INGENIERÍA DE SOFTWARE
1ASI0730 - APLICACIONES WEB
PRÁCTICA CALIFICADA 2
202520

NRC: 7454
Profesor: Velásquez Núñez, Ángel Augusto
Duración: 100 minutos
Indicaciones:

1. El examen consta de 1 pregunta, y tendrá 100 minutos para resolverlas.
2. La pregunta es de tipo Proyecto de Software y la entrega de su respuesta es a través de envío de archivo empaquetado **.zip** con nombre *pc2<NRC>u<código-estudiante>.zip*, conteniendo el proyecto de software, en la Actividad PC2 (Práctica calificada 2).

Enunciado:

Caso Delta Faucet Company

Delta Faucet Company (https://www.deltafaucetcompany.com/), también conocida como DFC, fabrica grifería residencial y comercial, además de otros productos para cocinas y baños que transforman la experiencia con el agua. Fundada por Masco Corporation en 1954 con la introducción del grifo monomando, Delta Faucet Company se enorgullece de ser el líder en innovación de grifos de Estados Unidos, con productos de las marcas Delta®, Brizo®, Kraus® y Peerless®.

DFC extiende la experiencia a través de DFC@Home™, una smartphone app (https://www.deltafaucet.com/app), con versiones en:

Google Play Store: https://play.google.com/store/apps/details?id=com.deltafaucet.DFCatHome&hl=en_US y Apple AppStore: https://apps.apple.com/us/app/dfc-home/id6474621406

DFC@Home™ es la solución integral para aprovechar al máximo los productos Delta® conectados. La aplicación brinda información de los dispositivos y permite la configuración remota de los mismos, como thresholds para temperature y water flow. La aplicación se empareja con los dispositivos disponibles en la red WiFi.

Para los Partners ofrece el portal https://www.dfcpro.com, donde brinda información técnica sobre los diferentes productos.

Pregunta 1 (20 p.).

Usted se integra al backend software developer team, a cargo de la creación de un **RESTful API** que brinde soporte a las operaciones de Delta Faucet Company.

El ecosistema de Delta Faucet Company requiere que la aplicación *de backend* cuente con **Endpoints** en el RESTful API, para el manejo de la información de Dispatch Orders (**DispatchOrder**) conformadas por los atributos dispatchOrderId (DispatchOrderId), productId (ProductId), requestedUnits (int), dispatchExpectedAt (datetime), qualityLevel (EQualityLevel), dispatchCompletedAt(datetime), notes(string).

Como reglas de negocio, Delta Faucet Company:

- No permite que se registre un **dispatch order** con un valor de **requested units menor o igual a 100 ni mayor a 5000**.
- El valor de **expected at** debe ser al menos 30 días menor o igual a la fecha actual.
- El valor de **completed at** debe ser mayor o igual que la fecha de **expected at**.
- El valor de **quality level** debe hacer referencia a una de las posibilidades de un enumeration *EQualityLevel*, definido como sigue:

Id	Name	Criterio de asignación
1	Outstanding	dispatchCompletedAt >= dispatchExpectedAt por 2 días o menos
2	Normal	dispatchCompletedAt > dispatchExpectedAt por mas de 2 días
3	BellowExpected	dispatchCompletedAt > dispatchExpectedAt por mas de 5 días
4	Inefficient	dispatchCompletedAt > dispatchExpectedAt por mas de 7 días

- El tipo **DispatchOrderId** es un value object type que representa el identificador de la orden de producción para el negocio. Contiene un valor de tipo Guid no nulo y no vacío. Es generado automáticamente al momento del registro.
- El tipo **ProductId** es un value object type que representa al identificador de un producto en el negocio, que se genera en otro bounded context. Contiene un valor de tipo Guid no nulo y no vacío, que debe ser proporcionado para proceder al registro.
- Establece que la información que se desea preservar de los ***Dispatch Orders*** incluye **id** (**int**, **obligatorio**, **autogenerado**, **llave primaria**), **dispatchOrderId** (**DispatchOrderId**, **obligatorio**), **productId** (**ProductId, obligatorio**), **requestedUnits** (int, obligatorio), **dispatchExpectedAt** (**datetime, obligatorio**)**, dispatchCompletedAt (datetime, obligatorio), qualityLevel (EQualityLevel, obligatorio), notes (string, opcional)**.
- El valor de **qualityLevel** no se solicita en el request, sino que se asigna internamente uno de los valores de **EQualityLevel** según los criterios de asignación de la tabla indicada líneas arriba.
- En el caso de responses que retorna el API con información de **dispatch orders**, para cada **DispatchOrder** se debe incluir **id**, **dispatchOrderId** (como string), **productId** (como string), **requestedUnits**, **qualityLevel (como string)**, **dispatchCompletedAt**, **dispatchDays** (número de días entre expected at y completed at), notes. No se incluye expected at.
- Considere que DispatchOrder es un aggregate root y por lo tanto es auditable, con el fin de tener un registro de las fechas de creación y última actualización.

Durante la etapa de desarrollo, le asignan trabajar en específico sobre el Endpoint:

/api/v1/dispatch-orders

Dispatch Orders Endpoint (http://localhost:{port}/api/v1/dispatch-orders)
Debe implementar **solo una** operación en el RESTful API: agregar (POST) un **dispatch order**. Los valores de **id** son autogenerados al momento de almacenar la información. Ante una adición satisfactoria, retorne el HTTP Status 201 y en el body del response el resource incluyendo la información del dispatch order y su id generado. En caso de error incluya el HTTP Status correcto según el error, así como el mensaje de error en el body del response.

Incluya como parte del desarrollo la implementación de las reglas de negocio.

Technical constraints

1. Elabore la solución con C#, NET 9 y ASP.NET Core Framework.
2. Cree su solución y el proyecto de software con el nombre ***pc2<NRC>u<código-estudiante>.API*** (por ejemplo, *pc27454u201621873.API*).
3. Considere que los conceptos **DispatchOrder, ProductId** e **EQualityLevel** pertenece al bounded context **SCM**.
4. Considere el bounded context **Shared** con los elementos que correspondan, incluyendo clases base, interfaces base, clases de propósito general, extensiones generales o de políticas (según ejemplos en clase).
5. La información debe ser persistente en una base de datos relacional (MySQL), en un esquema **dfc**.
6. Aplique buenas prácticas de Arquitectura de Software, enfoque de **Domain-Driven Design**, separación en bounded contexts, **layered architecture** (domain, application, interfaces, infrastructure), patrones de strategic y tactical Domain-Driven Design, patrón CQRS, Resource, Assembler, principios y patrones de diseño de software orientado a objetos, convenciones de nomenclatura en **inglés**, así como buenas prácticas de nomenclatura en C# (entre ellas Upper Camel Case para Clases, Properties, Métodos; Lower Camel Case para atributos privados) y buenas prácticas para nomenclatura de objetos de Base de Datos (entre ellas snake case, tablas en plural, sin mnemónicos, id como nombre de primary key).
7. Utilice minúsculas para los nombres de URL para todos los endpoints.
8. Utilice Properties para representar los atributos de las clases Entity.
9. Documente su código con **XML documentation comments**, colocando información de propósito para principales objetos de programación, así como propósito, parámetros y valor retornado en clases y métodos relevantes. Incluya como parte de la documentación sus nombres y apellidos como valor en un tag <remarks>.
10. Incluya documentación de Endpoints con OpenAPI con textos en inglés.
11. Todos los textos de mensajes deben ser en inglés.
12. Incluya en el archivo README.md, la información en inglés de la aplicación, descripción y su información como author.
13. Considere la gestión de excepciones en la aplicación.
14. Empaquete su solución como un archivo **.zip**. (**único formato válido**) con el nombre ***pc2<NRC>u<código-estudiante>.zip*** (por ejemplo, *pc27454u201621873.zip*).
15. Suba su archivo de solución en la Actividad indicada para la Práctica Calificada 2.

NO forma parte del alcance del proyecto:

1. Soporte de CORS.
2. Security.
3. Testing.

Rúbrica de calificación

Criterio de Calificación	Sobresaliente (S)	Esperado (E)	Necesita Mejorar (M)	Insuficiente (I)	Calificación
C01. Building y ejecución	Al abrir el proyecto y ordenar la ejecución, ésta se inicia sin problemas. El API es accesible en la ruta indicada.	La aplicación no llega a iniciar y ejecutarse, sin embargo el proceso de building llega a concluir.	Al cargar el proyecto el proceso de building presenta errores y no llega a concluir.	No elabora solución	
	2.0 puntos	**1.0 punto**	**0.5 puntos**	**0 puntos**	
C02. Endpoint	El RESTful API expone el endpoint solicitado en la ruta especificada. Se evidencia la funcionalidad de la operación CRUD especificada, según las convenciones REST, proporcionando en cada caso los valores esperados y respondiendo adecuadamente ante las excepciones. Se evidencia la persistencia de los objetos solicitados, cumpliendo con estructura según enunciado, en una tabla con nombre según convenciones en base de datos relacional indicada en un esquema según el nombre indicado. El Endpoint cuenta con documentación basada en OpenAPI.	El RESTful API expone el endpointsolicitado en la ruta especificada. Se evidencia parcialmente la funcionalidad de la operación CRUD especificada, según las convenciones REST, proporcionando en algunos casos los valores esperados y respondiendo de forma parcialmente adecuada ante las excepciones, o se evidencia parcialmente la persistencia de los objetos solicitados con estructura según enunciado, en una tabla con nombre según convenciones en base de datos relacional indicada en un esquema según el nombre indicado.	La aplicación implementa y expone el endpoint solicitado, pero no cumple con la ruta especificada o no se puede registrar elementos.	La aplicación no implementa o expone el endpoint solictado.	
	6.0 puntos	**4.0 puntos**	**2.0 puntos**	**0 puntos**	
C03. Business Rules	El desarrollo incluye la implementación de reglas de negocio, cubriendo de forma completa las condiciones y escenarios establecidos, siendo éstas ejecutables, con adecuado manejo de excepciones, implementando éstas en las capas más adecuadas, aplicando convenciones y buenas prácticas.	El desarrollo incluye la implementación de la mayoría de reglas de negocio, cubriendo de forma parcial las condiciones y escenarios establecidos, siendo éstas ejecutables, implementándolas en las capas adecuadas en la mayoría de casos, ó aplicando parcialmente convenciones y buenas prácticas.	El desarrollo incluye la implementación de algunas de las reglas de negocio, incumpliendo la mayoría de condiciones y escenarios establecidos, siendo éstas ejecutables, implementándolas en las capas adecuadas en muy pocos casos, ó con poca evidencia de aplicar convenciones y buenas prácticas.	No implementa reglas de negocio, o no cubre escenarios más allá de operaciones CRUD básicas, o éstas no son ejecutables.	
	4.0 puntos	**3.0 puntos**	**1.5 puntos**	**0 puntos**	
C04. Code Organization	El desarrollador organiza el código y los elementos de backend de la solución, aplicando buenas prácticas de C#, .NET Framework Core, ASP.NET Core y Domain-Driven Design, agrupando los elementos de la solución según convenciones, manteniendo organización de paquetes y carpetas recomendadas por el fabricante y buenas prácticas de la industria de software.	El desarrollador aplica en la mayoría de casos para el backend convenciones, recomendaciones y buenas prácticas de C#, .NET Framework Core, ASP.NET Core y Domain-Driven Design.	El desarrollador en algunos casos para el backend convenciones, recomendaciones y buenas prácticas de C#, .NET Framework Core, ASP.NET Core y Domain-Driven Design.	No se evidencia un criterio de organización para los elementos de la solución.	
	3.0 punto	**2.0 puntos**	**1.0 puntos**		
C05. Code Quality	Utiliza para el backend el lenguaje de programación C# La codificación tiene un estilo claro, indentando los bloques de código según los estándares de programación correspondientes al lenguaje, aplicando una lógica consistente en los métodos, condicionales sin escenarios no contemplados, uso adecuado de reutilización de código para evitar redundancia. Aplica patrones de arquitectura, principios y patrones de diseño. Distribuye el código en los niveles correspondientes, asignando lógica de persistencia, lógica de negocio, lógica de control, lógica de mapping y transferencia a las interfaces y clases que corresponden. Cumple de forma completa con los technical constraints.	Utiliza para el backend el lenguaje de programación C#. La codificación es funcional, aplica en la mayoría de casos los estándares de indentación de bloques de código, ó existen algunas ineficiencias en la codificación: redundancia ó inconsistencias en la lógica de programación. Aplica parcialmente patrones de arquitectura y patrones de diseño, o existe en algunas partes una distribución de la lógica en los niveles incorrectos. Cumple con la mayoría de los technical constraints.	Utiliza para el backend el lenguaje de programación C#. La codificación es funcional, pero sólo aplica algunos de los estándares de indentación de bloques de código, ó existen muchas ineficiencias en la codificación: redundancia ó inconsistencias en la lógica de programación. Aplica algunos patrones de arquitectura y patrones de diseño, o existe en muchos casos una distribución de la lógica en los niveles incorrectos. Cumple de con solo algunos de los technical constraints.	No utiliza el lenguaje de programación C# para el backend, ó la codificación es funcional pero no se evidencia aplicación de estándares ó criterios de eficiencia en la codificación, con ausencia de comentarios, ó no aplica patrones de arquitectura ni patrones de diseño, o la codificación no es funcional.	
	3.0 puntos	**2.0 punto**	**1.0 puntos**	**0 puntos**	
C06. Naming Standards	El desarrollador aplica en todos los nombres de objetos de programación como namespaces, componentes, interfaces, clases, objetos, variables, constantes y métodos la nomenclatura en inglés y la nomenclatura estándar para identificadores de clases, objetos, miembros de programación, así como los recursos.	El desarrollador aplica en la mayoría de casos la nomenclatura en inglés y la nomenclatura estándar para identificadores de clases, objetos, miembros de programación, así como los recursos.	El desarrollador aplica en muy pocos casos la nomenclatura en inglés y la nomenclatura estándar para identificadores de clases, objetos, miembros de programación, así como los recursos.	El desarrollador no aplica nomenclatura en inglés para los objetos de programación ó recursos.	
	2.0 puntos	**1.0 punto**	**0.5 puntos**	**0 puntos**	
Total	**20 puntos**	**13.0 puntos**	**6.5 puntos**	**0 puntos**	

Lima, 21 de noviembre del 2025

Anexos

Anexo A. Referencias

Comprimir y descomprimir archivos:
https://support.microsoft.com/es-es/windows/comprimir-y-descomprimir-archivos-8d28fa72-f2f9-712f-67df-f80cf89fd4e5

REST API Tutorial:
https://restfulapi.net/

Swashbuckle.AspNetCore - OpenAPI Library for ASP.NET Core:
https://github.com/domaindrivendev/Swashbuckle.AspNetCore

AutoMapper Documentation:
https://docs.automapper.org/en/latest/index.html

MySQL Connector/NET for Entity Framework – Entity Framework Core Support:
https://dev.mysql.com/doc/connector-net/en/connector-net-entityframework-core.html

How to change port number for a .NET Core project using IIS Express
https://blog.novacare.no/how-to-change-portnumber-in-dotnetcore/