TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

PRUEBA Y CALIDAD DE SOFTWARE

CASO PRÁCTICO COMPLETO: PRUEBA DE INTEGRACIÓN ENTRE CAPAS

CASO PRÁCTICO COMPLETO: PRUEBA DE INTEGRACIÓN ENTRE CAPAS

1. Modelo (entidad)

Figura 1. Modelo del proyecto

```java
@Entity
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nombre;
    private String correo;

    // Getters y Setters
}
```

En la imagen anterior se asignan los comentarios de getters and setters para indicar que se deben agregar, no se ponen por cuestión de minimizar la imagen.

2. Repositorio

Figura 2. Repositorio del proyecto

```java
@Repository
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {
    Optional<Usuario> findByCorreo(String correo);
}
```

3. Servicio

Figura 3. Servicio del proyecto

```java
@Service
public class UsuarioService {
    @Autowired
    private UsuarioRepository usuarioRepository;

    public Usuario guardar(Usuario usuario) {
        return usuarioRepository.save(usuario);
    }

    public Optional<Usuario> buscarPorCorreo(String correo) {
        return usuarioRepository.findByCorreo(correo);
    }
}
```

4. Controlador

Figura 4. Controlador del proyecto

```java
@RestController
@RequestMapping("/usuarios")
public class UsuarioController {

    @Autowired
    private UsuarioService usuarioService;

    @PostMapping
    public ResponseEntity<Usuario> crear(@RequestBody Usuario usuario) {
        return ResponseEntity.ok(usuarioService.guardar(usuario));
    }

    @GetMapping("/correo/{correo}")
    public ResponseEntity<Usuario> obtener(@PathVariable String correo) {
        return usuarioService.buscarPorCorreo(correo)
            .map(ResponseEntity::ok)
            .orElse(ResponseEntity.notFound().build());
    }
}
```

Con esto se termina de desarrollar el proyecto, un proyecto sencillo para poder explicar cómo sería una prueba de integración realizada de forma correcta y con un contexto claro.

Ahora lo que se procede es a crear la prueba de integración que permita validar el funcionamiento de todos los componentes del proyecto.

5. Prueba de integración completa

Figura 4. Prueba de Integración del proyecto

Explicación detallada línea por línea:

- @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT): inicia el contexto completo de la aplicación en un puerto aleatorio para que los endpoints REST estén disponibles durante la prueba.
- @Autowired private TestRestTemplate restTemplate;: inyecta un cliente HTTP que permite hacer llamadas REST simuladas al servidor Spring Boot embebido.
- Usuario nuevo = new Usuario();: se crea un nuevo objeto de tipo Usuario.
- nuevo.setNombre("Carlos Pérez");: se asigna el nombre al objeto.
- nuevo.setCorreo("carlos@demo.com");: se asigna el correo al objeto.
- restTemplate.postForEntity(...): se envía una solicitud POST con el objeto Usuario como cuerpo al endpoint /usuarios. Se espera que el backend lo reciba, lo procese y devuelva el objeto guardado.
- assertEquals(HttpStatus.OK, postResponse.getStatusCode());: se valida que la respuesta HTTP sea exitosa (200 OK).
- assertNotNull(postResponse.getBody().getId());: se comprueba que el usuario fue efectivamente guardado y recibió un ID generado por la base de datos.
- restTemplate.getForEntity(...): realiza una solicitud GET al endpoint /usuarios/correo/{correo} con el correo del usuario recién creado para recuperar sus datos.
- assertEquals(HttpStatus.OK, getResponse.getStatusCode());: verifica que la búsqueda fue exitosa.
- assertEquals("Carlos Pérez", getResponse.getBody().getNombre());: compara el nombre del usuario recibido con el esperado, asegurando la consistencia entre los datos creados y recuperados.