Fragmentos de Código para la Defensa

CUU - Publicar nuevo material

Entities

Material.java

```java
package com.particulares.tp.java.entities;

import jakarta.persistence.Basic;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.FetchType;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.Lob;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Table;

@Entity
@Table(name = "material")
public class Material {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String descripcion;
    @Lob
    @Basic(fetch = FetchType.LAZY)
    @Column(columnDefinition = "LONGBLOB")
    private byte[] archivo;

    @ManyToOne
    @JoinColumn(name = "dictado_clase_id", nullable = false)
    private DictadoClase dictadoClase;

    //getters y setters
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
```

```
    public String getDescripcion() {
        return descripcion;
    }
    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }
    public byte[] getArchivo() {
        return archivo;
    }
    public void setArchivo(byte[] archivo) {
        this.archivo = archivo;
    }
    public DictadoClase getDictadoClase() {
        return dictadoClase;
    }
    public void setDictadoClase(DictadoClase dictadoClase) {
        this.dictadoClase = dictadoClase;
    }

}
```

DictadoClase.java

```
package com.particulares.tp.java.entities;

import java.util.List;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;

@Entity
@Table(name = "dictado_clase")
public class DictadoClase {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Integer id;

    @ManyToOne
    @JoinColumn(name = "profesor_id")
    private Profesor profesor;

    @ManyToOne
    @JoinColumn(name = "nivel_id")
    private Nivel nivel;


    @ManyToOne
    @JoinColumn(name = "materia_id")
    private Materia materia;

    @OneToMany(mappedBy = "dictadoClase", cascade = CascadeType.ALL,
orphanRemoval = true)
    private List<Material> materiales;


    // Getters y setters

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public Profesor getProfesor() {
        return profesor;
    }

    public void setProfesor(Profesor profesor) {
        this.profesor = profesor;
    }

    public Nivel getNivel() {
        return nivel;
    }

    public void setNivel(Nivel nivel) {
```

```
        this.nivel = nivel;
    }

    public Materia getMateria() {
        return materia;
    }

    public void setMateria(Materia materia) {
        this.materia = materia;
    }
}
```

Repository

DictadoClaseRepository.java

```
    @Query("SELECT DC FROM DictadoClase DC WHERE DC.profesor.id = ?1")
    List <DictadoClase> findByProfesor(int idProfesor);
```

MaterialRepository.java

```
package com.particulares.tp.java.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import com.particulares.tp.java.entities.Material;

@Repository
public interface MaterialRepository extends JpaRepository<Material,
Integer> {

    @Query("SELECT m FROM Material m WHERE m.dictadoClase.profesor.id =
?1")
    List<Material> findByProfesorId(int idProfesor);
}
```

Service

MaterialService,java

```
package com.particulares.tp.java.service;
```

```java
import java.util.List;
import java.util.Optional;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.multipart.MultipartFile;

import com.particulares.tp.java.entities.Material;
import com.particulares.tp.java.entities.DictadoClase;
import com.particulares.tp.java.repository.MaterialRepository;
import com.particulares.tp.java.repository.DictadoClaseRepository;


@Service
public class MaterialService {
    @Autowired
    private MaterialRepository materialRepository;

    @Autowired
    private DictadoClaseRepository dictadoClaseRepository;

    @Transactional
    public void crearmaterial(String[] descripciones, int
idDictadoClase,  MultipartFile[] archivos) throws Exception {


        DictadoClase dictadoClase =
dictadoClaseRepository.findById(idDictadoClase).get();

        if (dictadoClase == null) {
            throw new Exception("El dictadoClase especificado no
existe.");
        }

        for (int i = 0; i < archivos.length; i++) {
            MultipartFile archivo = archivos[i];
            String descripcion = descripciones[i];
            validar(descripcion, idDictadoClase, archivo);
```

```
            if (!archivo.isEmpty()) {
                Material material = new Material();

                material.setDescripcion(descripcion);
                material.setDictadoClase(dictadoClase);
                material.setArchivo(archivo.getBytes());

                materialRepository.save(material);
            }
        }

    }

    @Transactional(readOnly = true)
    public List<Material> listarMaterialesPorProfesor(int idProfesor) {
        return materialRepository.findByProfesorId(idProfesor);
    }

    @Transactional(readOnly = true)
    public Material getOne(int id){
        return materialRepository.getReferenceById(id);
    }

    private void validar(String descripcion, int idDictadoClase,
MultipartFile archivo) throws Exception {
        if (descripcion.isEmpty() || descripcion == null) {
            throw new Exception("la descripcion no puede ser nulo o
estar vacío");
        }
        if (idDictadoClase <= 0) {
            throw new Exception("El idDictadoClase debe ser un numero
positivo");
        }
        if (archivo == null && archivo.isEmpty()) {
            throw new Exception("Debe proporcionar un archivo");
        }
    }

}
```

DictadoClaseService.java

```
    @Transactional(readOnly = true)
```

```
    public List<DictadoClase> obtenerDictadosPorProfesor(Integer
idProfesor){
        return dictadoClaseRepository.findByProfesor(idProfesor);
    }
```

Controller

MaterialController.java

```
package com.particulares.tp.java.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ContentDisposition;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.particulares.tp.java.entities.Material;
import com.particulares.tp.java.entities.Persona;
import com.particulares.tp.java.entities.Profesor;
import com.particulares.tp.java.service.DictadoClaseService;
import com.particulares.tp.java.service.MaterialService;
import com.particulares.tp.java.service.ProfesorService;

import jakarta.servlet.http.HttpSession;

@Controller
@RequestMapping("/material")
public class MaterialController {

    @Autowired
    private MaterialService materialService;
```

```
    @Autowired
    private DictadoClaseService dictadoClaseService;

    @Autowired
    private ProfesorService profesorService;

    @GetMapping("/registrar")
    public String registrar(HttpSession session,
                            ModelMap model) {
        Profesor profesor = (Profesor)
session.getAttribute("personaSession");
        model.put("dictados",
dictadoClaseService.obtenerDictadosPorProfesor(profesor.getId()));
        model.put("profesor", profesor);
        return "material/crear";
    }

    @PostMapping("/registro")
    public String registrarDocumentos(@RequestParam MultipartFile[]
archivos,
                                      @RequestParam String[]
descripciones,
                                      @RequestParam int idDictadoClase,
                                      RedirectAttributes
redirectAttributes, HttpSession session) {
        try {
            materialService.crearmaterial(descripciones,
idDictadoClase, archivos);

            redirectAttributes.addFlashAttribute("exito", "Documentos
cargados correctamente.");
                return "redirect:/material/listar/" + ((Profesor)
session.getAttribute("personaSession")).getId();
        } catch (Exception e) {
            redirectAttributes.addFlashAttribute("error", "Error al
cargar documentos: " + e.getMessage());
            return "material/crear";
        }
    }

    @GetMapping("/listar/{id}")
    public String listarMateriales(@PathVariable int id,
```

```
                                    ModelMap model, HttpSession
session) {
        Persona persona = (Persona)
session.getAttribute("personaSession");
        if (persona instanceof Profesor) {
            boolean esProfe = (persona != null && persona.getId() ==
id);
            model.put("esProfe", esProfe);
        }

        List<Material> materiales =
materialService.listarMaterialesPorProfesor(id);
        if (materiales.isEmpty()) {
            model.put("info", "No hay materiales cargados");
        } else {
            model.put("materiales", materiales);
        }
        model.put("profesor", profesorService.getOne(id));

        return "material/lista";
    }

    @GetMapping("/ver/{id}")
    public ResponseEntity<byte[]> materialProfesor(@PathVariable int
id) {
        Material material = materialService.getOne(id);
        byte[] archivo = material.getArchivo();

        if (archivo != null) {
            HttpHeaders headers = new HttpHeaders();
            headers.setContentType(MediaType.APPLICATION_PDF);

headers.setContentDisposition(ContentDisposition.inline().filename(mate
rial.getDescripcion() + ".pdf").build());
            return new ResponseEntity<>(archivo, headers,
HttpStatus.OK);
        } else {
            return ResponseEntity.notFound().build();
        }
    }
```

HTML

material/crear.html

```
<!DOCTYPE html>
<html lang="es" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Registrar Material</title>
    <link rel="icon" type="image/png" th:href="@{/img/favicon.png}" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstra
p-icons.css" rel="stylesheet">
    <link rel="stylesheet" th:href="@{/css/general.css}" />
    <link rel="stylesheet" th:href="@{/css/alumno.css}" />

    <script th:inline="javascript">
        let materiales = [];

        function agregarMaterial() {
            const archivo =
document.getElementById("archivo").files[0];
            const descripcion =
document.getElementById("descripcion").value;

            if (!archivo || !descripcion) {
                alert("Completá todos los campos antes de agregar
otro.");
                return;
            }

            const index = materiales.length;
            materiales.push({ archivo, descripcion });


            const lista = document.getElementById("listaMateriales");
            const item = document.createElement("li");
            item.textContent = `Material n° ${index + 1}:
${descripcion} `;

            const btnEliminar = document.createElement("button");
            btnEliminar.className = "btn-eliminar";
            btnEliminar.innerHTML =`<i class="bi bi-trash"></i>`;


            btnEliminar.onclick = () => {
```

```
                materiales.splice(index, 1);
                item.remove();
                actualizarLista();
            };

            item.appendChild(btnEliminar);
            lista.appendChild(item);

            document.getElementById("archivo").value = "";
            document.getElementById("descripcion").value = "";
        }

        function actualizarLista() {
            const lista = document.getElementById("listamateriales");
            lista.innerHTML = "";
            materiales.forEach((doc, i) => {
                const item = document.createElement("li");
                item.textContent = `Documento ${i + 1}:
${doc.descripcion} `;

                const btnEliminar = document.createElement("button");
                btnEliminar.className = "btn-eliminar";
                btnEliminar.innerHTML = `<i class="bi bi-trash"></i>`;

                btnEliminar.onclick = () => {
                    materiales.splice(i, 1);
                    actualizarLista();
                };

                item.appendChild(btnEliminar);
                lista.appendChild(item);
            });
        }

        function enviarMateriales() {
            const archivo =
document.getElementById("archivo").files[0];
            const descripcion =
document.getElementById("descripcion").value;
            const idDictadoClase =
document.getElementById("dictado").value;

            if (archivo && descripcion) {
```

```
                materiales.push({ archivo, descripcion });
            }

            if (materiales.length === 0) {
                alert("No hay materiales para enviar.");
                return;
            }

            if (!idDictadoClase) {
                alert("Seleccioná una materia antes de enviar.");
                return;
            }

            const formData = new FormData();

            materiales.forEach((mat) => {
                formData.append("archivos", mat.archivo);
                formData.append("descripciones", mat.descripcion);
                formData.append("idDictadoClase", idDictadoClase);
            });

            fetch("/material/registro", {
                method: "POST",
                body: formData
            })
            .then(response => {
                if (response.ok) {
                    alert("Materiales cargados correctamente.");
                    window.location.href =
/*[[@{/material/listar/{id}(id=${profesor.id})}]]*/ '';
                } else {
                    alert("Error al cargar materiales.");
                }
            })
        }
    </script>
</head>

<body>
    <div th:replace="~{fragments/headerGral :: fragment}"></div>
    <div class="form-container">
        <form id="formMateriales" enctype="multipart/form-data">
            <div class="titulo-con-flecha">
```

```
                <a th:href="@{ '/material/listar/' + ${profesor.id} }"
                class="btn-volver" title="Volver">
                    <i class="bi bi-arrow-left"></i>
                </a>
                <h2>Cargar Nuevo Material</h2>
            </div>
            <div th:replace="~{fragments/mensajes :: fragment}"></div>
            <div>
                <label for="archivo">Archivo PDF</label>
                <input type="file" id="archivo" accept=".pdf" />
            </div>

            <div>
                <label for="descripcion">Descripción</label>
                <input type="text" id="descripcion"
placeholder="Descripción"/>
            </div>
            <div>
                <label for="dictado">Seleccioná la materia:</label>
                <select id="dictado" name="idDictadoClase"
class="form-select" required>
                    <option value="" disabled selected>Elegí una
materia</option>
                    <option th:each="d : ${dictados}"
th:value="${d.id}" th:text="${d.materia.nombre + ' - ' +
d.nivel.descripcion}"></option>
                </select>
            </div>
            <div class="buttons">
                <button type="button" onclick="agregarMaterial()"
class="btn">Ingresar otro</button>
                <button type="button" onclick="enviarMateriales()"
class="btn">Aceptar</button>
            </div>
            <div>
                <h5>Materiales agregados:</h4>
                <ul id="listaMateriales" class="lista-materiales"></ul>
            </div>

        </form>
    </div>
    <div th:replace="~{fragments/footer :: fragment}"></div>
```

```
</body>
</html>
```

material/lista.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Lista Materiales</title>
    <link rel="icon" type="image/png" th:href="@{/img/favicon.png}" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.m
in.css" rel="stylesheet" integrity="sha384-..."
crossorigin="anonymous">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstra
p-icons.css" rel="stylesheet">
    <link rel="stylesheet" th:href="@{/css/general.css}" />
    <link rel="stylesheet" th:href="@{/css/alumno.css}" />
</head>
<body>
    <div th:replace="~{fragments/headerGral :: fragment}"></div>
    <div class="titulo-con-flecha">
        <a th:href="${esProfe} ? @{/inicio} : @{ '/profesor/ver/' +
${profesor.id} }"
        class="btn-volver" title="Volver">
            <i class="bi bi-arrow-left"></i>
        </a>
        <h2>Materiales del profesor</h2>
    </div>

    <div class="container mt-4">
        <div th:if="${esProfe}" class="text-end mb-3">
            <a th:href="@{'/material/registrar'}" class="btn">
                <i class="bi bi-plus-circle"></i> Nuevo material
            </a>
        </div>
        <div th:replace="~{fragments/mensajes :: fragment}"></div>
```

```
        <div th:if="${info}" class="text-center mb-4">
            <p th:text="${info}" class="fs-5 text-muted"></p>
        </div>


        <div class="row row-cols-1 row-cols-md-3 g-4">
            <div th:each="mat : ${materiales}" class="col">
                <a th:href="@{'/material/ver/' + ${mat.id}}"
target="_blank" style="text-decoration: none;">
                    <div class="card h-100 text-center shadow-sm
justify-content-center" style="cursor: pointer;">
                        <div class="card-body d-flex flex-column
justify-content-center">
                            <h5 th:text="${mat.descripcion}"></h5>
                            <p
th:text="${mat.dictadoClase.materia.nombre + ' - ' +
mat.dictadoClase.nivel.descripcion}"></p>
                        </div>

                        <!-- Botones solo si es profesor -->
                        <div th:if="${esProfe}" class="card-footer ">
                            <div class="d-flex justify-content-center
gap-2">
                                <a th:href="@{'/material/modificar/' +
${mat.id}}" class="btn">
                                    Modificar
                                </a>
                                <form
th:action="@{'/material/eliminar/' + ${mat.id}}" method="post">
                                    <button type="submit"
class="btn-eliminar" onclick="return confirm('¿Estás seguro de que
querés eliminar este material?')">
                                     <i class="bi bi-trash"></i>
                                    </button>
                                </form>
                            </div>
                        </div>
                    </div>
                </a>
            </div>
        </div>
    </div>
    <div th:replace="~{fragments/footer :: fragment}"></div>
```

```
</body>
</html>
```