Trabajo Práctico N°1 - (parte 1)

Fecha límite de entrega: Lunes 28/08/2023 a las 17:00 hs

Criterios de evaluación:

Funcionalidad y Requisitos:

- Completar todas las tareas requeridas.
- Cumplir con los requisitos específicos del enunciado.
- Manejar adecuadamente casos extremos y situaciones inusuales.

Diseño y Estructura del Código:

- Organizar el código en funciones/módulos coherentes.
- Mantener un código legible y bien comentado.
- Utilizar nombres descriptivos para variables y funciones.
- Seleccionar y emplear las estructuras de datos adecuadas.
- Dividir el código en módulos reutilizables.

Pruebas y Validación:

- Incluir casos de prueba relevantes y abarcar diversos escenarios.
- Obtener resultados coherentes con los esperados.
- Manejar errores de manera adecuada y sin fallos inesperados.

Creatividad y Mejoras:

- Agregar características adicionales que demuestren comprensión profunda.
- Contribuir con soluciones innovadoras o enfoques únicos.

Documentación:

- Proporcionar instrucciones claras para compilar, ejecutar y usar el programa.
- Acompañar el código con comentarios explicativos que aclaren su funcionamiento.

Formato de entrega:

- Código: se realizará a través de un repositorio de código en GitHub, cuyo nombre deberá seguir el siguiente formato: “nombre_del_alumno-trabajo_practico_1-backend”.
- Documento: se realizará en formato PDF (sin excepción) y con la fuente 'Times New Roman', tamaño N° 12 para los párrafos, N° 16 para los títulos y se permiten negritas e itálicas.

Extras:

- El trabajo es de modalidad individual (solamente una persona puede participar en la creación del trabajo y debe ser la persona responsable de entregar dicho trabajo en el aula virtual).
- Aquellos trabajos donde se demuestre una clara complicidad entre 2 o más involucrados terminará en la descalificación del trabajo práctico (sin excepción).

Consigas:

- **Crear un servidor http** que el cual permita hacer las 5 operaciones básicas sobre una entidad en particular de las que van a estar disponibles más adelante.

- **Además, el servidor** deberá tener la capacidad de registrar usuarios. La entidad seleccionada debe tener una relación de uno a muchos con la entidad de usuarios, lo que significa que un usuario podrá tener varios registros de la entidad seleccionada. Por lo tanto, si no hay un usuario previamente registrado, no se podrán crear registros de la entidad seleccionada.

- **Los campos de entrada** del servidor deben estar validados previamente antes de llegar a los controladores.

- **Aplicar** la división de carpetas y archivos correspondiente a cada módulo con la modularización correspondiente aplicando el patrón de diseño MVC.

- **Presentar un documento técnico** en el que se detalle la elección de la temática elegida, el porqué de dicha elección y donde también se grafique con un formato de red conceptual la lógica que se emplea en dicha temática.

- **El documento** deberá contener además un diagrama con las tablas y relaciones se utilizarán en la base de datos.

- **El trabajo** (código) deberá coincidir concretamente en la lógica y aplicación de los pasos con lo que se haya graficado en el punto anterior.

- **El trabajo** (código) deberá obligatoriamente utilizar un sistema ORM (como Sequelize) para crear las consultas de la base de datos, la cuál, deberá ser MySQL o PostgreSQL de forma obligatoria sin excepciones.

- **El trabajo (código)** debe presentar la sucesión de commits aplicados en el repositorio de código, con los comentarios correspondientes a cada una de las etapas de su desarrollo. Se recomienda (aunque no es obligatorio) seguir la convención de commits que se presenta en este post:https://www.conventionalcommits.org/es/v1.0.0/

Temática a elección:

Aclaraciones:

- Las temáticas son de elección del alumno, pero se deben respetar rigurosamente los requisitos de cada posibilidad presentada. Esto implica que se respetarán tanto las cardinalidades como las funcionalidades mínimas de cada una de las opciones, y estas deberán presentarse de esta manera tanto en los gráficos solicitados en la sección de documentación, como también deberán quedar reflejadas en el código.

- En todos los casos, los atributos se deben tener cada una de las entidades no será especificada pero será tenido en cuenta la capacidad del alumno de abstraer aquellas propiedades que crea necesaria en cada caso para con la nota final.

Temáticas:

Temática 1: "Usuario - Producto - Comentario"
Cada **"Usuario"** deberá poder tener uno o más **"Productos"** por ende deberá de haber una cardinalidad de uno a muchos y que a su vez sea capaz de tener uno o más **"Comentarios"**, por ende también deberá estar relacionada con una cardinalidad de uno a muchos con cada comentario.

Temática 2: "Usuario - Post - Comentario"
Cada **"Usuario"** puede crear uno o más **"Posts"**. Existe una relación de uno a muchos entre **"Usuario"** y **"Post"**. Cada **"Post"** puede tener varios **"Comentarios"**, lo que implica otra relación de uno a muchos entre **"Post"** y **"Comentario"**.

Temática 3: "Usuario - Proyecto - Tarea"
Cada "Usuario" puede estar asociado con uno o más **"Proyectos"**. Esto establece una relación de uno a muchos entre **"Usuario"** y **"Proyecto"**. Cada **"Proyecto"** puede tener varias **"Tareas"**, lo que implica una relación de uno a muchos entre **"Proyecto"** y **"Tarea"**.

Temática 4: "Usuario - Evento - Asistencia"
Cada **"Usuario"** puede crear uno o más **"Eventos"**. Existe una relación de uno a muchos entre **"Usuario"** y **"Evento"**. Cada "Evento" puede tener múltiples **"Asistencias"**, lo que establece una relación de uno a muchos entre **"Evento"** y **"Asistencia"**.

Temática 5: "Usuario - Playlist - Canción"
Cada **"Usuario"** puede crear una o más **"Playlists"**. Existe una relación de uno a muchos entre **"Usuario"** y **"Playlist"**. Cada **"Playlist"** puede contener varias **"Canciones"**, lo que implica una relación de uno a muchos entre **"Playlist"** y **"Canción"**.