¿Cuál de las siguientes afirmaciones es verdadera acerca de la aplicación de CQRS (Command Query Responsibility Segregation) en la organización de la lógica de negocio?

a

CQRS propone un único modelo que maneja tanto comandos como consultas, simplificando la interacción con la base de datos.

b

**CQRS separa comandos, que cambian el estado, de consultas, que retornan resultados, mejorando la organización de la lógica de negocio.**

c

CQRS se enfoca exclusivamente en separar la lógica de negocio de la capa externa.

d

En CQRS, tanto los comandos como las consultas tienen la responsabilidad de cambiar el estado del sistema.

¿Cuál de las siguientes afirmaciones describe correctamente los dobles de prueba?

a

Los dobles de prueba no son útiles para controlar y verificar las interacciones entre diferentes componentes de la aplicación.

b

**Los dobles de prueba son objetos reales utilizados para probar el comportamiento de la aplicación en tiempo real.**

c

Los dobles de prueba siempre deben ser utilizados en todas las pruebas unitarias sin excepciones.

d

Los dobles de prueba son objetos que se instalan en lugar del objeto real con la intención de ejecutar la prueba.

#### base de datos

¿Cuál de las siguientes afirmaciones describe correctamente el propósito de los casos de uso en la organización del dominio en una arquitectura limpia?

a

Los casos de uso son responsables de interactuar directamente con la capa de presentación y la capa de persistencia.

b

Los casos de uso reemplazan completamente a las entidades en el modelo de dominio.

c

**Los casos de uso coordinan el flujo de datos desde y hacia el modelo de dominio, implementando reglas de negocio específicas de la aplicación.**

En el contexto de una arquitectura limpia, ¿qué práctica se recomienda evitar al implementar una aplicación web?

a

Utilizar inyección de dependencias para conectar diferentes capas de la aplicación.

b

Aislar la lógica de negocio en la capa de aplicación, manteniéndola alejada de la capa externa.

c

**Colocar lógica de negocio en la capa de presentación o en los controladores.**

d

Colocar lógica de negocio en la capa de aplicación.

Un sistema tiene un disparador (scheduler) que se ejecuta automáticamente todos los días a las 5 PM para enviar correos electrónicos. En una arquitectura hexagonal, ese disparador se consideraría:

a

Un actor primario. Inicia la interacción y lleva a la ejecución de la aplicación.

b

Un componente de la capa de aplicación, ya que está relacionado con la funcionalidad principal del sistema.

c

**Parte del modelo de dominio, ya que es responsable de una tarea específica dentro del sistema.**

d

**Un actor secundario. La aplicación será siempre el actor primario del sistema.**

¿Cuál es el número ideal de capas en el dominio?

a

**Depende del problema. No existe un número correcto.**

b

Uno. Tal como se ve en la arquitectura hexagonal.

c

Dos. Tal como se ve en clean architecture.

d

Tres. Tal como se ve en la arquitectura cebolla.

¿Cuál de las siguientes formas de implementar el dominio permite organizarlo usando procedimientos?

a

**Script de transacción**

b

CQRS

c

Modelo de dominio

Una de las siguientes opciones es FALSA respecto al dominio en una arquitectura limpia

a

Al dominio también se le conoce como lógica de negocio o lógica de dominio.

b

El dominio puede ser organizado a través de casos de uso.

c

El dominio es la razón de ser del sistema.

d

**El dominio puede retornar datos formateados para una interfaz gráfica específica.**

Una capa de servicios interactuando con un modelo de dominio se asemeja a uno de los siguientes patrones de diseño.

a

**Façade (fachada en español). La capa de servicios esconde detalles de la implementación del modelo de dominio.**

b

Adapter. La capa de servicios adapta las solicitudes de la capa externa al modelo de dominio.

c

Strategy. La capa de servicios implementa distintos algoritmos (estrategias) para interactuar con el modelo de dominio.

¿Cuál es la gran ventaja del modelo de dominio sobre el script de transacción?

a

**Aprovecha múltiples beneficios de la programación orientada a objetos, como los datos, el comportamiento, las relaciones, entre otros.**

b

Funciona mejor con bases de datos relacionales ya que se puede tener una entidad por cada tabla.

c

Es mucho más simple de implementar ya que se pueden usar patrones de diseño.

Supón que trabajas en un sistema de un hospital que gestiona historias médicas de los pacientes. Tienes un requerimiento para enviar un reporte via FTP a un servicio gubernamental. De acuerdo a la nomenclatura del curso (modelo de dominio, aplicación, capa externa), ¿dónde van la interfaz y la implementación del adaptador?

a

**La interfaz se ubicaría en la aplicación y la implementación en la capa externa.**

b

La interfaz se ubicaría en el modelo de dominio y la implementación en la aplicación.

c

Tanto la interfaz como la implementación van en el modelo de dominio.

d

Tanto la interfaz como la implementación van en la aplicación.

e

Tanto la interfaz como la implementación van en la capa externa.

¿Qué significa el Principio de Abierto Cerrado en SOLID?

a

Un módulo debería estar abierto a extensiones y cerrado a modificaciones.

b

Un módulo debería estar abierto a una sola razón para ser cambiado.

c

Un módulo debería estar abierto a modificaciones y cerrado a extensiones.

d

Ninguna de las opciones es correcta.

En el caso de Netflix usando arquitecturas limpias, se menciona que lograron cambiar rápidamente un repositorio que interactuaba usando JSON a uno que utilizaba GraphQL. ¿Qué principio SOLID se ve directamente aplicado en esta situación?

a

**Principio de Abierto Cerrado. El sistema se extendió para acceder al API en GraphQL, sin tener que modificar la implementación existente.**

b

Principio de Segregación de Interfaces. Las distintas interfaces (APIs) que el sistema podía acceder fueron separadas, y esto facilitó el reemplazo.

c

Principio de Sustitución de Liskov. Se reemplazó la implementación en JSON por la implementación en GraphQL.

Una de las siguientes es una diferencia entre una arquitectura limpia y la arquitectura de 3 capas

a

El uso de la inyección de dependencias.

b

**La dependencia entre el dominio y el acceso a datos.**

c

Todas las opciones son correctas..

d

La dependencia entre la presentación y el dominio.

Laravel es un framework de desarrollo web para PHP. ¿A qué capa pertenece y por qué?

a

Dominio. La lógica de negocio se puede implementar directamente en los objetos que provee Laravel.

b

Capa externa. La lógica de negocio debe ser independiente del framework que se utilice.

c

Capa externa. El usuario accederá la aplicación a través de la web.

d

Dominio. Sin Laravel, implementar la lógica de negocio puede ser más complejo.

¿En qué arquitectura limpia de referencia el dominio se modela a través de casos de uso y entidades?

a

Arquitectura de 3 capas

b

**Clean Architecture**

c

Arquitectura Cebolla

d

Arquitectura Hexagonal

En una arquitectura cebolla, ¿qué capa se encuentra en el centro?

a

Servicios de dominio

b

Servicios de aplicación

c

**Modelo de dominio**

d

Infraestructura

¿Cuál de las siguientes afirmaciones relacionadas con pruebas es correcta?

a

Las arquitecturas limpias no permiten la diferenciación entre pruebas unitarias y de integración.

b

**La testabilidad es una característica clave de las arquitecturas limpias.**

c

Las arquitecturas limpias dificultan la realización de pruebas en la aplicación.

d

Las pruebas en el modelo de dominio son siempre pruebas de integración.

Supón que trabajas en un sistema de un hospital que gestiona historias médicas de los pacientes. Tienes un requerimiento para consultar la historia de un paciente con un doctor en particular. De acuerdo a la nomenclatura del curso (modelo de dominio, aplicación, capa externa), ¿dónde van la interfaz y la implementación del repositorio?

a

La interfaz se ubicaría en la capa de aplicación y la implementación en la capa externa.

b

La interfaz se ubicaría en el modelo de dominio y la implementación en la aplicación.

c

Tanto la interfaz como la implementación van en la capa externa.

d

Tanto la interfaz como la implementación van en la aplicación.

e

Tanto la interfaz como la implementación van en el modelo de dominio.

¿Cuál de los siguientes desafíos es común al implementar una arquitectura limpia en un proyecto?

a

**Mantener un equilibrio adecuado de responsabilidades entre la capa de aplicación y el modelo de dominio.**

b

Asegurar que todas las capas de la aplicación tengan acceso directo a la base de datos.

c

Asegurar que las capas internas dependan directamente de las capas externas.

Supón que trabajas en un sistema de un hospital que gestiona historias médicas de los pacientes. Tienes un requerimiento para consultar la historia de un paciente con un doctor en particular. De acuerdo a la nomenclatura del curso (modelo de dominio, aplicación, capa externa), ¿dónde van la interfaz y la implementación del repositorio?

a La interfaz se ubicaría en la capa de aplicación y la implementación en la capa externa.

b La interfaz se ubicaría en el modelo de dominio y la implementación en la aplicación.

c Tanto la interfaz como la implementación van en la capa externa.

d Tanto la interfaz como la implementación van en la aplicación.

**e Tanto la interfaz como la implementación van en el modelo de dominio.**

En el contexto de una arquitectura limpia, ¿qué beneficio se obtiene al mantener la base de datos en la capa externa?

a

Asegura que la base de datos sea la única fuente de datos en la aplicación.

b

Elimina la necesidad de realizar pruebas en la capa de acceso a datos.

c

**Permite acceder a múltiples fuentes de datos, como bases de datos de legado o sistemas de terceros.**

d

Facilita la migración entre diferentes bases de datos, ya que es un proceso muy común.

¿Cuál de los siguientes **NO se considera un elemento externo** en una arquitectura limpia?

a

Interfaz gráfica

b

**Dominio**

c

Un framework de desarrollo web

d

Un sistema de recursos humanos

e

Bases de datos

Un desarrollador decidió poner parte de la lógica de negocio en una aplicación de escritorio. ¿Cuál es la principal limitación de esta decisión?

a

Escalabilidad. Incluir lógica de negocio en una aplicación de escritorio afecta su escalabilidad.

b

**Reutilización. Otras partes del sistema no podrán usar fácilmente esa lógica.**

c

Rendimiento. Poner lógica de negocio en una aplicación de escritorio puede hacer el sistema más lento.

En el curso se menciona que una arquitectura limpia no es muy útil en sistemas de vida corta, como una prueba de concepto o un producto mínimo viable. ¿A qué se debe esto?

a

La complejidad de una arquitectura limpia puede ser demasiado en caso de que el sistema a implementar sea muy sencillo.

b

El esfuerzo adicional de implementar una nueva funcionalidad puede ir en contra de la agilidad que se requiere en sistemas de vida corta.

c

Todas las opciones son correctas.

d

Uno de los fuertes de estas arquitecturas es la mantenibilidad, y ese tipo de sistemas no la requieren.

**Un desarrollador decidió poner parte de la lógica de negocio en una aplicación de escritorio. ¿Cuál es la principal limitación de esta decisión?**

a Escalabilidad. Incluir lógica de negocio en una aplicación de escritorio afecta su escalabilidad.

b Reutilización. Otras partes del sistema no podrán usar fácilmente esa lógica.

c Rendimiento. Poner lógica de negocio en una aplicación de escritorio puede hacer el sistema más lento.

¿En qué lenguajes de programación se puede implementar una arquitectura limpia?

a

Lenguajes de programación compilados (como Java y C#)

b

En cualquiera, mientras tenga orientación a objetos.

c

Lenguajes de programación dinámicos (como JavaScript)

La regla de la dependencia implica que la capa externa depende del dominio y no al revés. Esto es fundamental en una arquitectura limpia. ¿Por qué?

a

**Todas las opciones son correctas**

b

Porque protege el dominio de cambios en la infraestructura

c

Porque permite cambiar una implementación por otra en la capa externa

d

Porque facilita las pruebas al dominio