package com.example.taskmanagement.repository;

import com.example.taskmanagement.entity.Task;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import java.util.Date;

import java.util.List;

@Repository

public interface TaskRepository extends JpaRepository<Task, Integer> {

List<Task> findByPriorityAndStatus(String priority, String status);

List<Task> findByDueDateBefore(Date currentDate);

List<Task> findByDueDateBetween(Date startDate, Date endDate);

List<Task> findByUserIdAndStatus(int userId, String status);

List<Task> findByProjectId(int projectId);

}

package com.example.taskmanagement.service;

import com.example.taskmanagement.entity.Task;

import com.example.taskmanagement.exception.TaskNotFoundException;

import com.example.taskmanagement.repository.TaskRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.Date;

import java.util.List;

@Service

public class TaskService {

@Autowired

private TaskRepository taskRepository;

public Task addTask(Task task) {

return taskRepository.save(task);

}

public List<Task> getAllTasks() {

return taskRepository.findAll();

}

public List<Task> getTasksByPriorityAndStatus(String priority, String status) {

return taskRepository.findByPriorityAndStatus(priority, status);

}

public List<Task> getOverdueTasks(Date currentDate) {

return taskRepository.findByDueDateBefore(currentDate);

}

public List<Task> getTasksDueSoon(Date startDate, Date endDate) {

return taskRepository.findByDueDateBetween(startDate, endDate);

}

public List<Task> getTasksByUserAndStatus(int userId, String status) {

return taskRepository.findByUserIdAndStatus(userId, status);

}

public List<Task> getTasksByProject(int projectId) {

return taskRepository.findByProjectId(projectId);

}

public Task getTaskById(int taskId) {

return taskRepository.findById(taskId)

.orElseThrow(() -> new TaskNotFoundException("Task with ID " + taskId + " not found"));

}

public Task updateTask(Task task) {

if (!taskRepository.existsById(task.getTaskId())) {

throw new TaskNotFoundException("Task with ID " + task.getTaskId() + " not found");

}

return taskRepository.save(task);

}

public void deleteTask(int taskId) {

if (!taskRepository.existsById(taskId)) {

throw new TaskNotFoundException("Task with ID " + taskId + " not found");

}

taskRepository.deleteById(taskId);

}

}

package com.example.taskmanagement.controller;

import com.example.taskmanagement.entity.Task;

import com.example.taskmanagement.service.TaskService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.Date;

import java.util.List;

@RestController

@RequestMapping("/api/tasks")

public class TaskController {

@Autowired

private TaskService taskService;

@PostMapping("/post")

public ResponseEntity<Task> addTask(@RequestBody Task task) {

Task createdTask = taskService.addTask(task);

return ResponseEntity.ok(createdTask);

}

@GetMapping("/overdue")

public ResponseEntity<List<Task>> getOverdueTasks() {

List<Task> overdueTasks = taskService.getOverdueTasks(new Date());

return ResponseEntity.ok(overdueTasks);

}

@GetMapping("/priority/{priority}/status/{status}")

public ResponseEntity<List<Task>> getTasksByPriorityAndStatus(@PathVariable String priority, @PathVariable String status) {

List<Task> tasks = taskService.getTasksByPriorityAndStatus(priority, status);

return ResponseEntity.ok(tasks);

}

@GetMapping("/due-soon")

public ResponseEntity<List<Task>> getTasksDueSoon(@RequestParam Date startDate, @RequestParam Date endDate) {

List<Task> tasks = taskService.getTasksDueSoon(startDate, endDate);

return ResponseEntity.ok(tasks);

}

@GetMapping("/user/{userId}/status/{status}")

public ResponseEntity<List<Task>> getTasksByUserAndStatus(@PathVariable int userId, @PathVariable String status) {

List<Task> tasks = taskService.getTasksByUserAndStatus(userId, status);

return ResponseEntity.ok(tasks);

}

@GetMapping("/category/{projectId}")

public ResponseEntity<List<Task>> getTasksByProject(@PathVariable int projectId) {

List<Task> tasks = taskService.getTasksByProject(projectId);

return ResponseEntity.ok(tasks);

}

@PutMapping("/update/{taskId}")

public ResponseEntity<Task> updateTask(@PathVariable int taskId, @RequestBody Task taskDetails) {

Task updatedTask = taskService.getTaskById(taskId); // Throws exception if not found

updatedTask.setTaskName(taskDetails.getTaskName());

updatedTask.setDescription(taskDetails.getDescription());

updatedTask.setDueDate(taskDetails.getDueDate());

updatedTask.setPriority(taskDetails.getPriority());

updatedTask.setStatus(taskDetails.getStatus());

updatedTask.setProjectId(taskDetails.getProjectId());

updatedTask.setUserId(taskDetails.getUserId());

return ResponseEntity.ok(taskService.updateTask(updatedTask));

}

@DeleteMapping("/{taskId}")

public ResponseEntity<Void> deleteTask(@PathVariable int taskId) {

taskService.deleteTask(taskId);

return ResponseEntity.noContent().build();

}

}

package com.example.taskmanagement.exception;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.RestControllerAdvice;

@RestControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(TaskNotFoundException.class)

public ResponseEntity<?> handleTaskNotFoundException(TaskNotFoundException ex) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(new ErrorResponse("TASK\_NOT\_FOUND", ex.getMessage()));

}

@ExceptionHandler(Exception.class)

public ResponseEntity<?> handleGenericException(Exception ex) {

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(new ErrorResponse("INTERNAL\_SERVER\_ERROR", ex.getMessage()));

}

// Helper class for consistent error responses

static class ErrorResponse {

private String code;

private String message;

public ErrorResponse(String code, String message) {

this.code = code;

this.message = message;

}

public String getCode() {

return code;

}

public String getMessage() {

return message;

}

}

}

package com.example.taskmanagement.exception;

public class TaskNotFoundException extends RuntimeException {

public TaskNotFoundException(String message) {

super(message);

}

}