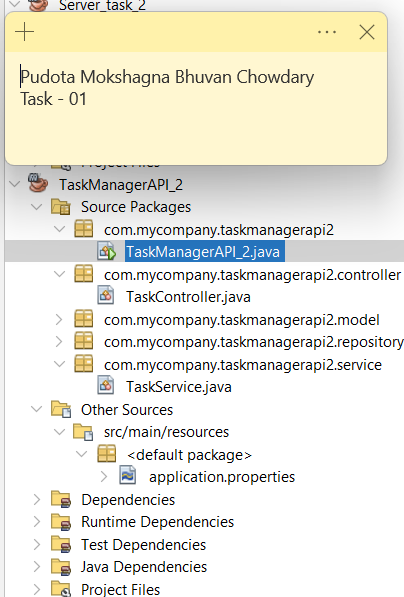
**Task – 01 Java backend and REST API example.**

1. **Objective:** The objective of this project is to build a robust Java Spring Boot REST API for managing and executing "tasks"—each representing a shell command that can run in a Kubernetes pod. Every task tracks its unique identifier, name, owner, command string, and a history of past executions (including timestamps and command output). All data is stored in MongoDB to enable persistence and scalability.

This API is designed to support:

* **Task creation, searching, updating, and deletion** via different endpoints
* **Execution and logging of shell commands** remotely over HTTP
* **Easy auditing of task history and results**, with each execution tracked by start time, end time, and output
* **Clear, modular code structure** for maintainability and extensibility

2. **Application Layout:**



The project's structure follows best practices for a Spring Boot backend, ensuring code modularity, maintainability, and clarity. The major components and their responsibilities in the layout are:

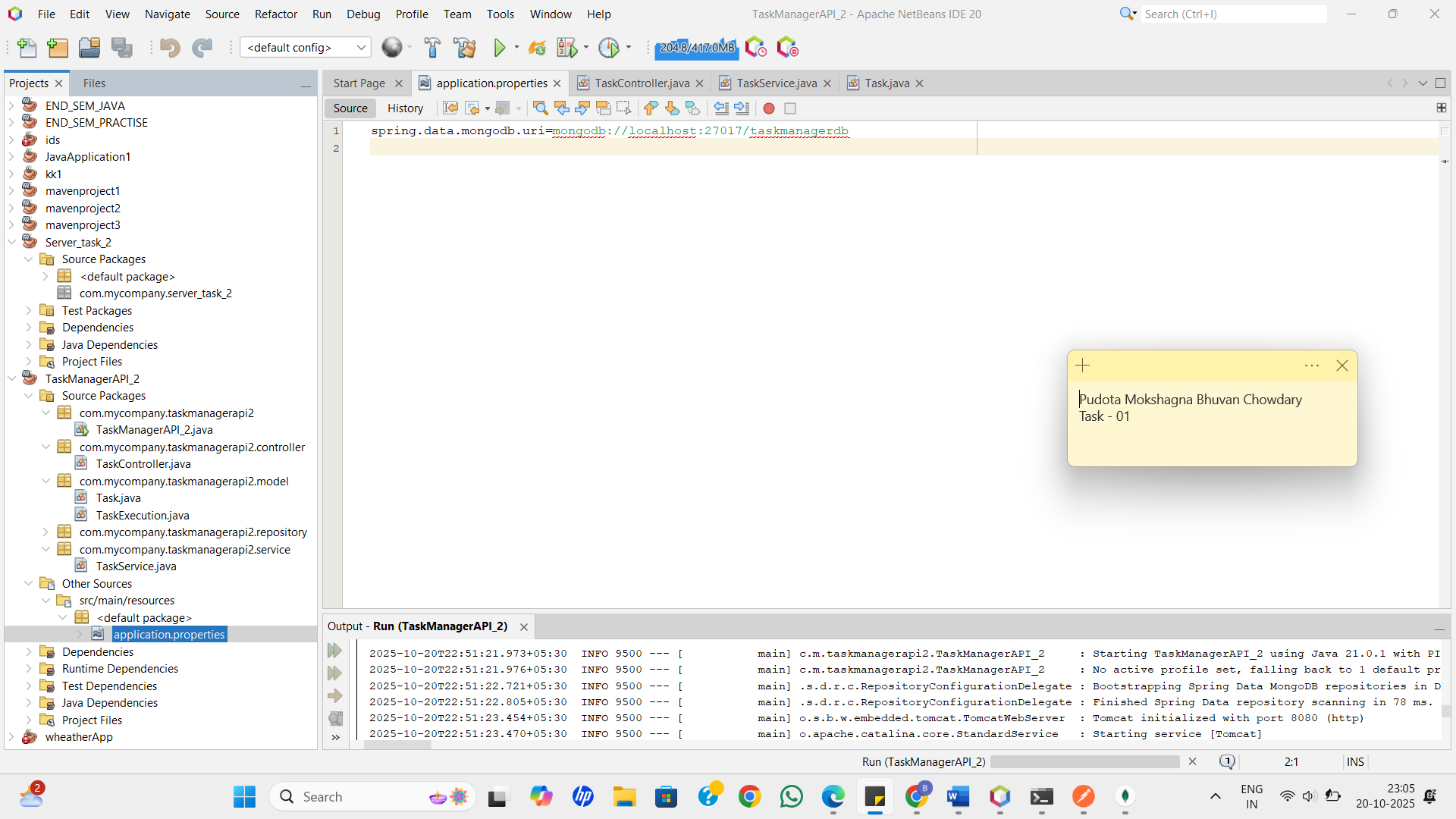
* **Main Application Class (TaskManagerAPI\_2.java):**  
  Entry point for the Spring Boot service, responsible for initializing and running the application.
* **Controller Layer (TaskController.java):**  
  Exposes REST endpoints for clients, handles incoming HTTP requests for creating, updating, searching, deleting, and executing tasks.
* **Model Layer (com.mycompany.taskmanagerapi2.model):**  
  Contains classes that define the data structure for a Task and its execution history, directly mapping to MongoDB documents.
* **Repository Layer (com.mycompany.taskmanagerapi2.repository):**  
  Implements Spring Data MongoDB repositories that facilitate CRUD operations and custom queries for Task objects.
* **Service Layer (TaskService.java):**  
  Encapsulates business logic, including validation, command execution, and interaction with the repository.
* **Configuration (src/main/resources/application.properties):**  
  Manages MongoDB connection settings and core Spring Boot configuration.

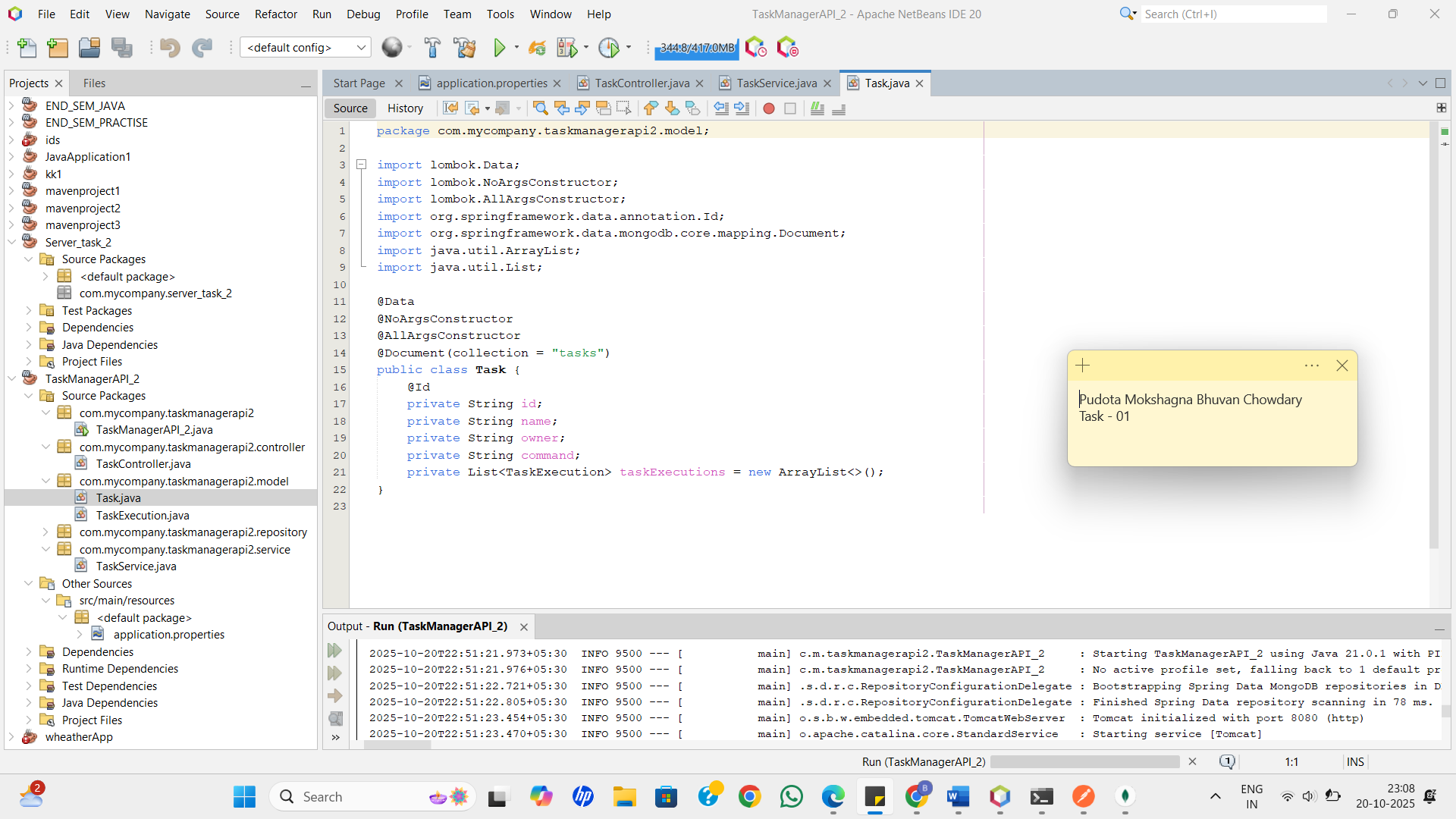
This layout enables separation of concerns, easy testing, and future extensibility of the application.

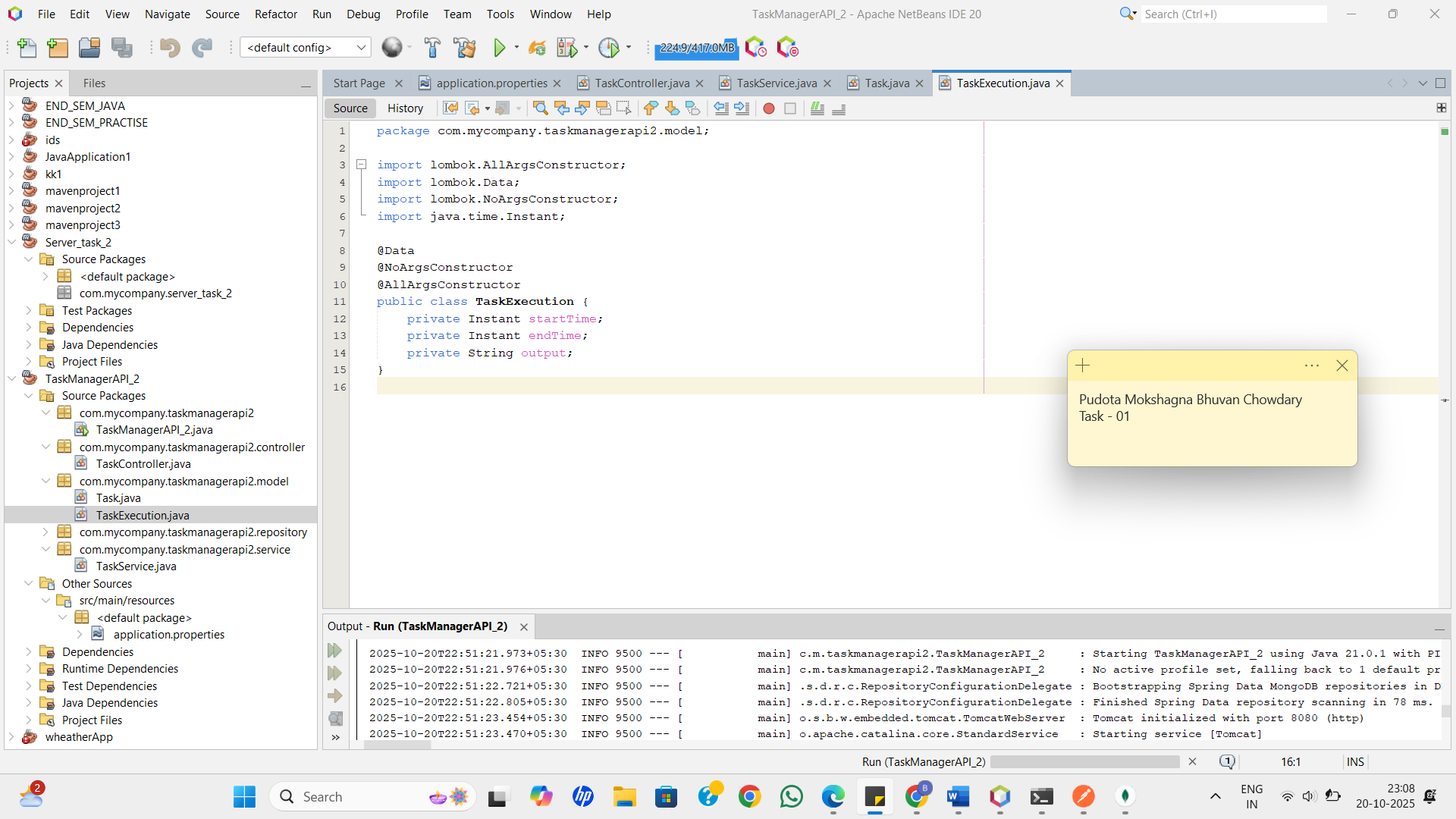
3. **Project Structure and Core Model Classes**

The following screenshots illustrate the complete application layout and the core model classes essential to the TaskManagerAPI\_2 project. The first screenshot displays the organized project directory with clearly separated packages for the main application, controller, model, repository, and service layers, exemplifying best practices in Spring Boot backend development.​

The subsequent images show the contents of the critical configuration file (application.properties) and the implementation of the primary Java data structures: Task and TaskExecution. These model classes define the key attributes for each task and its execution history, enabling robust data persistence and command tracking within MongoDB



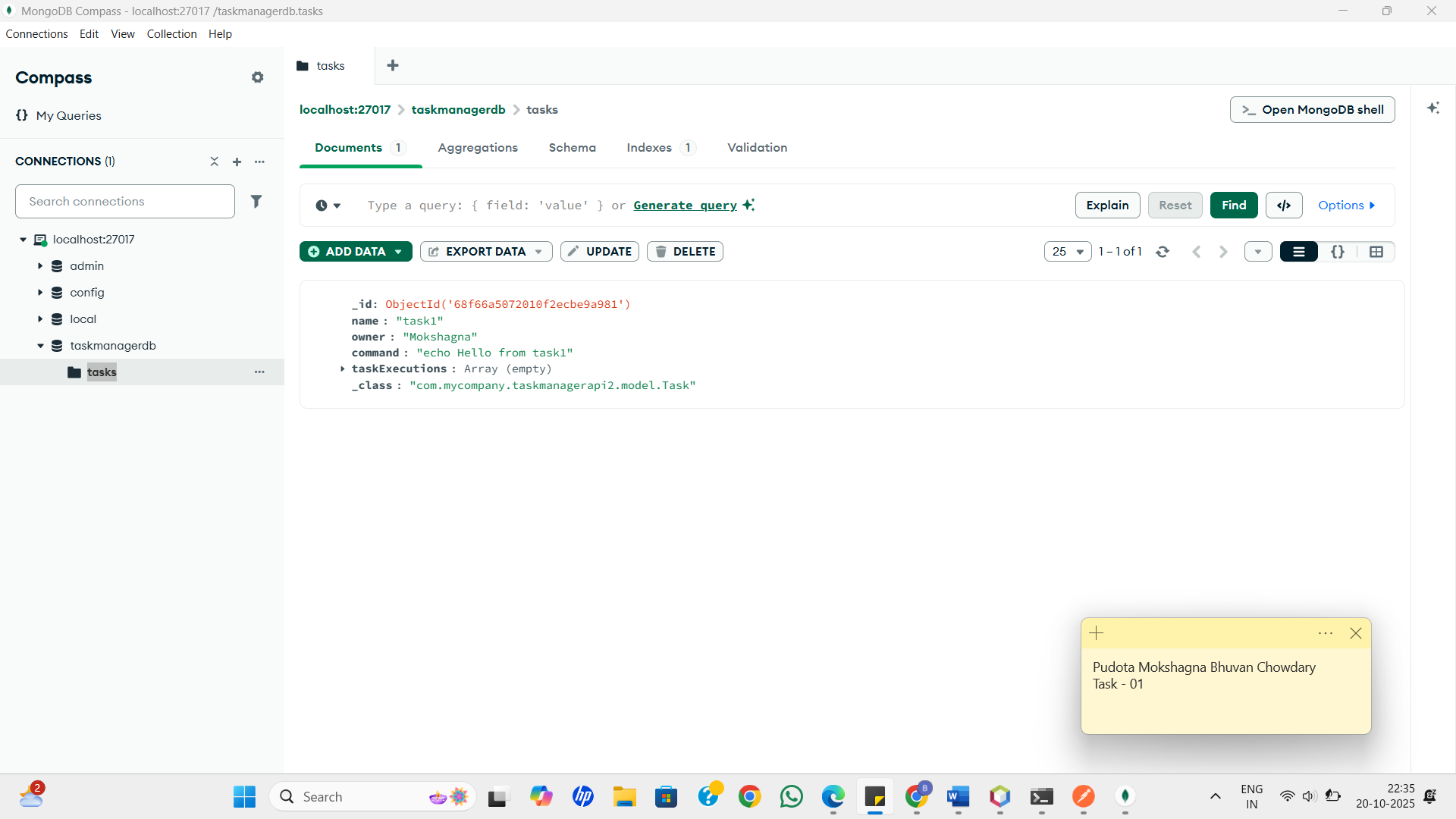




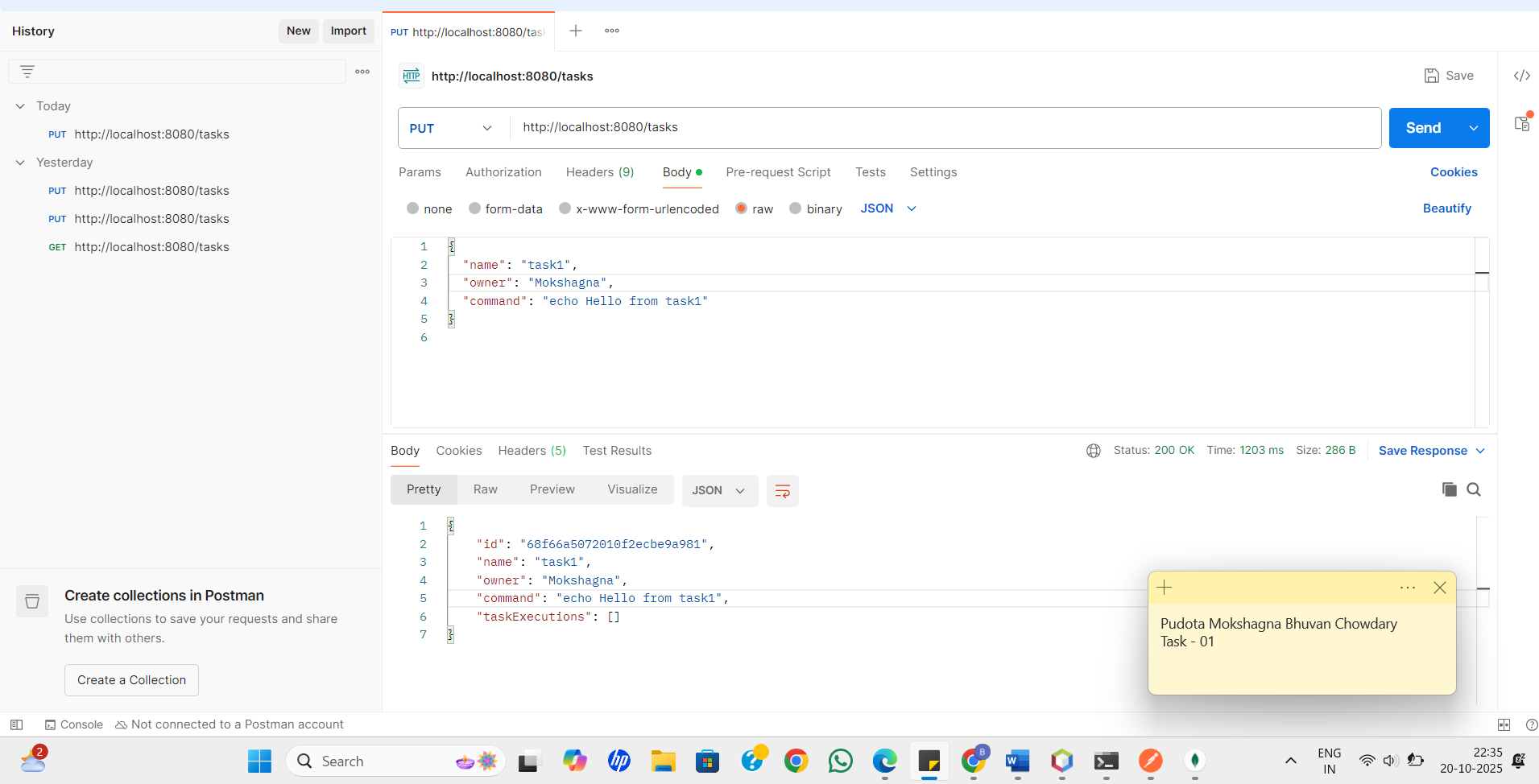
**4. Implementation and Verification of Task Creation and Execution**

The following screenshots demonstrate the full workflow of task management and execution in the application, with both API interactions and the resulting database state.

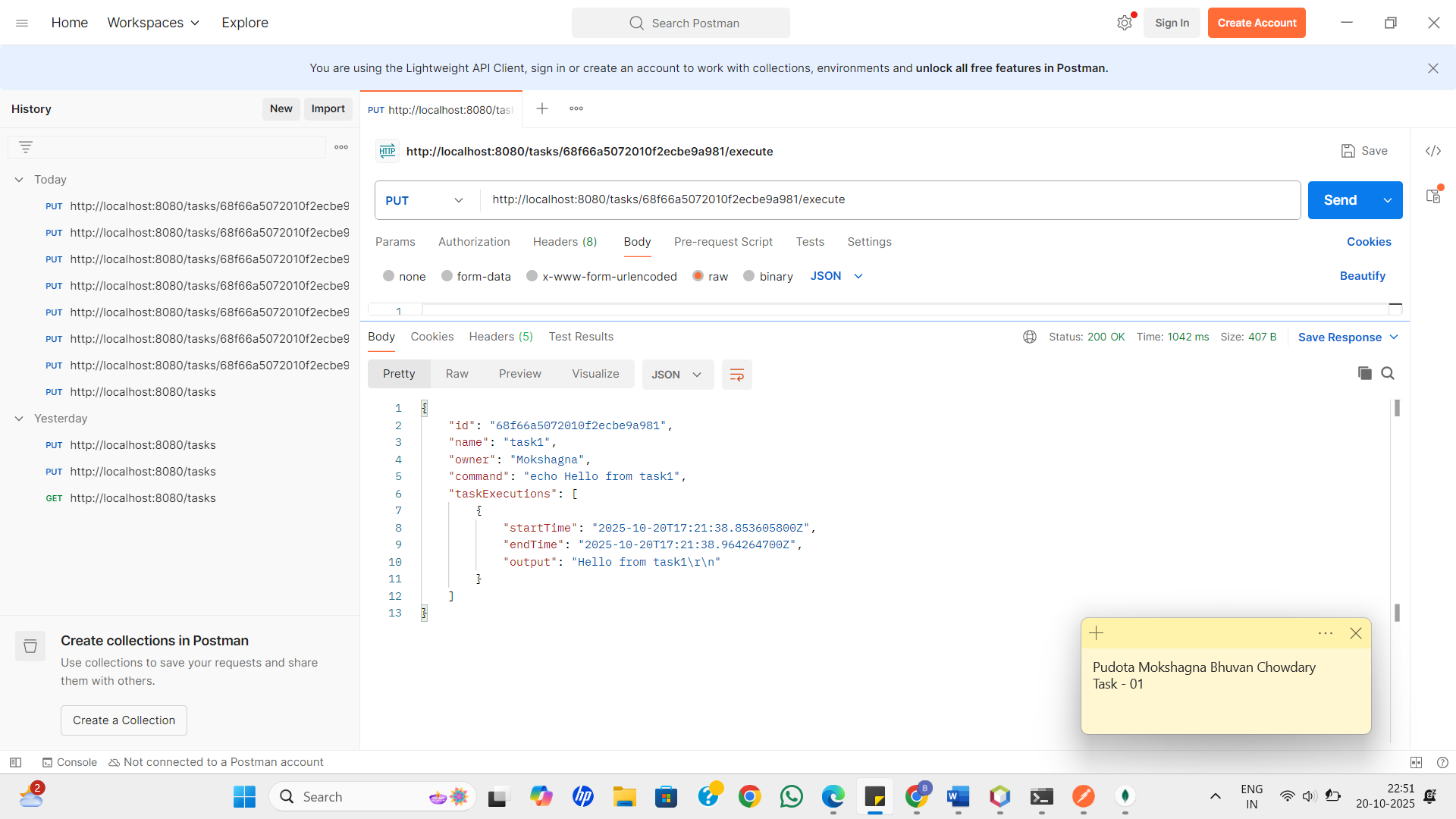
* The first MongoDB Compass image shows a Task document before execution, with the taskExecutions array empty, confirming that a new task was successfully created and persisted.​



* The initial Postman screenshot details a successful task creation request: the API returns the saved task object and verifies that the backend and database integration are working as expected.​



* The subsequent Postman screenshot captures the use of the /tasks/{id}/execute endpoint, showing that the command output and timestamps are correctly recorded in the taskExecutions field when the task is executed.​



* The final MongoDB Compass screenshot validates that the execution history is accurately stored and reflected in the underlying MongoDB document, closing the loop between the REST API, backend logic, and data persistence layers.

