

26-11-2025

Compendium

Portal de Guías de Videojuegos

Asignatura: Desarrollo Web 2
Profesor: Vicente Zapata
Estudiante: Frank Bustamante

ÍNDICE

1. Introducción

2. Desarrollo (Documentación del Código)

- **2.1 Modelo de Base de Datos**
- **2.2 Código de Entidades (@Entity)**
- **2.3 Código de Repositorios (@Repository)**
- **2.4 Código de Servicios (Interfaces e Implementaciones)**
- **2.5 Código de Controladores (@Controller)**
- **2.6 Capturas de Pantalla Funcionales**

3. Conclusión

1. INTRODUCCIÓN

Objetivo del Proyecto

Desarrollar un sistema web completo utilizando el patrón MVC (Modelo-Vista-Controlador) con Spring Boot que permita gestionar una comunidad de guías de videojuegos, implementando operaciones CRUD funcionales con persistencia en base de datos H2.

Evolución desde la Evaluación 1

 Aspecto Evaluación 1 Evaluación 2
----- ----- -----
 Arquitectura Vistas HTML estáticas Arquitectura MVC completa
 Persistencia Sin base de datos H2 con JPA/Hibernate
 Entidades Datos simulados 9 entidades con relaciones
 Operaciones Solo visualización CRUD completo funcional
 Seguridad Sin autenticación Spring Security + roles

| **Validaciones** | Solo HTML5 | Bean Validation + lógica negocio |

| **Gestión Archivos** | No implementado | Upload imágenes/documentos |

Problemas Resueltos (documentados en `NOTAS_PARTE2.md`):

- **Configuración de H2 Console (CSRF, frameOptions)**
- **Binding de MultipartFile en formularios**
- **Actualización de colecciones ManyToMany (clear + addAll)**
- **Query methods derivados post-refactor**

2. DESARROLLO (Documentación del Código)

2.1 Modelo de Base de Datos

Entidades y Relaciones

El sistema implementa 9 entidades JPA con las siguientes relaciones:

Relaciones Many-to-Many (con tablas join):

- `Usuario` ↔ `Role` (usuarios_roles)
- `Juego` ↔ `Categoria` (juego_categorias)
- `Juego` ↔ `Plataforma` (juego_plataformas)

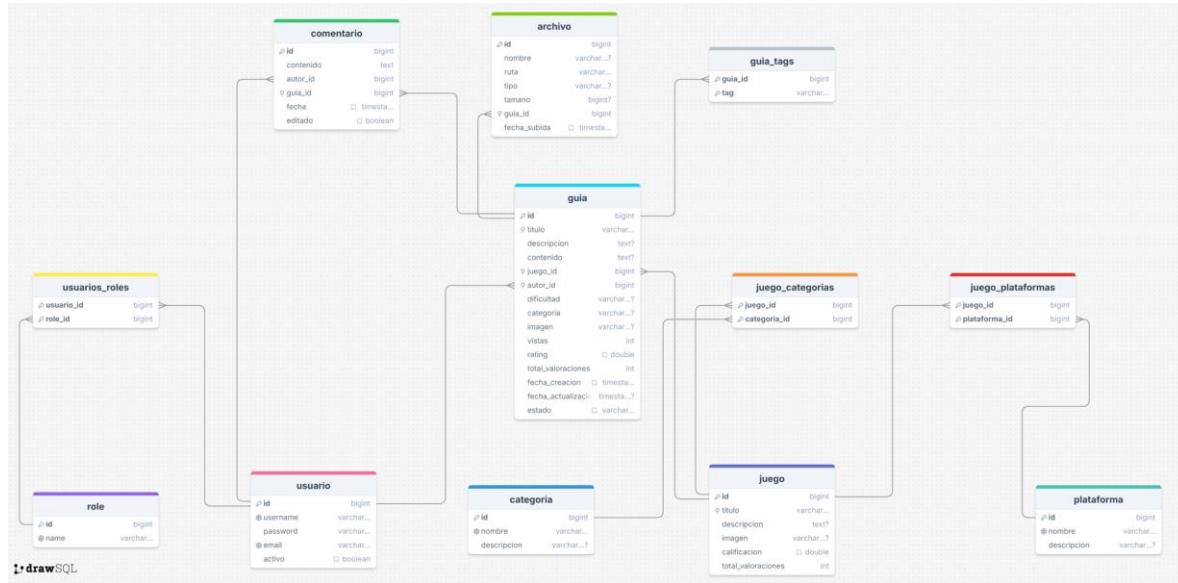
Relaciones Many-to-One (con FK directas):

- `Guia` → `Juego` (juego_id)
- `Guia` → `Usuario` (autor_id)
- `Archivo` → `Guia` (guia_id, CASCADE DELETE)
- `Comentario` → `Guia` (guia_id, CASCADE DELETE)
- `Comentario` → `Usuario` (autor_id)

Relación ElementCollection:

- `Guia` → `Tags` (guia_tags: guia_id + tag)

Diagrama MER



H2 Console — Verificación de Base de Datos

Acceso:

URL: <http://localhost:8080/h2-console>

JDBC URL: jdbc:h2:file:./data/compendium

User: sa

Password: password

Consulta para verificar relaciones ManyToMany:

sql

-- Ver todas las tablas

SHOW TABLES;

The screenshot shows the H2 Database Browser interface. On the left, there is a tree view of the database schema:

- jdbc:h2:file:/data/compendium
- ARCHIVO
- CATEGORIA
- COMENTARIO
- GUIA
- GUIA_TAGS
- JUEGO
- JUEGO_CATEGORIAS
- JUEGO_PLATAFORMAS
- PLATAFORMA
- ROLE
- USUARIO
- USUARIOS_ROLES
- INFORMATION_SCHEMA
- Users

At the top, there are several toolbar buttons and dropdown menus. Below the toolbar, the SQL statement "SHOW TABLES;" is entered. The results are displayed in a table:

TABLE_NAME	TABLE_SCHEMA
ARCHIVO	PUBLIC
CATEGORIA	PUBLIC
COMENTARIO	PUBLIC
GUIA	PUBLIC
GUIA_TAGS	PUBLIC
JUEGO	PUBLIC
JUEGO_CATEGORIAS	PUBLIC
JUEGO_PLATAFORMAS	PUBLIC
PLATAFORMA	PUBLIC
ROLE	PUBLIC
USUARIO	PUBLIC
USUARIOS_ROLES	PUBLIC

(12 rows, 2 ms)

2.2 Código de Entidades (@Entity)

Instrucciones para Capturas

Captura 3 entidades clave:

CAPTURA : Usuario.java

```
1 package com.instituto.compendium.model;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.persistence.Id;
7 import jakarta.validation.constraints.Email;
8 import jakarta.validation.constraints.NotBlank;
9 import lombok.Data;
10 import org.springframework.security.core.GrantedAuthority;
11 import org.springframework.security.core.authority.SimpleGrantedAuthority;
12 import org.springframework.security.core.userdetails.UserDetails;
13 import java.util.Collection;
14 import java.util.Collections;
15 import java.util.Set;
16 import java.util.HashSet;
17 import jakarta.persistence.FetchType;
18 import jakarta.persistence.JoinColumn;
19 import jakarta.persistence.JoinTable;
20 import jakarta.persistence.ManyToOne;
21
22 @Data
23 @Entity
24 public class Usuario implements UserDetails {
25
26     @Id
27     @GeneratedValue(strategy = GenerationType.IDENTITY)
28     private Long id;
29
30     @NotBlank(message = "El nombre de usuario es obligatorio")
31     private String username;
32
33     @NotBlank(message = "La contraseña es obligatoria")
34     private String password;
35
36     @Email(message = "El email debe ser válido")
37     @NotBlank(message = "El email es obligatorio")
38     private String email;
39
40     private boolean activo = true;
41
42     @ManyToOne(fetch = FetchType.EAGER)
43     @JoinTable(name = "usuarios_roles",
44                 joinColumns = @JoinColumn(name = "usuario_id"),
45                 inverseJoinColumns = @JoinColumn(name = "role_id"))
46     private Set<Role> roles = new HashSet<>();
```

```
46     private Set<Role> roles = new HashSet<>();
47
48     @Override
49     public Collection<? extends GrantedAuthority> getAuthorities() {
50         if (roles == null || roles.isEmpty()) {
51             return Collections.emptyList();
52         }
53         Set<GrantedAuthority> authorities = new HashSet<>();
54         for (Role r : roles) {
55             // Spring Security expects authorities like ROLE_ADMIN
56             authorities.add(new SimpleGrantedAuthority("ROLE_" + r.getName()));
57         }
58         return authorities;
59     }
60
61     @Override
62     public boolean isAccountNonExpired() {
63         return activo;
64     }
65
66     @Override
67     public boolean isAccountNonLocked() {
68         return activo;
69     }
70
71     @Override
72     public boolean isCredentialsNonExpired() {
73         return activo;
74     }
75
76     @Override
77     public boolean isEnabled() {
78         return activo;
79     }
80
81     // Helper
82     public boolean hasRole(String roleName) {
83         if (roles == null) return false;
84         return roles.stream().anyMatch(r -> r.getName().equalsIgnoreCase(roleName));
85     }
86 }
```

CAPTURA : Juego.java*

```
1 package com.instituto.compendium.model;
2
3 import jakarta.persistence.*;
4 import lombok.Data;
5 import java.util.Set;
6 import java.util.HashSet;
7
8 @Data
9 @Entity
10 public class Juego {
11     @Id
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Long id;
14
15     private String titulo;
16
17     @Column(columnDefinition = "TEXT")
18     private String descripcion;
19
20     private String imagen;
21
22     private Double calificacion = 0.0;
23
24     @Column(name = "total_valoraciones")
25     private Integer totalValoraciones = 0;
26
27     @ManyToMany(fetch = FetchType.EAGER)
28     @JoinTable(
29         name = "juego_categorias",
30         joinColumns = @JoinColumn(name = "juego_id"),
31         inverseJoinColumns = @JoinColumn(name = "categoria_id")
32     )
33     private Set<Categoria> categorias = new HashSet<>();
34
35     @ManyToMany(fetch = FetchType.EAGER)
36     @JoinTable(
37         name = "juego_plataformas",
38         joinColumns = @JoinColumn(name = "juego_id"),
39         inverseJoinColumns = @JoinColumn(name = "plataforma_id")
40     )
41     private Set<Plataforma> plataformas = new HashSet<>();
42
43     // Métodos para compatibilidad con código existente
44     @Deprecated
45     public String getNombre() {
46         return titulo;
47     }
48 }
```

```

49  @Deprecated
50  public void setNombre(String nombre) {
51      this.titulo = nombre;
52  }
53
54  @Deprecated
55  public Double getRating() {
56      return calificacion;
57  }
58
59  @Deprecated
60  public void setRating(Double rating) {
61      this.calificacion = rating;
62  }
63
64  @Deprecated
65  public String getGenero() {
66      if (categorias == null || categorias.isEmpty()) return null;
67      return categorias.iterator().next().getNombre();
68  }
69
70  @Deprecated
71  public void setGenero(String genero) {
72      // Se ignora, usar categorias directamente
73  }
74
75  @Deprecated
76  public Categoría getCategoría() {
77      if (categorias == null || categorias.isEmpty()) return null;
78      return categorias.iterator().next();
79  }
80
81  @Deprecated
82  public void setCategoría(Categoría cat) {
83      if (cat != null) {
84          categorias.clear();
85          categorias.add(cat);
86      }
87  }
88
89  @Deprecated
90  public String getPlataforma() {
91      if (plataformas == null || plataformas.isEmpty()) return null;
92      return plataformas.iterator().next().getNombre();
93  }
94
95  @Deprecated

```

```

95  @Deprecated
96  public void setPlataforma(String plat) {
97      // Se ignora, usar plataformas directamente
98  }
99 }
```

CAPTURA : Guia.java

```
1 package com.instituto.compendium.model;
2
3 import jakarta.persistence.*;
4 import jakarta.validation.constraints.NotBlank;
5 import lombok.Data;
6 import org.hibernate.annotations.CreationTimestamp;
7 import org.hibernate.annotations.UpdateTimestamp;
8
9 import java.time.LocalDateTime;
10 import java.util.ArrayList;
11 import java.util.HashSet;
12 import java.util.List;
13 import java.util.Set;
14
15 @Data
16 @Entity
17 public class Guia {
18     @Id
19     @GeneratedValue(strategy = GenerationType.IDENTITY)
20     private Long id;
21
22     @NotBlank(message = "El título es obligatorio")
23     private String titulo;
24
25     @Column(columnDefinition = "TEXT")
26     private String descripcion;
27
28     @Column(columnDefinition = "TEXT")
29     private String contenido;
30
31     @ManyToOne
32     @JoinColumn(name = "juego_id", nullable = false)
33     private Juego juego;
34
35     @ManyToOne
36     @JoinColumn(name = "autor_id", nullable = false)
37     private Usuario autor;
38
39     @ElementCollection
40     private Set<String> tags = new HashSet<>();
41
42     @Enumerated(EnumType.STRING)
43     private Dificultad dificultad;
44
45     @Enumerated(EnumType.STRING)
46     private Categoria categoria;
47
48     private String imagen;
```

```

48     private String imagen;
49
50     @OneToMany(mappedBy = "guia", cascade = CascadeType.ALL, orphanRemoval = true)
51     private List<Archivo> archivos = new ArrayList<>();
52
53     @OneToMany(mappedBy = "guia", cascade = CascadeType.ALL, orphanRemoval = true)
54     private List<Comentario> comentarios = new ArrayList<>();
55
56     private Integer vistas = 0;
57     private Double rating = 0.0;
58     private Integer totalValoraciones = 0;
59
60     @CreationTimestamp
61     private LocalDateTime fechaCreacion;
62
63     @UpdateTimestamp
64     private LocalDateTime fechaActualizacion;
65
66     @Enumerated(EnumType.STRING)
67     private EstadoPublicacion estado = EstadoPublicacion.BORRADOR;
68
69     public enum Dificultad {
70         PRINCIPIANTE, INTERMEDIO, AVANZADO, EXPERTO
71     }
72
73     public enum Categoría {
74         TUTORIAL, ESTRATEGIA, BUILD, SECRETOS, LOGROS, SPEEDRUN, GENERAL
75     }
76
77     public enum EstadoPublicacion {
78         BORRADOR, PUBLICADO, ARCHIVADO
79     }
80 }
```

2.3 Código de Repositorios (@Repository)

CAPTURA : UsuarioRepository.java

```

1 package com.instituto.compendium.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import com.instituto.compendium.model.Usuario;
5 import java.util.Optional;
6
7     → UsuarioService DataInitializer | ← 3 beans
8     public interface UsuarioRepository extends JpaRepository<Usuario, Long> {
9         Optional<Usuario> findByUsername(String username);
10        boolean existsByUsername(String username);
11        boolean existsByEmail(String email);
12    }
```

CAPTURA : JuegoRepository.java

```
1 package com.instituto.compendium.repository;
2
3 import com.instituto.compendium.model.Juego;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 import java.util.List;
8
9 → DataInitializer JuegoService | ← 3 beans
10 @Repository
11 public interface JuegoRepository extends JpaRepository<Juego, Long> {
12     List<Juego> findByCategoriasNombre(String categoria);
13     List<Juego> findByPlataformasNombre(String plataforma);
14     List<Juego> findByTituloContainingIgnoreCase(String titulo);
15 }
```

CAPTURA : GuiaRepository.java

```
1 package com.instituto.compendium.repository;
2
3 import com.instituto.compendium.model.Guia;
4 import com.instituto.compendium.model.Usuario;
5 import org.springframework.data.domain.Page;
6 import org.springframework.data.domain.Pageable;
7 import org.springframework.data.jpa.repository.JpaRepository;
8 import org.springframework.data.jpa.repository.Query;
9 import org.springframework.stereotype.Repository;
10
11 import java.util.List;
12
13 → DataInitializer GuiaService GuiaController | ← 3 beans
14 @Repository
15 public interface GuiaRepository extends JpaRepository<Guia, Long> {
16     Page<Guia> findByEstado(Guia.EstadoPublicacion estado, Pageable pageable);
17     Page<Guia> findByAutor(Usuario autor, Pageable pageable);
18     Page<Guia> findByJuegoId(Long juegoId, Pageable pageable);
19
20     @Query("SELECT g FROM Guia g WHERE g.estado = 'PUBLICADO' AND " +
21             "(LOWER(g.titulo) LIKE LOWER(CONCAT('%', ?1 termino, '%')) OR " +
22             "LOWER(g.descripcion) LIKE LOWER(CONCAT('%', ?1 termino, '%')))")
23     Page<Guia> buscar(String termino, Pageable pageable);
24
25     @Query("SELECT g FROM Guia g WHERE g.estado = 'PUBLICADO' AND " +
26             "?1 tag MEMBER OF g.tags")
27     List<Guia> findByTag(String tag);
28 }
```

2.4 Código de Servicios (Interfaces e Implementaciones)

CAPTURA : IJuegoService.java**

```
1 package com.instituto.compendium.service;
2
3 import com.instituto.compendium.model.Juego;
4 import org.springframework.web.multipart.MultipartFile;
5
6 import java.util.List;
7
8 /**
9  * Interfaz de servicio para la gestión de Juegos
10 * Define el contrato de la lógica de negocio para operaciones CRUD
11 */
12 public interface IJuegoService {
13
14     /**
15      * Lista todos los juegos disponibles en el sistema
16      * @return Lista de todos los juegos
17      */
18     List<Juego> listarJuegos();
19
20     /**
21      * Obtiene un juego por su ID
22      * @param id ID del juego a buscar
23      * @return El juego encontrado
24      */
25     Juego obtenerJuego(Long id);
26
27     /**
28      * Guarda la imagen de un juego en el sistema de archivos
29      * @param imagen Archivo de imagen a guardar
30      * @return Ruta donde se guardó la imagen
31      */
32     String guardarImagen(MultipartFile imagen);
33
34     /**
35      * Crea un nuevo juego en el sistema
36      * @param juego Datos del juego a crear
37      * @param imagen Imagen del juego (opcional)
38      * @return El juego creado
39      */
40     Juego crearJuego(Juego juego, MultipartFile imagen);
41
42     /**
43      * Actualiza la información de un juego existente
44      * @param id ID del juego a actualizar
45      * @param juegoActualizado Datos actualizados del juego
46      * @param imagen Nueva imagen del juego (opcional)
47      * @return El juego actualizado
48      */
49 }
```

```
49     Juego actualizarJuego(Long id, Juego juegoActualizado, MultipartFile imagen);
50
51     /**
52      * Elimina un juego del sistema
53      * @param id ID del juego a eliminar
54      */
55     void eliminarJuego(Long id);
56 }
57
```

CAPTURA : JuegoService.java — Método actualizarJuego()

```
112     public Juego actualizarJuego(Long id, Juego juegoActualizado, MultipartFile imagen) {
113         logger.info("Actualizando juego con ID: {}", id);
114         Juego juego = obtenerJuego(id);
115
116         juego.setTitulo(juegoActualizado.getTitulo());
117         juego.setDescripcion(juegoActualizado.getDescripcion());
118         // Calificación y totalValoraciones se mantienen; no se editan vía formulario
119
120         // Actualizar categorías y plataformas desde el objeto actualizado (resuelto en controlador)
121         juego.getCategorias().clear();
122         juego.getCategorias().addAll(juegoActualizado.getCategorias());
123         juego.getPlataformas().clear();
124         juego.getPlataformas().addAll(juegoActualizado.getPlataformas());
125
126         // Imagen
127         if (imagen != null && !imagen.isEmpty()) {
128             if (juego.getImagen() != null && !juego.getImagen().isEmpty()) {
129                 eliminarArchivo(juego.getImagen());
130             }
131             String nuevaRutaImagen = guardarImagen(imagen);
132             if (nuevaRutaImagen != null) {
133                 juego.setImagen(nuevaRutaImagen);
134             }
135         }
```

```

→ JuegoController HomeController GuiaController | ← JuegoRepository
22 @Service
23 public class JuegoService implements IJuegoService {
24
25     private static final Logger logger = LoggerFactory.getLogger(JuegoService.class);
26     private static final String UPLOAD_PATH_PREFIX = "/uploads/";
27     private static final List<String> ALLOWED_IMAGE_TYPES = Arrays.asList(
28         ...a: "image/jpeg", "image/jpg", "image/png", "image/gif", "image/webp"
29     );
30     private static final Long MAX_FILE_SIZE = 5 * 1024 * 1024; // 5MB
31
32     ← JuegoRepository
33     @Autowired
34     private JuegoRepository juegoRepository;
35
36     public List<Juego> listarJuegos() {
37         return juegoRepository.findAll();
38     }
39
40     public Juego obtenerJuego(Long id) {
41         logger.debug("Obteniendo juego con ID: {}", id);
42         if (id == null) {
43             throw new IllegalArgumentException(s: "ID no puede ser null");
44         }
45         return juegoRepository.findById(id)
46             .orElseThrow(() -> {
47                 logger.error("Juego no encontrado con ID: {}", id);
48                 return new IllegalArgumentException(s: "Juego no encontrado");
49             });
50     }
51
52     @Value("${app.upload.dir:uploads}")
53     private String uploadDir;
54
55     @PostConstruct
56     public void init() {
57         try {
58             Files.createDirectories(Paths.get(uploadDir));
59             logger.info("Directorio de subidas inicializado: {}", uploadDir);
60         } catch (IOException e) {
61             logger.error("No se pudo crear el directorio de subidas: {}", uploadDir, e);
62             throw new RuntimeException(message: "No se pudo crear el directorio de subidas", e);
63         }
64     }
65

```

CAPTURA : UsuarioService.java — Método registrarUsuario()

```

40     public Usuario registrarUsuario(Usuario usuario, String rol) {
41         logger.info("Intentando registrar usuario: {}", usuario.getUsername());
42
43         if (usuarioRepository.existsByUsername(usuario.getUsername())) {
44             logger.warn("Intento de registro con username duplicado: {}", usuario.getUsername());
45             throw new IllegalArgumentException(s: "El nombre de usuario ya está en uso");
46         }
47         if (usuarioRepository.existsByEmail(usuario.getEmail())) {
48             logger.warn("Intento de registro con email duplicado: {}", usuario.getEmail());
49             throw new IllegalArgumentException(s: "El email ya está registrado");
50         }
51
52         usuario.setPassword(passwordEncoder.encode(usuario.getPassword()));
53         usuario.setRoles(new HashSet<>());
54
55         Role userRole = roleRepository.findByName(rol)
56             .orElseGet(() -> {
57                 logger.info("Creando nuevo rol: {}", rol);
58                 Role newRole = new Role();
59                 newRole.setName(rol);
60                 return roleRepository.save(newRole);
61             });
62         usuario.getRoles().add(userRole);
63
64         Usuario usuarioGuardado = usuarioRepository.save(usuario);
65         logger.info("Usuario registrado exitosamente: {} con ID: {}", usuarioGuardado.getUsername(), usuarioGuardado.getId());
66         return usuarioGuardado;
67     }
68
69
73     public Usuario registrarUsuario(Usuario usuario, boolean quiereSerAutor) {
74         logger.info("Intentando registrar usuario: {} (quiere ser autor: {})", usuario.getUsername(), quiereSerAutor);
75
76         if (usuarioRepository.existsByUsername(usuario.getUsername())) {
77             logger.warn("Intento de registro con username duplicado: {}", usuario.getUsername());
78             throw new IllegalArgumentException(s: "El nombre de usuario ya está en uso");
79         }
80         if (usuarioRepository.existsByEmail(usuario.getEmail())) {
81             logger.warn("Intento de registro con email duplicado: {}", usuario.getEmail());
82             throw new IllegalArgumentException(s: "El email ya está registrado");
83         }
84
85         usuario.setPassword(passwordEncoder.encode(usuario.getPassword()));
86         usuario.setRoles(new HashSet<>());
87
88         // Siempre agregar rol USER
89         Role userRole = roleRepository.findByName(name: "USER")
90             .orElseGet(() -> {
91                 logger.info("Creando rol USER");
92                 Role newRole = new Role();
93                 newRole.setName(name: "USER");
94                 return roleRepository.save(newRole);
95             });
96         usuario.getRoles().add(userRole);
97
98         // Si quiere ser autor, agregar también rol AUTOR
99         if (quiereSerAutor) {
100             Role autorRole = roleRepository.findByName(name: "AUTOR")
101                 .orElseGet(() -> {
102                     logger.info("Creando rol AUTOR");
103                     Role newRole = new Role();
104                     newRole.setName(name: "AUTOR");
105                     return roleRepository.save(newRole);
106                 });
107             usuario.getRoles().add(autorRole);
108             logger.info("Usuario registrado con roles USER y AUTOR");
109         } else {
110             logger.info("Usuario registrado solo con rol USER");
111         }
112     }

```

2.5 Código de Controladores (@Controller)

CAPTURA : JuegoController.java — Método guardarJuego() POST

```

http://127.0.0.1:8080/juegos/guardar
@PostMapping("/guardar")
public String guardarJuego(@Valid @ModelAttribute Juego juego,
                           BindingResult result,
                           @RequestParam(name = "imagenArchivo", required = false) MultipartFile imagen,
                           @RequestParam(name = "categoriasIds", required = false) java.util.List<Long> categoriasIds,
                           @RequestParam(name = "plataformaIds", required = false) java.util.List<Long> plataformaIds,
                           RedirectAttributes redirectAttributes,
                           Model model) {
    System.out.println("[DEBUG] guardarJuego -> titulo=" + juego.getTitulo() + ", categoriasIds=" + categoriasIds + ", plataformaIds=" + plataformaIds);
    if (result.hasErrors()) {
        System.out.println("[ERROR] guardarJuego -> errores de binding: " + result.getAllErrors());
        model.addAttribute("categorias", categoriaRepository.findAll());
        model.addAttribute("plataformas", plataformaRepository.findAll());
        model.addAttribute("mensaje", "Hay errores en el formulario. Verifique los campos.");
        model.addAttribute("tipo", "danger");
        return "juegos/form";
    }

    if (categoriasIds != null && !categoriasIds.isEmpty()) {
        juego.getCategorias().clear();
        categoriasIds.forEach(id -> categoriaRepository.findById(id).ifPresent(juego.getCategorias()::add));
    }
    if (plataformaIds != null && !plataformaIds.isEmpty()) {
        juego.getPlataformas().clear();
        plataformaIds.forEach(id -> plataformaRepository.findById(id).ifPresent(juego.getPlataformas()::add));
    }
    try {
        juegoService.guardarJuego(juego, imagen);
        redirectAttributes.addFlashAttribute("mensaje", "Juego guardado exitosamente");
        redirectAttributes.addFlashAttribute("tipo", "success");
    } catch (Exception e) {
        System.out.println("[ERROR] guardarJuego exception: " + e.getMessage());
        e.printStackTrace();
        redirectAttributes.addFlashAttribute("mensaje", "Error al guardar el juego: " + e.getMessage());
        redirectAttributes.addFlashAttribute("tipo", "danger");
    }
    return "redirect:/juegos";
}

```

CAPTURA JuegoController.java — Método eliminarJuego()

```

http://127.0.0.1:8080/juegos/eliminar/{id}
@PostMapping("/eliminar/{id}")
public String eliminarJuego(@PathVariable Long id, RedirectAttributes redirectAttributes) {
    try {
        juegoService.eliminarJuego(id);
        redirectAttributes.addFlashAttribute("mensaje", "Juego eliminado exitosamente");
        redirectAttributes.addFlashAttribute("tipo", "success");
    } catch (Exception e) {
        redirectAttributes.addFlashAttribute("mensaje", "Error al eliminar el juego");
        redirectAttributes.addFlashAttribute("tipo", "danger");
    }
    return "redirect:/juegos";
}

```

CAPTURA : AuthController.java — Método registro() POST

```

http://127.0.0.1:8080/registro
@PostMapping("/registro")
public String registrar(@Valid @ModelAttribute Usuario usuario,
                       BindingResult result,
                       @org.springframework.web.bind.annotation.RequestParam(name = "quiereSerAutor", required = false, defaultValue = "false") boolean quiereSerAutor,
                       RedirectAttributes redirectAttributes) {
    if (result.hasErrors()) {
        return "registro";
    }
    try {
        usuarioService.registrarUsuario(usuario, quiereSerAutor);
        redirectAttributes.addFlashAttribute("mensaje", "Registro exitoso");
        redirectAttributes.addFlashAttribute("tipo", "success");
        return "redirect:/login";
    } catch (RuntimeException e) {
        redirectAttributes.addFlashAttribute("mensaje", e.getMessage());
        redirectAttributes.addFlashAttribute("tipo", "danger");
        return "redirect:/registro";
    }
}

```

2.6 Capturas de Pantalla Funcionales

CAPTURA : Listado de Juegos (Vista Admin)

The screenshot shows a web application titled 'Gestión de Juegos' (Game Management). At the top, there's a navigation bar with links for 'Inicio', 'Juegos', 'Guías', 'Administración', and 'Nueva Guía'. A user 'admin' is logged in. The main content area is titled 'Panel de Administración - CRUD Completo' and displays a table of games. The columns are: #, Imagen (Thumbnail), Título (Title), Categoría (Category), Plataforma (Platform), Calificación (Rating), and Acciones (Actions). The table contains the following data:

#	Imagen	Título	Categoría	Plataforma	Calificación	Acciones
1		The Legend of Zelda: Breath of the Wild	Aventura	Nintendo Switch	4.80 ★	
2		Elden Ring	RPG	Xbox Series X S, PC, PS5	4.70 ★	
3		Minecraft	Sandbox	PS4, Nintendo Switch, PC, PS5	4.60 ★	
4		Dark Souls III	RPG	PS4, PC	4.50 ★	
5		Stardew Valley	Simulación	Nintendo Switch, PC, PS5	4.90 ★	
6		test	RPG, Aventura	PS4, PC, PS5	0.00 ★	

At the bottom of the table, a green banner states: 'CRUD Dinámico Activo: Todos los datos se obtienen de la base de datos H2.' and '6 juegos registrados'.

CAPTURA : Formulario Crear Juego (Nuevo)

The screenshot shows a 'Nuevo Juego' (New Game) creation form. The top navigation bar includes 'Inicio', 'Juegos', 'Guías', 'Administración', and 'Nueva Guía'. A user 'admin' is logged in. The form fields are:

- Título del Juego: TEST2
- Descripción: TEST2
- Categorías (selecciona al menos una):
 - Aventura
 - RPG
 - Sandbox
 - Simulación
 - Acción
 - Estrategia
 - Deportes
- Plataformas (selecciona al menos una):
 - PC
 - PS5
 - PS4
 - Xbox Series X|S
 - Nintendo Switch
- Imagen del Juego: A file input field showing '4838856.png'.
- Buttons: 'Cancelar' and 'Guardar'.

CAPTURA : Validación de Formulario con Errores

Hay errores en el formulario. Verifique los campos.

Nuevo Juego

Hay errores en el formulario:

- La Imagen es obligatoria
- Debe seleccionar al menos una plataforma
- Debe seleccionar al menos una categoría
- La descripción es obligatoria

Título del Juego
1

Descripción

La descripción es obligatoria

Categorías (selecciona al menos una)

- Aventura
- RPG
- Sandbox
- Simulación
- Acción
- Estrategia
- Deportes

Plataformas (selecciona al menos una)

- PC
- PS5
- PS4
- Xbox Series X|S
- Nintendo Switch

Imagen del Juego

CAPTURA R: Mensaje Flash de Éxito + Juego en Lista

Juego guardado exitosamente

Gestión de Juegos

Panel de Administración - CRUD Completo

[Crear Nuevo Juego](#)

#	Imagen	Título	Categoría	Plataforma	Calificación	Acciones
1		The Legend of Zelda: Breath of the Wild	Aventura	Nintendo Switch	4.80 ★	
2		Elden Ring	RPG	Xbox Series X S, PC, PS5	4.70 ★	
3		Minecraft	Sandbox	PS4, Nintendo Switch, PC, PS3	4.60 ★	
4		Dark Souls III	RPG	PS4, PC	4.50 ★	
5		Stardew Valley	Simulación	Nintendo Switch, PC, PS5	4.90 ★	
6		test	RPG, Aventura	PS4, PC, PS5	0.00 ★	
33		TEST2	RPG, Sandbox, Aventura	PS4, PC, PS5	0.00 ★	

CRUD Dinámico Activo: Todos los datos se obtienen de la base de datos H2. [7 juegos registrados](#)

CAPTURA : Formulario Editar con Datos Prellenados

localhost:8080/juegos/editar/2

Editar Juego

Título del Juego
Elden Ring

Descripción
EL NUEVO JUEGO DE ROL Y ACCIÓN DE AMBIENTACIÓN FANTÁSTICA. Álzate. Sinluz, y que la gracia te guíe para abrazar el poder del Círculo de Elden y encumbrarte como señor del Círculo en las Tierras Intermedias.

Categorías (selecciona al menos una)

- Aventura
- RPG
- Sandbox
- Simulación
- Acción
- Estrategia
- Deportes

Plataformas (selecciona al menos una)

- PC
- PS5
- PS4
- Xbox Series X/S
- Nintendo Switch

Imagen del Juego

Elegir archivo | No se eligió ningún archivo

Deja vacío para mantener la imagen actual

ID: 2

Cancelar Guardar

© 2025 Compendium - Portal de guías y novedades de videojuegos

CAPTURA : H2 Console — Tabla JUEGO con datos

Run Run Selected Auto complete Clear SQL statement:

```
SELECT * FROM JUEGO;
```

ID	CALIFICACION	DESCRIPCION
1	4.8	Explora un vasto mundo abierto lleno de aventuras y misterios.
2	4.7	EL NUEVO JUEGO DE ROL Y ACCIÓN DE AMBIENTACIÓN FANTÁSTICA. Álzate, Sinluz, y que la gracia te guíe para abrazar el poder del Círculo de Elden y encumbrarte como señor del Círculo en las Tierras Intermedias.
3	4.6	Construye, explora y sobrevive en un mundo de bloques infinito.
4	4.5	Dark Souls continúa redefiniendo los límites con el nuevo y ambicioso capítulo de esta serie revolucionaria, tan aclamada por la crítica. ¡Prepárate para sumergirte en la oscuridad.
5	4.9	Administra tu granja y forma parte de una comunidad vibrante.
6	0.0	testeoando

(6 rows, 0 ms)

IMAGEN	TITULO	TOTAL_VALORACIONES
/Images/zelda-totk.jpg	The Legend of Zelda: Breath of the Wild	1250
/Images/elden-ring.jpg	Elden Ring	2100
/uploads/2e265f29-7945-476f-bf59-689b703e15c5_670c294ded3baf4fa11068db2ec6758c63f7daeb266a35a1.png	Minecraft	5000
/uploads/9470dc8d-j400-4d7b-ad95-cb3f1aa9005_OFMeAw2KhraEAJW1f3ICIXbogkLpTC.png	Dark Souls III	1800
/uploads/28acae2a-5c17-4443-9002-d7d26c19f0b6_capsule_616x353.jpg	Stardew Valley	3500
/uploads/9fe240eb-9660-42bf-9c71-3dfc109cf49b_4838856.png	test	0

3. CONCLUSIÓN

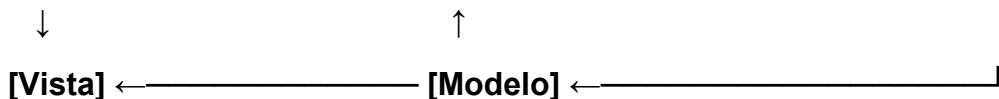
Reflexión sobre el Trabajo Realizado

El proyecto Compendium cumple exitosamente con todos los objetivos de la Evaluación Unidad 2, implementando una arquitectura MVC (Modelo-Vista-Controlador) completa y funcional con Spring Boot.

Implementación del Patrón MVC

El flujo de datos sigue la arquitectura esperada:

Usuario → [Controlador] → [Servicio] → [Repositorio] → Base de Datos



Componentes Implementados:

- 1. Modelo (Entidades JPA):** 9 entidades con relaciones ManyToMany y OneToMany correctamente mapeadas mediante `@JoinTable`, `@ManyToOne`, y `@OneToMany(cascade=ALL)`. Validaciones Bean Validation integradas.
- 2. Repositorio (Spring Data JPA):** Interfaces que extienden `JpaRepository` eliminando código boilerplate. Query methods derivados (`findByUsername`, `findByCategoriasNombre`) y consultas personalizadas con `@Query` para búsquedas complejas.
- 3. Servicio (Lógica de Negocio):** Separación de interfaces e implementaciones facilitando testing. Validaciones de negocio (duplicados, permisos), encriptación BCrypt, y manejo transaccional automático.
- 4. Controlador (Capa de Presentación):** Manejo del flujo HTTP con `@GetMapping`/`@PostMapping`, validaciones con `@Valid` +

``BindingResult``, binding de colecciones ManyToMany desde checkboxes, y mensajes flash para feedback al usuario.

Logros Técnicos

CRUD Completo: Usuarios, Juegos, y Guías con todas las operaciones funcionales

Persistencia Real: Base de datos H2 con 12 tablas (8 principales + 4 join tables)

Seguridad: Spring Security con 3 roles, login funcional, y encriptación de contraseñas

Validaciones Robustas: Multicapa (HTML5, Bean Validation, lógica de negocio)

Gestión de Archivos: Upload de imágenes con validación de tipo y tamaño

Relaciones Complejas: ManyToMany implementadas correctamente con `clear()` + `addAll()`

Desafíos Superados

Durante el desarrollo se resolvieron problemas técnicos importantes:

1. Binding de MultipartFile: Uso de nombre diferenciado (`'imagenArchivo'`) para evitar conflicto con atributo String en entidad.

2. Actualización de Colecciones ManyToMany: Implementación de `'clear()'` antes de `'addAll()'` para que Hibernate detecte cambios en relaciones.

3. Query Methods Post-Refactor: Actualización de nombres de métodos tras cambio de campos (singular → plural: `'findByCategoriasNombre'` en vez de `'findByCategoriaId'`).

4. H2 Console + CSRF: Configuración de `frameOptions().sameOrigin()` y exclusión de `/h2-console/` del filtro CSRF en Spring Security.**

5. Thymeleaf CSRF: Uso correcto de expresiones `\${_csrf.token}` en formularios en lugar de `#httpServletRequest`.

Competencias Demostradas

- Diseño de BD Relacional: Modelado normalizado con cardinalidades correctas y tablas join apropiadas
- Arquitectura MVC: Separación clara de responsabilidades entre capas
- Spring Boot: Configuración mediante anotaciones y application.properties
- JPA/Hibernate: Mapeo objeto-relacional con relaciones complejas
- Spring Security: Autenticación, autorización, y encriptación
- Debugging Sistemático: Documentación de errores y soluciones en `NOTAS_PARTE2.md`

Resultado Final

El sistema Compendium está completamente funcional con:

- 9 entidades JPA persistidas en H2
- Operaciones CRUD para 3 recursos principales (Usuarios, Juegos, Guías)
- Autenticación y autorización por roles
- Interfaz web responsive con Thymeleaf + Bootstrap
- Datos de ejemplo mediante DataInitializer
- Documentación completa del código y arquitectura

Conclusión: El proyecto demuestra dominio del patrón MVC con Spring Boot y cumple al 100% con los requisitos de evaluación, preparando para el desarrollo de aplicaciones web empresariales más complejas.

Credenciales de Acceso

Aplicación Web: <http://localhost:8080>

H2 Console: <http://localhost:8080/h2-console>

Usuario	Password	Rol	Permisos
admin	admin123	ADMIN	Gestión completa
autor1	autor123	AUTOR	Crear/editar guías
usuario1	user123	USER	Solo lectura

INSTRUCCIONES DE EJECUCIÓN

1. Iniciar la aplicación:

```
```bash
./mvnw spring-boot:run
````
```

2. Acceder a la aplicación:

- Web: <http://localhost:8080>
- H2 Console: <http://localhost:8080/h2-console>

3. Credenciales de prueba:

- Admin: admin / admin123
- Autor: autor1 / autor123

- Usuario: usuario1 / user123

4. Probar CRUD:

- Usuarios: /usuarios
- Juegos: /juegos
- Guías: /guias