«Talento Tech»

Back-End

Java

Clase 15

Buenos Aires Ciudad

Clase 15: Repaso General y Preparación del Proyecto Final

Índice

Objetivos de la Clase

- Reafirmar los conceptos clave del curso.
- Profundizar en las buenas prácticas de programación en Java.
- Resolver dudas o problemas técnicos antes de la presentación del proyecto.
- Brindar pautas para la presentación del proyecto final.
- Garantizar que cada estudiante tenga una implementación funcional y optimizada de su backend.

Repaso de conceptos claves

Estructura y Modularización de un Proyecto en Java

Organizar correctamente el código es fundamental para el mantenimiento y escalabilidad del proyecto. En Java, es recomendable seguir una estructura de paquetes bien definida:

- `com.empresa.proyecto.controller` → Controladores REST.
- `com.empresa.proyecto.service` → Lógica de negocio.
- `com.empresa.proyecto.repository` → Acceso a datos con JPA.
- `com.empresa.proyecto.model` → Entidades y modelos de datos.
- `com.empresa.proyecto.config` → Configuraciones del sistema.

Cada capa del sistema debe estar bien separada y comunicarse a través de interfaces y servicios bien definidos.

Principios de POO: Encapsulamiento, Herencia, Polimorfismo

- **Encapsulamiento**: Controlar el acceso a los datos utilizando modificadores de acceso (`private`, `protected`, `public`).
- **Herencia**: Permite reutilizar código y definir jerarquías de clases.
- **Polimorfismo**: Implementar la misma interfaz de diferentes maneras para mayor flexibilidad.

Ejemplo de herencia y polimorfismo:

```
public abstract class Animal {

   public abstract void hacerSonido();

}

public class Perro extends Animal {

   @Override

   public void hacerSonido() {

       System.out.println("Guau Guau");

   }}
```

Manejo de Base de Datos con JPA y MySQL

Spring Data JPA facilita la interacción con bases de datos relacionales. Se recomienda definir las entidades adecuadamente:

```
@Entity

public class Usuario {

   @Id

   @GeneratedValue(strategy = GenerationType.IDENTITY)

   private Long id;

   private String nombre;

   private String email;

}
```

Para consultas personalizadas se pueden usar `@Query` en los repositorios:

```
@Repository

public interface UsuarioRepository extends JpaRepository<Usuario, Long>
{

   @Query("SELECT u FROM Usuario u WHERE u.email = ?1")

   Usuario encontrarPorEmail(String email);

}
```

Exposición de APIs REST con Spring Boot

Un controlador REST define los endpoints de la API:

```
@RestController

@RequestMapping("/usuarios")

public class UsuarioController {

   @Autowired

   private UsuarioService usuarioService;

   @GetMapping("/{id}")

   public ResponseEntity<Usuario> obtenerUsuario(@PathVariable Long id)
{

       return
ResponseEntity.ok(usuarioService.obtenerUsuarioPorId(id));}}
```

Se recomienda utilizar `ResponseEntity` para manejar respuestas HTTP adecuadamente.

Integración con el Frontend

Para que el frontend pueda consumir la API, es importante permitir CORS y estructurar respuestas JSON consistentes.

Ejemplo de configuración CORS en Spring Boot:

```
@Configuration

public class CorsConfig {

   @Bean

   public WebMvcConfigurer corsConfigurer() {

       return new WebMvcConfigurer() {

           @Override

           public void addCorsMappings(CorsRegistry registry) {


registry.addMapping("/**").allowedOrigins("http://localhost:3000");

           }

       };

   }

}
```

Manejo de Errores y Optimización del Rendimiento

Para evitar errores inesperados y mejorar el rendimiento:

- Utilizar `@ExceptionHandler` para manejar excepciones personalizadas.
- Implementar paginación y caché para consultas grandes.
- Optimizar las consultas a la base de datos con `JOIN FETCH`.
- Aplicar logs con SLF4J y Logback.

Ejemplo de manejo de errores:

```
@ControllerAdvice

public class GlobalExceptionHandler {

   @ExceptionHandler(ResourceNotFoundException.class)

   public ResponseEntity<String>
handleNotFoundException(ResourceNotFoundException e) {

       return
ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());

   }

}
```

Orientaciones para la Presentación del Proyecto

La presentación del proyecto debe cumplir con los siguientes criterios:

1. **Explicación del problema y solución:** Describir la funcionalidad de la aplicación y los objetivos del backend.
2. **Demostración en vivo:** Ejecutar la aplicación, mostrar la API funcionando y su interacción con la base de datos.
3. **Organización del código:** Mostrar estructura de carpetas, modularización y principios aplicados.
4. **Retos y Soluciones:** Explicar los principales desafíos enfrentados y cómo fueron resueltos.

Es fundamental que cada estudiante prepare su presentación con anticipación, asegurándose de que su aplicación esté lista para ser mostrada. Si alguien no está seguro de cómo presentar su trabajo, debe consultar al profesor para recibir orientación y asegurarse de que su demostración cumpla con las expectativas establecidas.

Cierre del Proyecto

¡Ha llegado el momento de la entrega final!

Después de meses de trabajo duro, aprendizaje, análisis y desarrollo de software, ahora deberás consolidar todo el conocimiento adquirido en un proyecto final integrador.

El camino que recorrieron hasta ahora refleja lo que sucede en la industria del desarrollo de software. Empresas como Sibelius, que confían en desarrolladores para construir sus plataformas, esperan que sus equipos entreguen productos funcionales, bien documentados y optimizados. Ahora, en TechLab, la tarea es similar: la entrega final del Trabajo Final Integrador, una API RESTful para un e-commerce.

Deberás demostrar tus habilidades construyendo una API backend en Java y Spring Boot con MySQL, implementando endpoints que permitan gestionar productos, categorías y un carrito de compras. Además, se evaluará la calidad del código, la organización del repositorio, la documentación y la integración con el frontend.

La API debe cumplir con los siguientes aspectos clave:

1. Arquitectura RESTful:
    - Implementación de una API que siga las convenciones REST.
    - Uso correcto de métodos HTTP (GET, POST, PUT, DELETE).
2. Estructura del Proyecto:
    - Organización clara de paquetes (controllers, services, repositories, models).
    - Aplicación de principios de POO y buenas prácticas de programación.
3. Tecnologías a Utilizar:
    - Java 8 o superior.
    - Spring Boot como framework principal.
    - MySQL para la base de datos.
    - Maven para la gestión de dependencias.
    - IntelliJ IDEA o Eclipse como IDE recomendado.
4. Funcionalidades Clave:
    - Productos: Crear, actualizar, listar y eliminar productos.
    - Categorías: Administración de categorías con relación a los productos.
    - Carrito de Compras: Agregar, actualizar y eliminar productos en el carrito.
    - Validaciones y Manejo de Errores: Implementar validaciones para evitar datos incorrectos y manejar errores adecuadamente.
5. Control de Versiones y Documentación:
    - Repositorio en GitHub:
        - Subida del código a un repositorio público o privado.
        - Historial de commits bien documentado.

    - README.md:
        - Explicación clara del proyecto, las tecnologías utilizadas y cómo ejecutarlo.
    - Documentación de la API (Opcional):
        - Uso de Swagger u otra herramienta similar para documentar endpoints.
6. Formato de Entrega y Evaluación:
    - Nombre del repositorio: `proyecto-final-backend-ecommerce-[nombre-apellido]`.
    - Evaluación:
        - Estructura del Código: Claridad, organización y aplicación de principios de POO.
        - Correctitud de los Endpoints: Cumplimiento de los requisitos especificados.
        - Manejo de Errores: Mensajes claros y uso adecuado de códigos HTTP.
        - Integración con el Frontend: Asegurar que el frontend pueda consumir la API correctamente.
        - Control de Versiones y Documentación: Calidad del README y frecuencia de commits.

Este es el momento de demostrar todo lo aprendido.

Preguntas para Reflexionar

- ¿Mi código sigue principios de buenas prácticas?
- ¿Mi API es eficiente y está bien documentada?
- ¿Cómo puedo mejorar la seguridad y rendimiento de mi aplicación?

Buenos Aires Ciudad