JUSTIFICACIÓN USO PATRONES DE DISEÑO
PROGRAMACIÓN AVANZADA
CORTE 2

UNIVERSIDAD CENTRAL

PATRON DE DISEÑO	***VIABILIDAD USO***	***USO***	***JUSTIFICACIÓN***
Interfaces	SI	SI	Para el usuario debe ser trasparente el proceso que se siga para la creación de las preguntas de diferente tipo, si solicita una pregunta de selección múltiple o una de Falso o Verdadero debe retornar una pregunta.
Abstract Parent Class	SI	SI	Aunque las preguntas son de diferentes tipos comparten ciertos atributos y métodos, por ende y con el fin de evitar la redundancia que crearon clases abstractas.
Private Methods	SI	SI	Se requiere que los exámenes puedan ser modificados solamente por la misma clase cuestionarios.
Accessor Methods	SI	SI	Cuando se requiere generar un cuestionario, se requiere el uso de clases como: Materia, Tipo Producto, Complejidad etc, sin embargo estas clases no tienen acceso a la modificación del cuestionario, por lo anterior se requiere diseñar los métodos de acceso para obtener y actualizar los valores de las clases.
Constant Data Manager	SI	NO	Aunque este patrón ayuda en la organización de la aplicación, no tenemos un conjunto de constantes las cuales sean agrupadas.
Inmutable Object	SI	SI	Este patrón se implementa para asegurar que la instancia creada del servidor, mantenga siempre actualizados sus atributos para todas las clases que intervengan con el.

JUSTIFICACIÓN USO PATRONES DE DISEÑO
PROGRAMACIÓN AVANZADA
CORTE 2

UNIVERSIDAD CENTRAL

PATRON DE DISEÑO	***VIABILIDAD USO***	***USO***	***JUSTIFICACIÓN***
Monitor	NO	NO	Este patrón no se implemento ya que aunque existen varios clientes, el uso de la aplicación se realizará de forma independiente y el usuario no podrá modificar información ya registrada en la base de datos, evitando afectar a otros clientes. (Sólo podrán visualizar los exámenes)
Singleton	NO	NO	Dentro del diseño de la aplicación no se requiere este patrón, ya que no existe ninguna clase que se requiera instanciar una sóla vez y que deba ser de acceso global.
Prototype	SI	SI	Dado que el objetivo de la aplicación es la generación de un número indeterminado de exámenes (conjunto de preguntas con diferentes características particulares, pero guardando una misma estructura general) podría resultar bastante costosa la creación - construcción de cada uno de los exámenes. Prototype se implementó con el fin de a partir de una instancia de cuestionario creado, clonarlo y cambiar sus valor reduciendo el uso de memoria y agilizando la generación de los exámenes.
Builder	SI	NO	Este patrón podría solucionar el tema de la creación de los cuestionarios, sin embargo no se incluyó en el diseño, ya que se había implementado Abstract Factory para resolver este problema.
Factory Method	SI	NO	Aunque se requiere una fabrica para la creación de varios objetos, los cuestionarios son objetos complejos ya que su construcción depende de varias clases, por lo anterior se requiere guardar una relación entre esas clases y no se implementa este patrón.
Abstract Factory	SI	SI	La preguntas que se incluyan en los cuestionarios pueden tener diferente configuración dependiendo del tipo de pregunta elegido, por ejemplo, una pregunta del falso o verdadero, tiene sólo una respuesta, sin embargo una pregunta de selección múltiple puede tener más de una respuesta, así que se requiere un control dentro de la aplicación en el que de acuerdo al tipo de pregunta tenga un determinado número de respuestas posibles además de la complejidad y el porcentaje. Se implementa este patrón para guardar una relación entre el tipo de pregunta elegido y las características de la pregunta.

JUSTIFICACIÓN USO PATRONES DE DISEÑO
PROGRAMACIÓN AVANZADA
CORTE 2

UNIVERSIDAD CENTRAL

PATRON DE DISEÑO	***VIABILIDAD USO***	***USO***	***JUSTIFICACIÓN***
Flyweight	SI	SI	Dado que los exámenes que se deben generar cumplen con un conjunto de características comunes, este patrón mejorará el rendimiento en uso de memoria en el proceso de creación de los exámenes.
Adapter	SI	NO	A la fecha no se ha diseñado alguna clase que sea incompatible con otra que la requiera, por ende aun no se usará de este patrón.
Composite	SI	NO	Aunque se puede ver un examen como un objeto compuesto de otros menores: Un examen esta compuesto de Preguntas, las cuales están compuestas por un tema, un tipo de respuestas, un conjunto de respuestas, entre otros. No se ve necesario implementarlos ya que la estructura aun no es muy compleja y ya están definidas las clases que componen las preguntas.
Decorator	SI	NO	A la fecha no se requiere agregar responsabilidades de forma dinámica a las clases ya definidas.
Facade	SI	SI	Este patrón será usado para separar los subsistemas del servidor y el cliente.