MSSE672 – Component Based Software

Student: John Michael Kreski

Instructor: Mohammad Abu Matar

Assignment: Week 3 – Geometry App JDBC Integration

Date: 07/20/2025

File Name: HWExecution.doc

## Summary

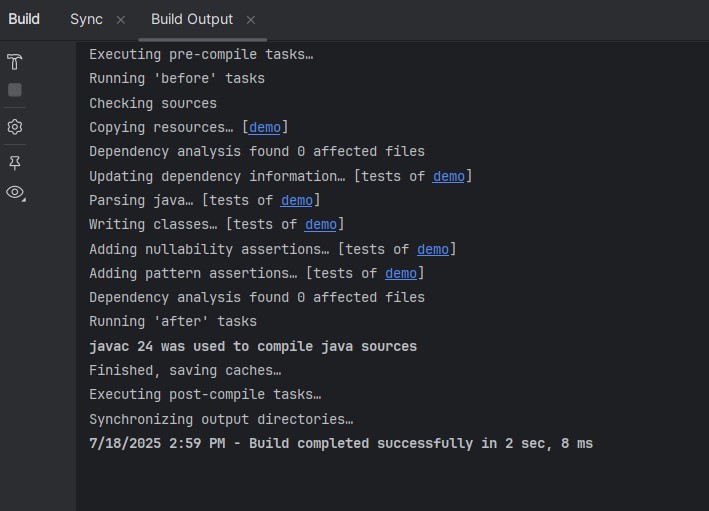
This assignment focused on integrating persistent data storage into the GeometryApp using Spring Boot with a MySQL database. The original in-memory implementation was replaced (or overridden) by a JDBC-based service implementation using JdbcTemplate.

The application now supports full CRUD functionality through a REST API, allowing users to add, view, update, and delete quadrilaterals from the database. REST endpoints were tested interactively through Swagger UI. The new service layer uses SQL queries to perform insert, retrieve, truncate, and aggregate operations against a quads table, with robust logging and validation.

The assignment also includes integration testing using an H2 in-memory test database to ensure correctness of all database interactions. Logging was added to each major service method for traceability. Code was modularized into Model, Service, and Controller layers in accordance with best practices.

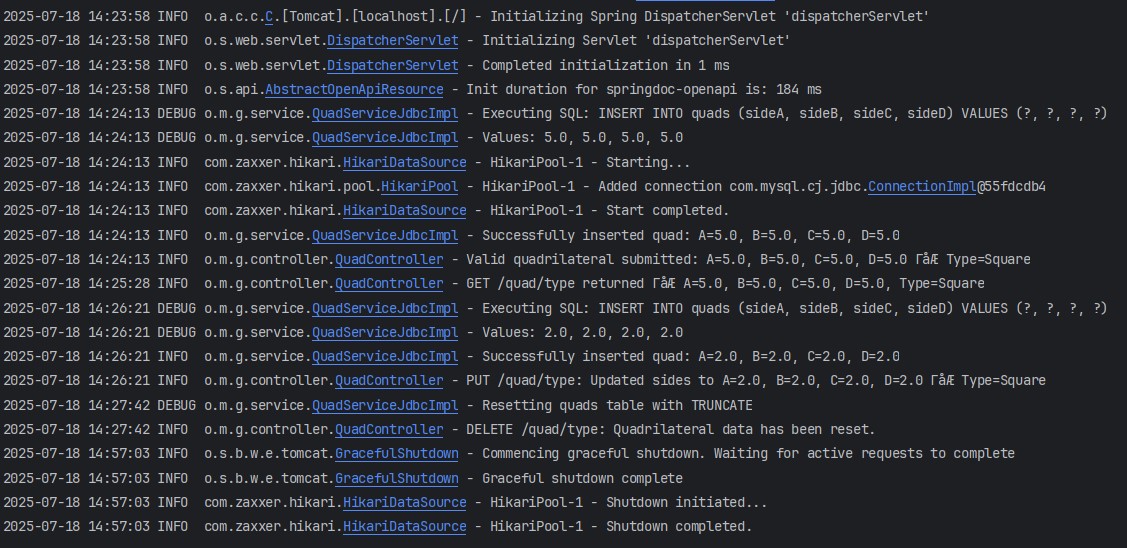
## Successful Compilation

*Screenshot of successful compilation from IntelliJ IDEA*

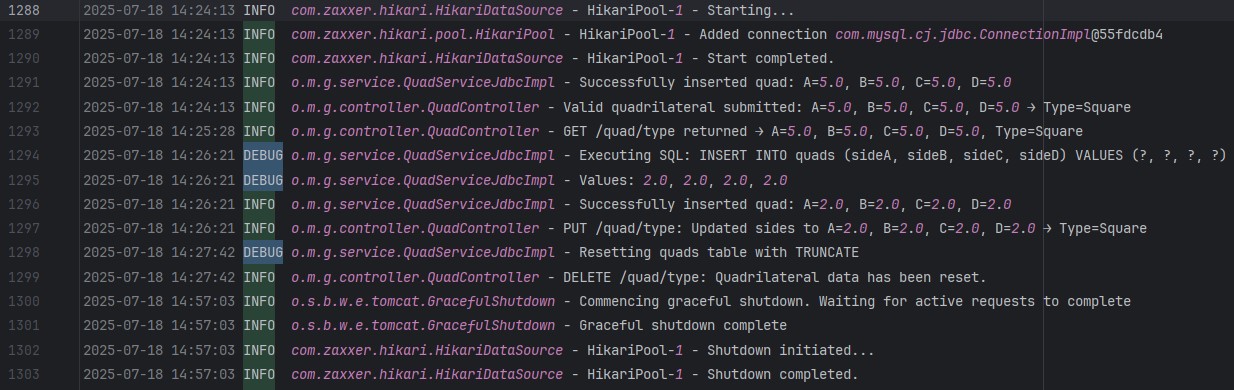
**

## Logs

## Screenshot 1 – Console Logs

*This screenshot displays runtime console output showing log messages at various levels (INFO, DEBUG, etc.).*

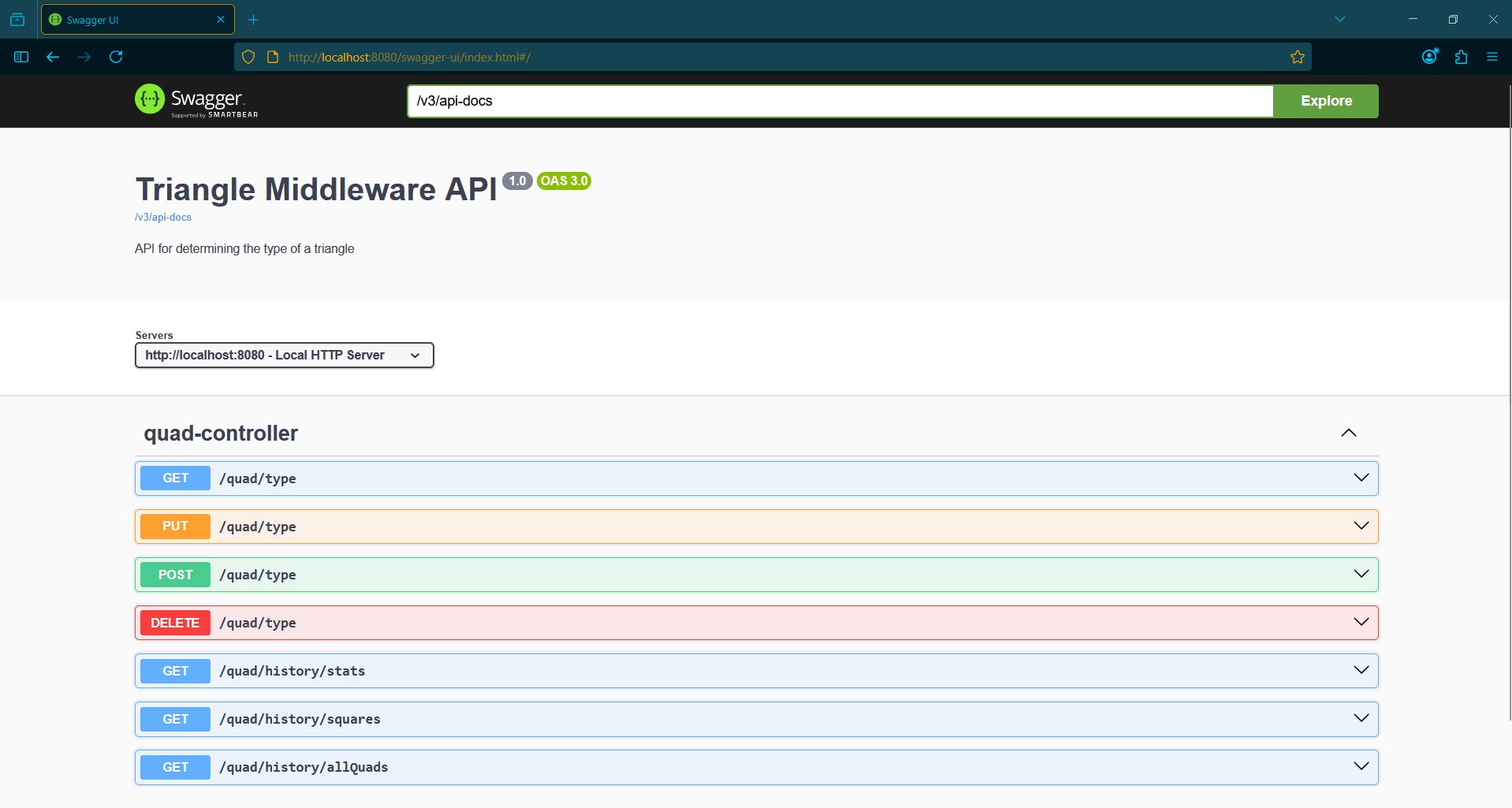
## Screenshot 2 – App Logs

*This screenshot shows the contents of the triangle-app.log file.*

## Application Execution

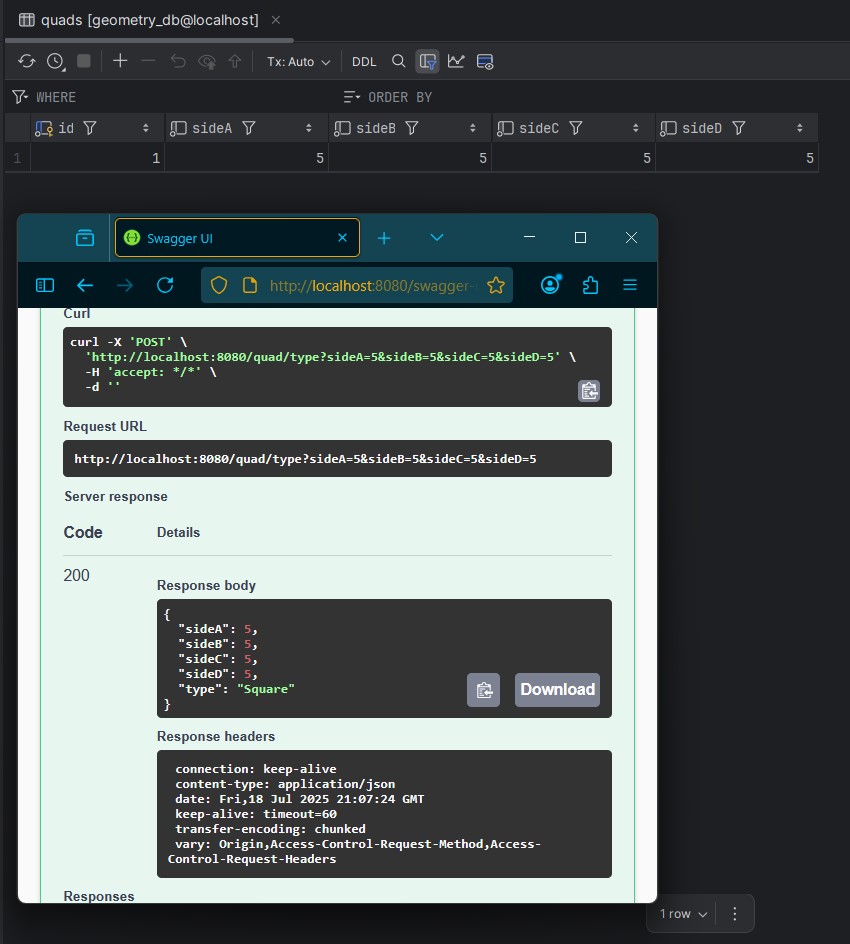
## Swagger UI

## Screenshot 1

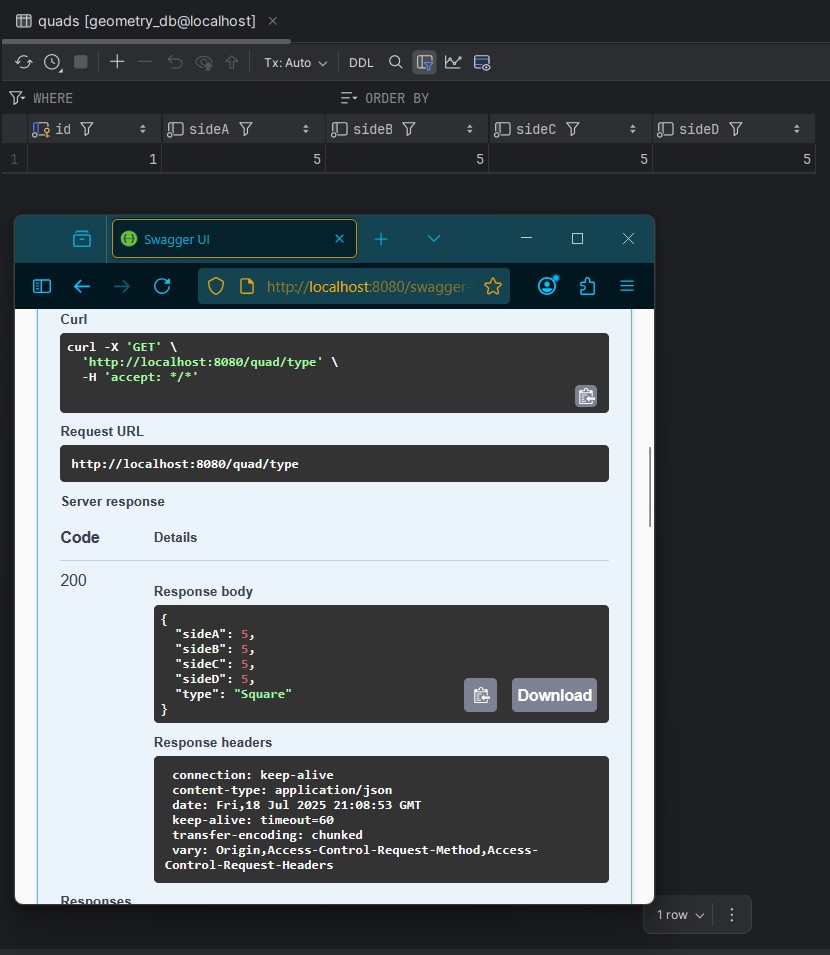
*Screenshot of Swagger UI running at: http://localhost:8080/swagger-ui.html*

**REST Endpoint Verification**

## Screenshot 1 - POST /quads

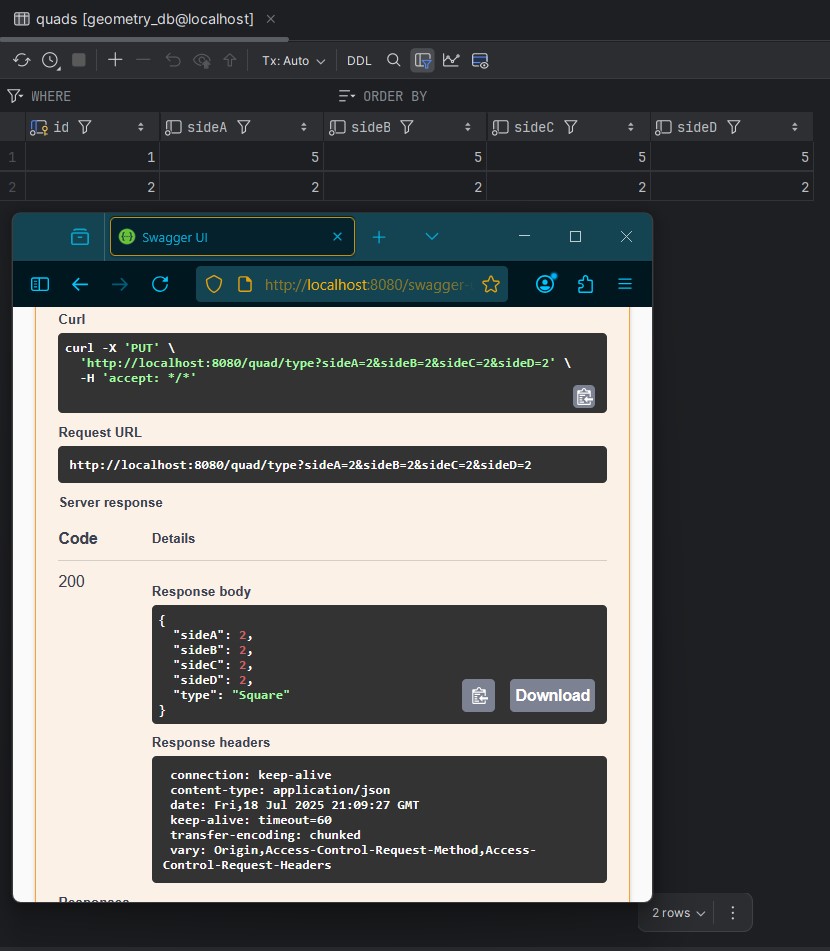
*Inserts a new quadrilateral into the MySQL database*

## Screenshot 2 - GET /quads

*Returns the most recently inserted quadrilateral*

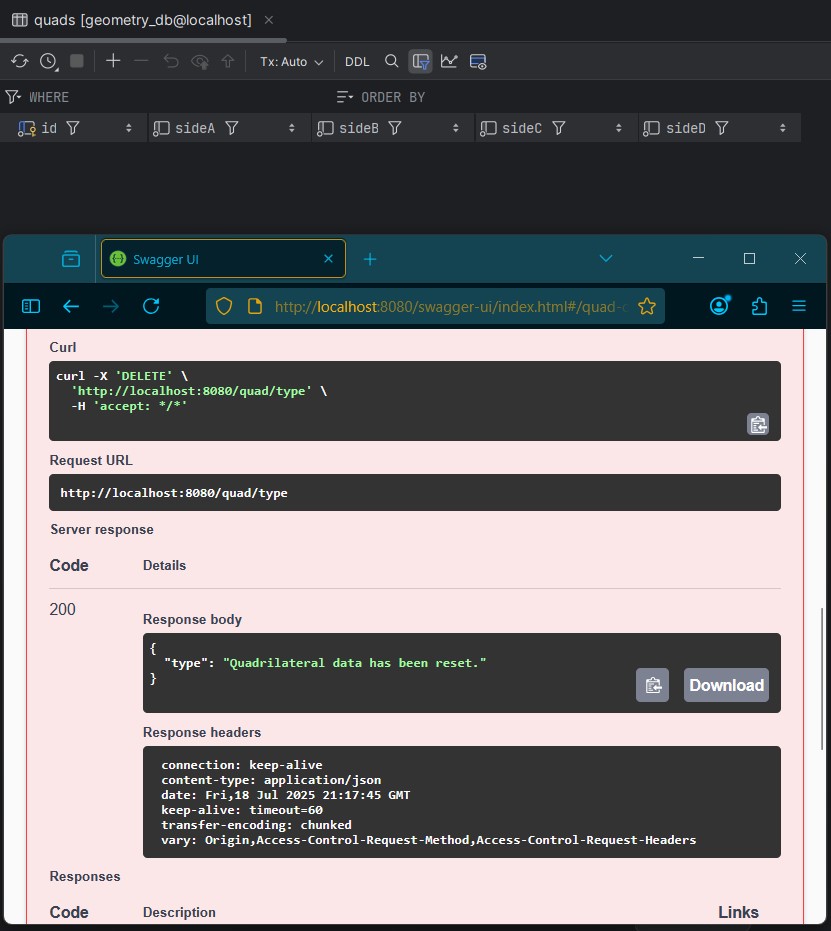
## Screenshot 3 - PUT /quads

*Currently inserts a new quadrilateral instead of updating an existing one*

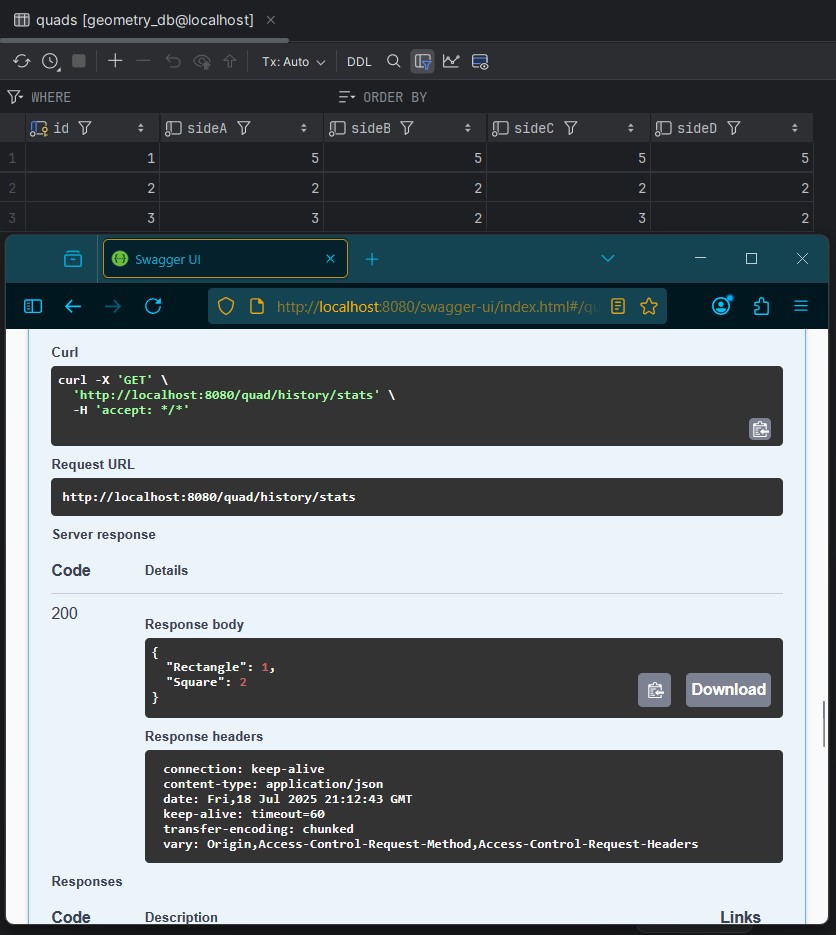


## Screenshot 4 - DELETE /quads

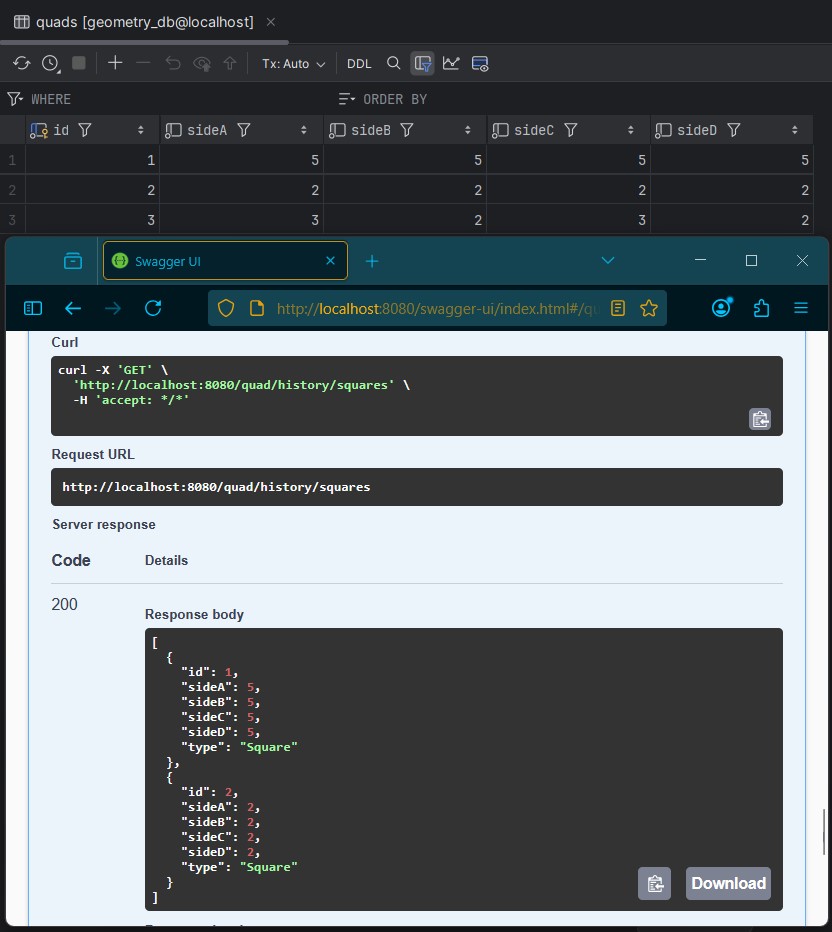
*Deletes all records from the table*



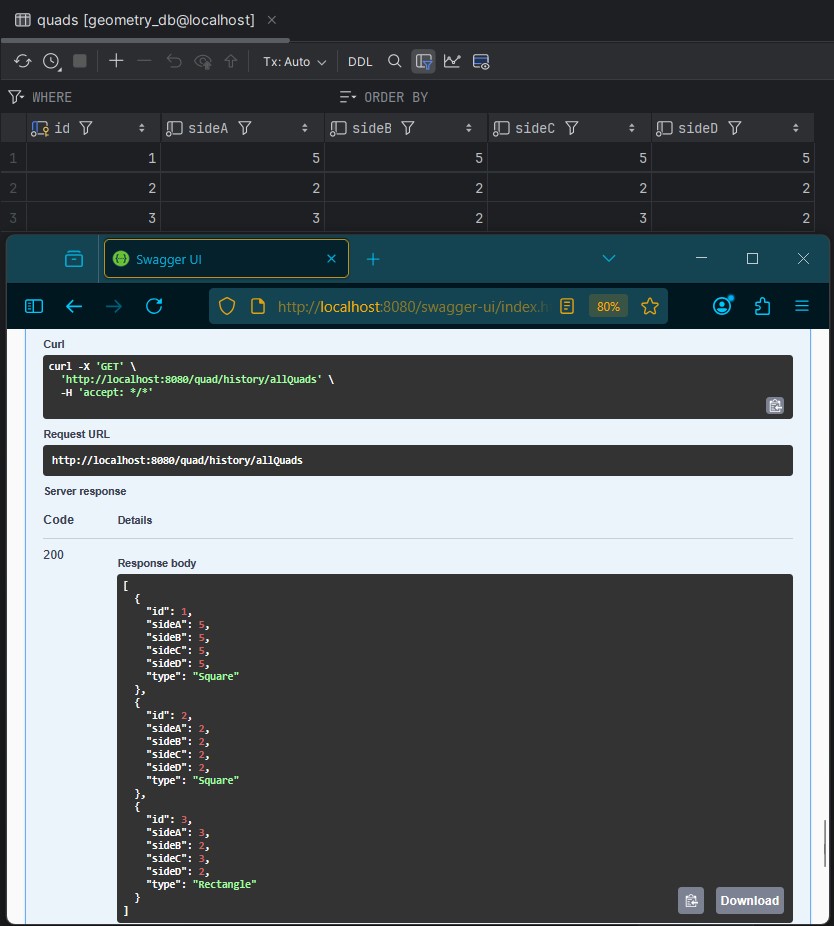
## Screenshot 5 - GET /quads/history/stats

*Gets all quad type stats submitted*

## Screenshot 6 - GET /quads/history/squares

*Gets all squares submitted*

## Screenshot 7 - GET /quads/history/allQuads

*Gets all records submitted*

## Refactored Files

* QuadService.java
  + New service interface that defines the contract for quadrilateral operations. Both QuadServiceJdbcImpl and the original QuadInMemoryImpl now implement this interface, allowing the application to switch between in-memory and database-backed logic.
* QuadServiceJdbcImpl.java
  + JDBC-based implementation of QuadService that uses JdbcTemplate to execute SQL queries. Declared as @Primary so it's used by default at runtime.
* QuadInMemoryImpl.java
  + The original in-memory implementation of QuadService, preserved for comparison and testability. Now conforms to the shared QuadService interface.
* Quadrilateral.java
  + Enhanced model class for storing and validating four-sided shapes. Includes validation, type-detection logic, and a formatted toString() method for logging and testing.
* QuadServiceIntegrationTest.java
  + New test class using @SpringBootTest and an H2 in-memory database to validate all database operations. Covers insert, reset, type filtering, and error handling.
* application.properties
  + Updated to use a live MySQL database connection for runtime (profile = dev).
* application-test.properties
  + Defines an isolated H2 environment with logging enabled for integration tests. Ensures test data does not affect production DB.
* schema.sql
  + SQL script to initialize the quads table when needed. Used during test runs or schema setup.

## Known Limitations & Future Enhancements

* Add quad types to DB.
* PUT currently acts as an insert (upsert) rather than an update. This will be addressed by adding @PathVariable Long id and checking existence before writing.
* GET /quads returns only the most recently inserted quad. I plan to enhance this by returning the full list or adding filtering and history endpoints.
* No error is thrown for duplicate or invalid inputs if validation is skipped. Additional input validation and 400-level responses are planned.
* Angle support will be added in future versions to support enhanced shape classification (e.g., distinguishing rhombus vs. square).
* All endpoints currently operate without user authentication or frontend filtering. This will be considered in a future sprint.

## Notes

* This project will be maintained in a public GitHub repository for version control and learning practice.
* A `.zip` file containing only `.java` files and the `/Docs` folder is submitted as per course instructions.
  + Docs/week3
* Screenshots are provided in this single document for clarity, as per the professor’s recent guidance.
* All screenshots referenced in this document are included in the project ZIP file under the directory: Docs/Week3/assets/.