardada que muestra datos sin almacenarlos.
* Se usa para simplificar consultas y mejorar seguridad.

**¿Cuándo usarla?**

* Cuando necesitas simplificar consultas complejas reutilizables.
* Para mejorar la seguridad, restringiendo el acceso directo a las tablas.

**Ejemplo: Un reporte de clientes activos que se consulta regularmente**, **sin exponer toda la tabla**

CREATE VIEW VistaClientes AS

**SELECT** Id, Nombre FROM Clientes **WHERE** Activo = 1;

**Uso:**

SELECT \* FROM VistaClientes;

* **PROCEDURE (Procedimiento Almacenado)**
* Ejecuta un conjunto de instrucciones SQL y puede modificar datos.

**¿Cuándo usarlo?**

* Cuando necesitas ejecutar operaciones de lectura y escritura en la base de datos.
* Para automatizar tareas como insertar, actualizar o eliminar datos.

**Ejemplo: Registrar una nueva venta en la base de datos**

CREATE PROCEDURE RegistrarVenta

@ClienteId INT,

@Total DECIMAL

AS

BEGIN

INSERT INTO Ventas (ClienteId, Total, Fecha) VALUES (@ClienteId, @Total, GETDATE());

END;

* **FUNCTION (Función)**
* Devuelve un valor o una tabla, pero no puede modificar datos.
* Se usa dentro de consultas para cálculos o transformaciones.

**¿Cuándo usarla?**

* Cuando necesitas realizar cálculos dentro de consultas SQL.
* Si necesitas devolver un valor específico sin modificar datos.

**Ejemplo**: Calcular el precio con descuento en una consulta

CREATE FUNCTION CalcularDescuento(@precio DECIMAL)

RETURNS DECIMAL

AS

BEGIN

RETURN @precio \* 0.90;

END;

**Uso:**

**SELECT** Nombre, Precio, dbo. CalcularDescuento (Precio) AS PrecioConDescuento

**FROM** Productos**;**

**2.3.** ¿Qué ventajas tiene DAPPER frente a Entity Framework para proyectos con alta demanda de rendimiento?

* Mayor velocidad de ejecución
* Uso eficiente de la memoria
* Control total sobre las consultas SQL
* Mejor rendimiento en lectura de datos
* Ideal para API de alto tráfico

**3.1.** Explica la diferencia entre componentes y servicios en Angular. ¿Cuándo deberías usar un servicio?

Los componentes manejan la presentación y la interacción con el usuario. Sin embargo, no son ideales para manejar lógica compleja o compartir datos entre diferentes partes de la aplicación mientras que los servicios se registran en el Dependency Injection (DI) , lo que permite que sean reutilizados en toda la aplicación.

Los servicios deben usarse cuando:

Se necesita **compartir datos** entre varios componentes.

Se requiere **hacer peticiones HTTP** a un backend.

Se tiene lógica **de negocio** que no debe estar en un componente.

Se quiere **separar responsabilidades** para mejorar la mantenibilidad.

**3.2.** Construye un componente en Angular que consuma la API REST creada en la sección 2, usando Angular Material para mostrar los productos en una tabla.

**3.3.** ¿Qué ventajas ofrece Bootstrap en el desarrollo web responsive? Menciona al menos dos clases útiles.

Diseño Responsivo Automático

Componentes y Clases Predefinidas

Compatibilidad con Navegadores

Personalización y Extensibilidad

Entre las clases útiles tenemos: Row, col , container, container-fluid

**4.1.** Diseña una arquitectura de software en Azure que contemple:

* Azure App Service
* Azure SQL Database
* Azure Blob Storage
* Azure Key Vault
* Azure DevOps  
  (Justifica la elección de cada servicio)

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

* **Azure App Service (Aplicación Web/API)**
* **Función:** Hospeda la aplicación web y la API backend.
* **Justificación:**  
  ✅ Permite despliegue rápido sin administrar infraestructura.  
  ✅ Compatible con .NET, Node.js, Python y más.  
  ✅ Autoescalado y alta disponibilidad integrada.  
  ✅ Integración con Azure AD para autenticación.
* **Azure SQL Database (Base de Datos Relacional)**
* **Función:** Almacena datos estructurados y soporta transacciones SQL.
* **Justificación:**  
  ✅ Administración automática de parches y backups.  
  ✅ Seguridad avanzada (Always Encrypted, TDE).  
  ✅ Soporta escalabilidad horizontal con **Elastic Pools**.  
  ✅ Integración con **Azure Key Vault** para manejar credenciales de acceso.
* **Azure Blob Storage (Almacenamiento de Archivos)**
* **Función:** Almacena archivos como imágenes, documentos o registros de auditoría.
* **Justificación:**  
  ✅ Escalabilidad masiva y bajo costo.  
  ✅ Soporta redundancia geográfica (RA-GRS).  
  ✅ Integración con Azure CDN para mejorar rendimiento.  
  ✅ Soporte de acceso seguro mediante **SAS Tokens**.
* **Azure Key Vault (Gestión de Secretos y Certificados)**
* **Función:** Almacena claves de cifrado, contraseñas y certificados de manera segura.
* **Justificación:**  
  ✅ Protege credenciales de acceso a Azure SQL Database.  
  ✅ Control de acceso mediante **RBAC y Azure AD**.  
  ✅ Soporta **rotación automática de claves y certificados**.
* **Azure DevOps (Gestión de Código y CI/CD)**
* **Función:** Repositorio de código y automatización de despliegues.
* **Justificación:**  
  ✅ Pipelines para despliegue continuo en Azure App Service.  
  ✅ Integración con GitHub y repositorios privados.  
  ✅ Monitoreo y pruebas automáticas en cada cambio.  
  ✅ Control de versiones con Azure Repos.

**Flujo de Desarrollo y Despliegue**

1️ **Desarrolladores** suben el código a **Azure Repos (Git)** en Azure DevOps.  
2️ **Azure Pipelines** compila y despliega automáticamente en **Azure App Service**.  
3️ La aplicación accede a **Azure SQL Database** para obtener datos.  
4️ Los archivos se almacenan en **Azure Blob Storage**.  
5️ Las credenciales de base de datos y claves API se gestionan en **Azure Key Vault**.  
6️ La aplicación se escala automáticamente según la demanda.

**4.2.** ¿Cómo aplicarías principios de seguridad como “least privilege” o “zero trust” en una solución basada en Azure?

* **Principio de Least Privilege**

Uso de Azure Role-Based Access Control (RBAC)

Uso de Managed Identities

Utilizar **Azure Key Vault** para almacenar credenciales y restringir su acceso mediante RBAC.

Restringir el acceso a recursos de red en Azure según necesidad.

Habilitar el acceso temporal a roles de administración para evitar privilegios permanentes.

* **Principio de Zero Trust**

Autenticación Multifactor (MFA), Requerir MFA en todas las cuentas, especialmente en administradores y usuarios con acceso crítico

Identity Protection con Azure AD, Habilitar detección de riesgos en accesos sospechosos y aplicar políticas de respuesta automática.

Azure Conditional Access, Aplicar reglas de acceso basadas en el contexto (ubicación, dispositivo, nivel de riesgo).

Implementar **Azure Firewall** y **Azure Defender for Cloud** para inspección de tráfico y protección proactiva.

Monitoreo con Azure Security Center y Microsoft Sentinel,

**4.3.** Explica cómo implementarías un pipeline de CI/CD usando Azure DevOps para desplegar una aplicación web en Azure App Service.

* **Crear el Repositorio en Azure DevOps**

**Pasos:**

1. Ir a **Azure DevOps** → Crear un **Proyecto**.
2. En **Repos**, crear o subir el código de la aplicación web (puede ser .NET, Node.js, Python, etc.).

* **Configurar el Pipeline de CI (Integración Continua)**

1. Ir a Pipelines → New Pipeline.
2. Seleccionar el repositorio (Azure Repos, GitHub, etc.).
3. Elegir Starter Pipeline o YAML y definir la configuración.
4. Guardar y ejecutar el **primer build** para validar.

* **Configurar el Pipeline de CD (Despliegue Continuo)**

🔹 **Pasos:**

1. Ir a **Pipelines** → **Releases** → **New Release Pipeline**.
2. Seleccionar **Azure App Service Deployment**.
3. Conectar con la **Cuenta de Azure** y elegir el **App Service**.
4. Definir el **artifacts** desde el pipeline de CI.
5. Configurar la **fase de despliegue**, usando tareas como:
   * **Azure Web App Deploy:** Publica la aplicación en App Service.
   * **Slot Swap:** Si usas deployment slots (Staging/Production).
6. Habilitar **CD automático** al finalizar el pipeline de CI.

* **Configurar Variables y Permisos**

🔹 **Variables:** Definir **conexiones a bases de datos, claves API, entornos** en **Library → Variable Groups**.  
🔹 **Permisos:** Asegurar que el agente de Azure DevOps tenga permisos en Azure (Azure DevOps Service Connection).

* **Validar y Monitorear**

✅ Ejecutar el pipeline y verificar los logs en **Azure DevOps → Pipelines**.  
✅ Monitorear el despliegue en **Azure Portal → App Service → Deployment Center**.

**SECCIÓN 5: SEGURIDAD EN AZURE**

**5.1.** ¿Qué mecanismos ofrece Azure para proteger una API expuesta públicamente? Menciona al menos tres.

|  |  |  |
| --- | --- | --- |
| **.** | **¿Qué protege?** | **Características clave** |
| **API Management (APIM)** | Acceso y autenticación | OAuth 2.0, rate limiting, JWT, Azure AD |
| **Application Gateway + WAF** | Seguridad a nivel HTTP/HTTPS | Bloqueo de ataques web, reglas de acceso |
| **DDoS Protection** | Protección contra ataques volumétricos | Mitigación automática de tráfico malicioso |

**5.2.** ¿Cuál es la función de Azure Security Center y cómo ayuda en el cumplimiento de normativas de seguridad?

La función principal es **proteger y mejorar la seguridad** de los recursos en la nube mediante:

✅ **Monitoreo continuo** de la seguridad en entornos de Azure y multinube.  
✅ **Detección y respuesta a amenazas** en tiempo real.  
✅ **Evaluación de vulnerabilidades** y recomendaciones de seguridad.  
✅ **Cumplimiento de normativas** mediante auditorías y reportes.

Ayuda mediante:

**Monitoreo continuo** → Revisa la seguridad de los recursos y verifica si cumplen con normativas como **ISO 27001, GDPR o PCI-DSS**.

**Alertas y recomendaciones** → Si detecta configuraciones inseguras, sugiere cambios para corregirlas automáticamente.

**Generación de reportes** → Crea informes que sirven para auditorías y demuestran que se cumplen las regulaciones.

**Protección contra amenazas** → Detecta y bloquea ataques que podrían poner en riesgo el cumplimiento de las normativas.

**SECCIÓN 6: CASO PRÁCTICO – ANÁLISIS Y PROPUESTA DE SOLUCIÓN**

**6.1.** Un cliente indica que su aplicación web en Azure está funcionando muy lentamente y presenta picos de carga durante ciertas horas. ¿Cómo analizarías el problema y qué tipo de solución propondrías?

**Plan de Acción**

**Paso 1**: Monitorear métricas con Azure Monitor y Application Insights.  
 **Paso 2**: Identificar el principal cuello de botella (CPU, base de datos, escalabilidad).  
 **Paso 3**: Aplicar la solución adecuada (optimización, escalado, balanceo de carga).  
 **Paso 4**: Evaluar mejoras y realizar ajustes si es necesario.