MSSE672 – Component Based Software

Student: John Michael Kreski

Instructor: Mohammad Abu Matar

Assignment: Week 4 – Geometry App Hibernate Integration

Date: 07/27/2025

File Name: HWExecution.doc

## Summary

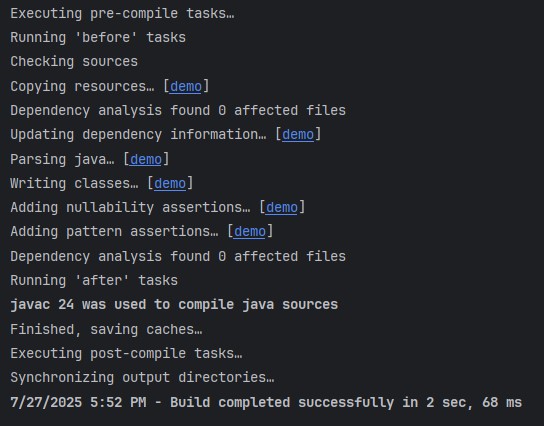
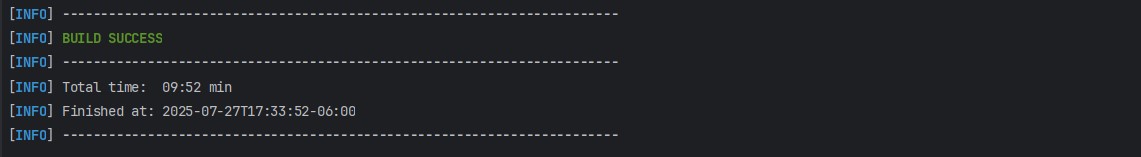
This assignment focused on enhancing the GeometryApp by integrating Hibernate as a full-featured Object-Relational Mapping (ORM) solution for persistent data storage. The JDBC-based service introduced last week was retained but supplemented with a new QuadServiceHibernateImpl implementation, allowing the application to toggle between JDBC and Hibernate-backed persistence using @Qualifier.

The application now supports full CRUD operations and analytical endpoints using Hibernate and Spring Data JPA, streamlining SQL operations into high-level repository methods. REST endpoints were tested interactively via Swagger UI. Significant portions of the boilerplate JDBC code have been replaced with concise, declarative logic via the QuadRepository, improving maintainability and readability.

All database operations, including insert, update, delete, and aggregation logic, are backed by robust validation and service-layer logging. The Hibernate service passed a comprehensive set of integration tests using a dedicated H2 in-memory database under the test profile.

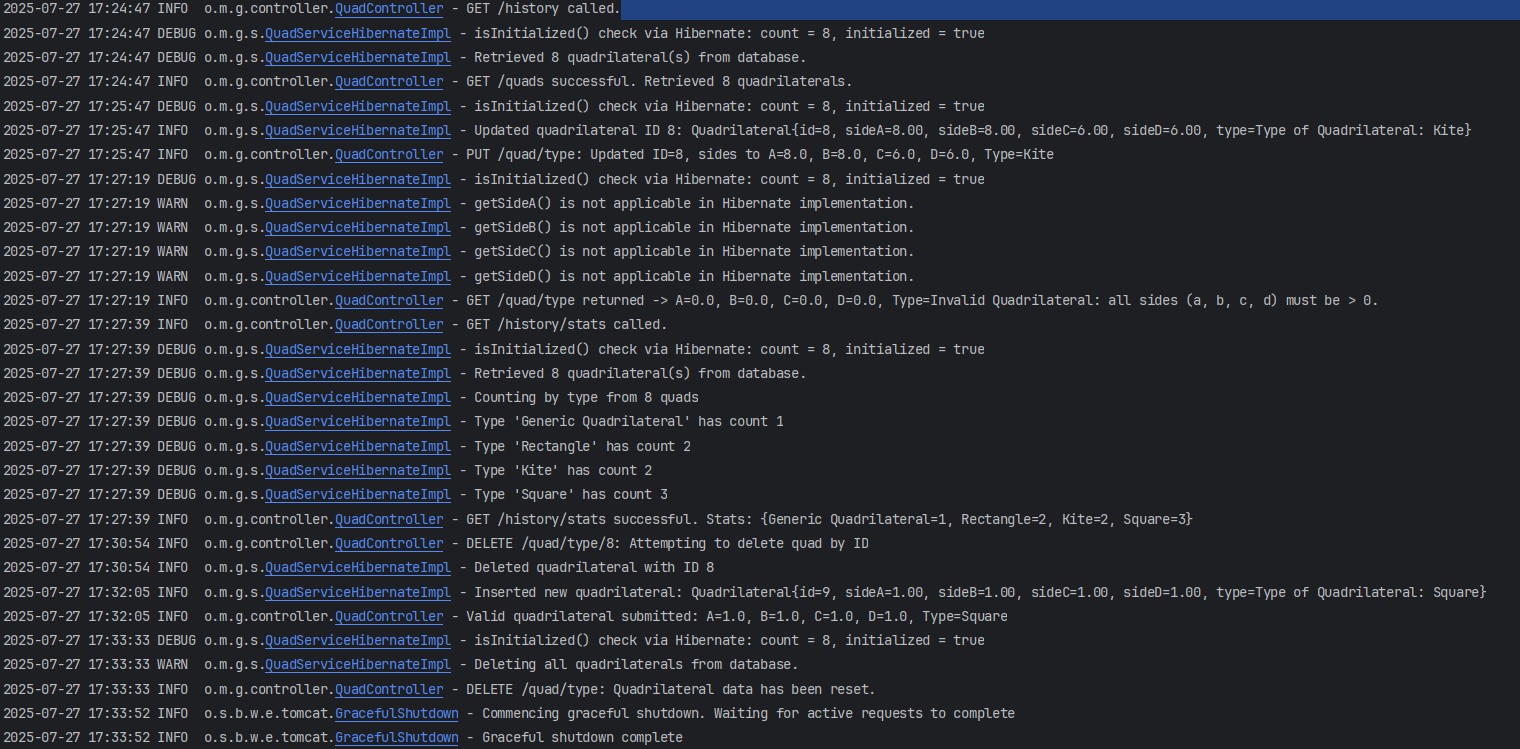
## Successful Compilation

*Screenshot of successful compilation from IntelliJ IDEA & Spring Terminal after completion.*

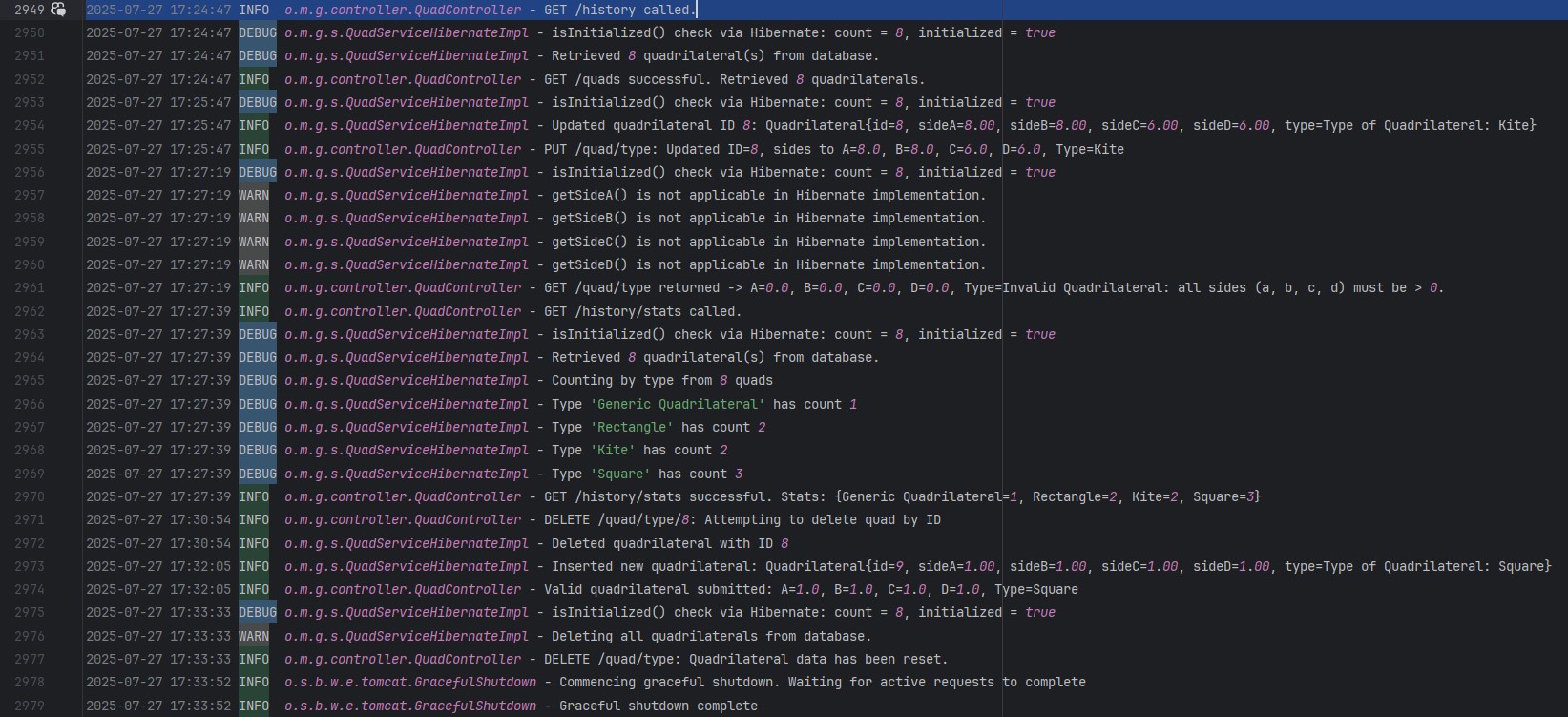
**

## Logs

## Screenshot 1 – Console Logs

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

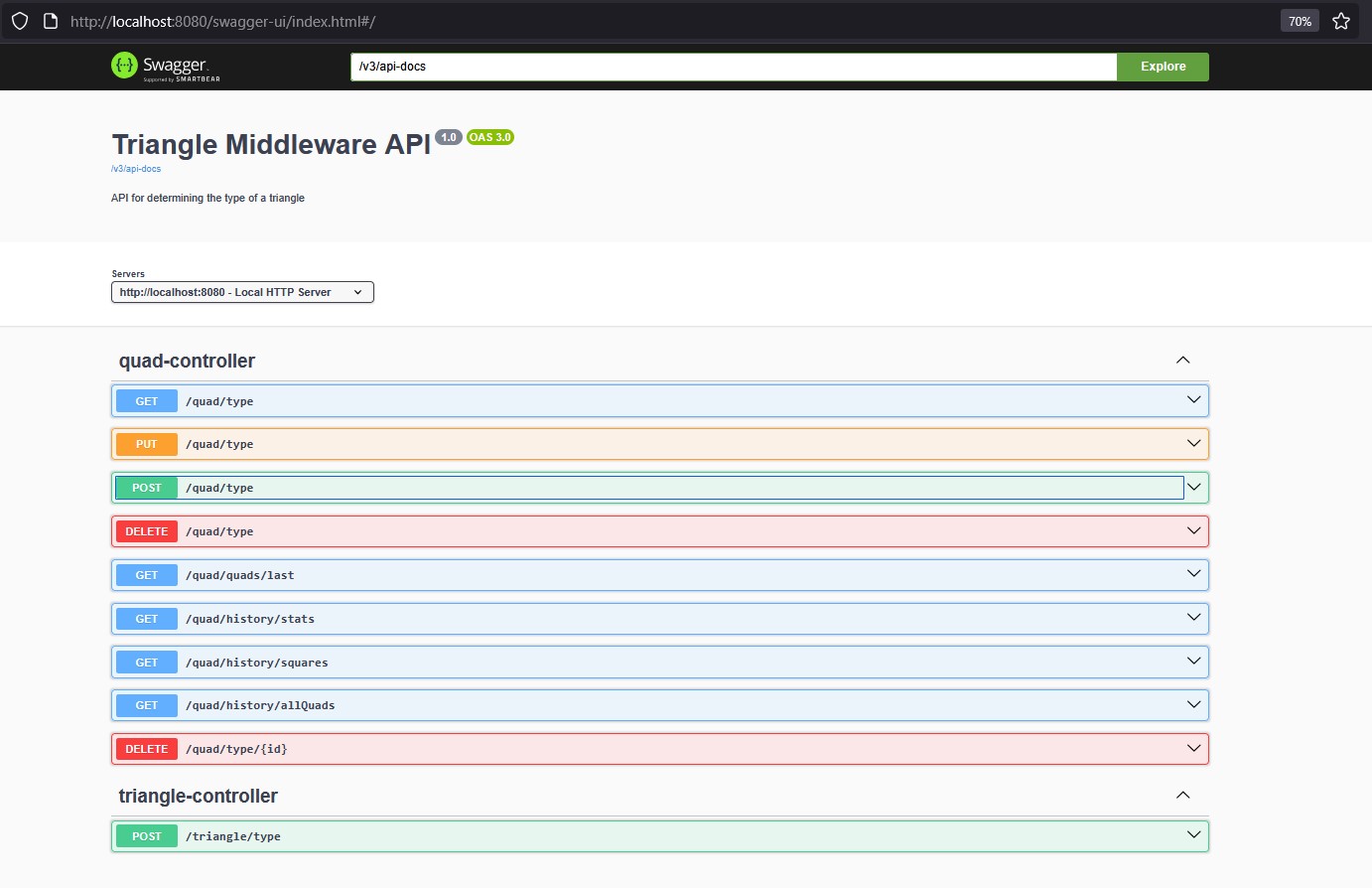
## Screenshot 2 – App Logs

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

## Application Execution

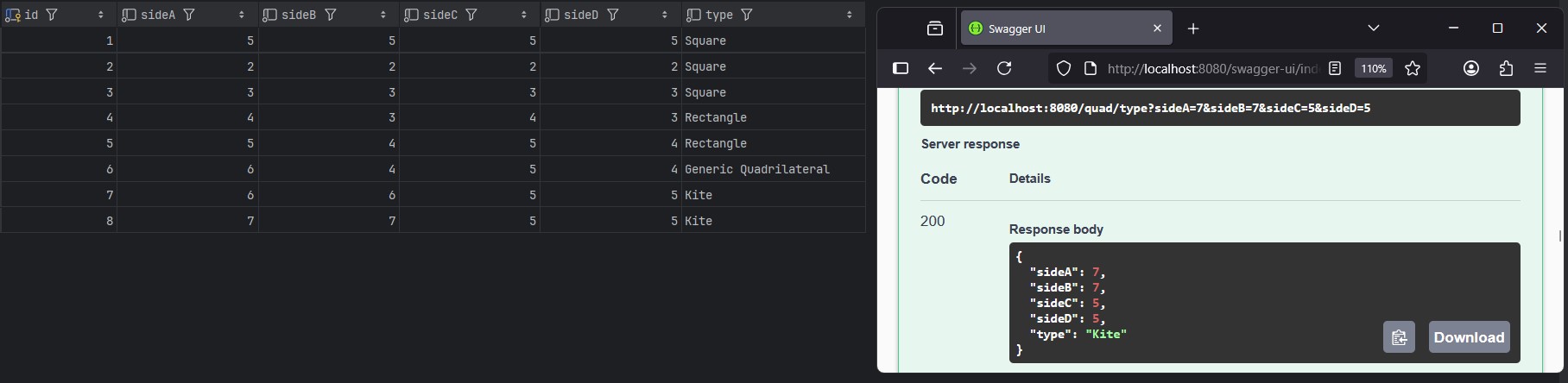
## Swagger UI

## Screenshot 1

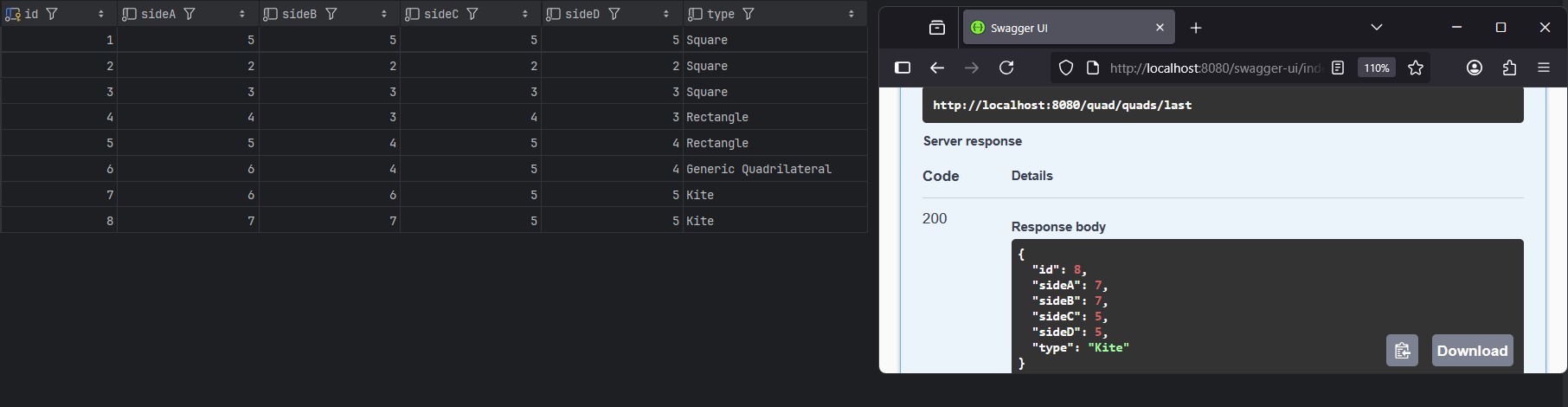
*Screenshot of Swagger UI running at:* [*http://localhost:8080/swagger-ui.html*](http://localhost:8080/swagger-ui.html)

**REST Endpoint Verification**

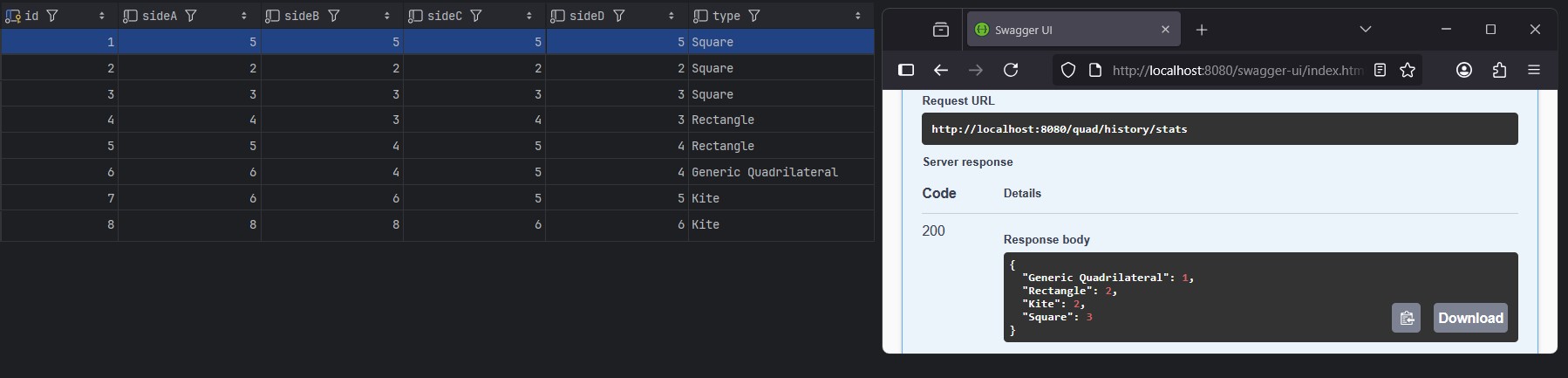
## Screenshot 1 - POST /quads

*Inserts a new quadrilateral into the MySQL database*

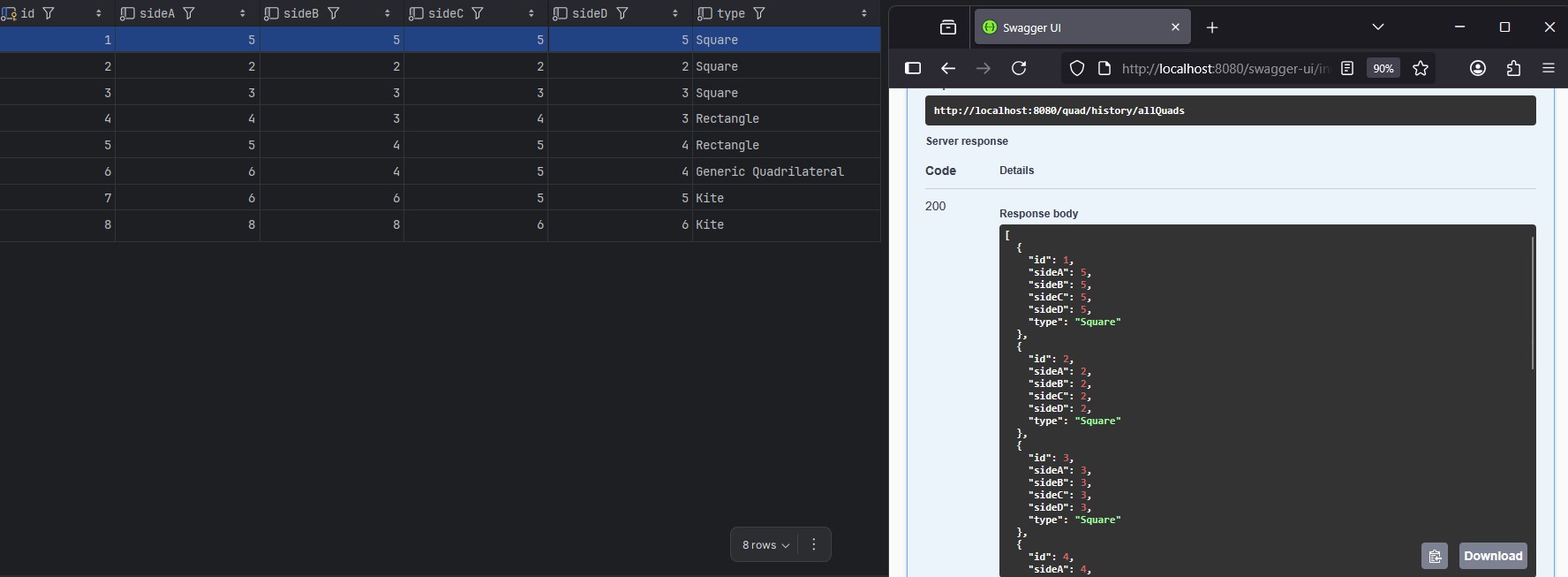
## Screenshot 2 – GET Last

*Returns the most recently inserted quadrilateral*

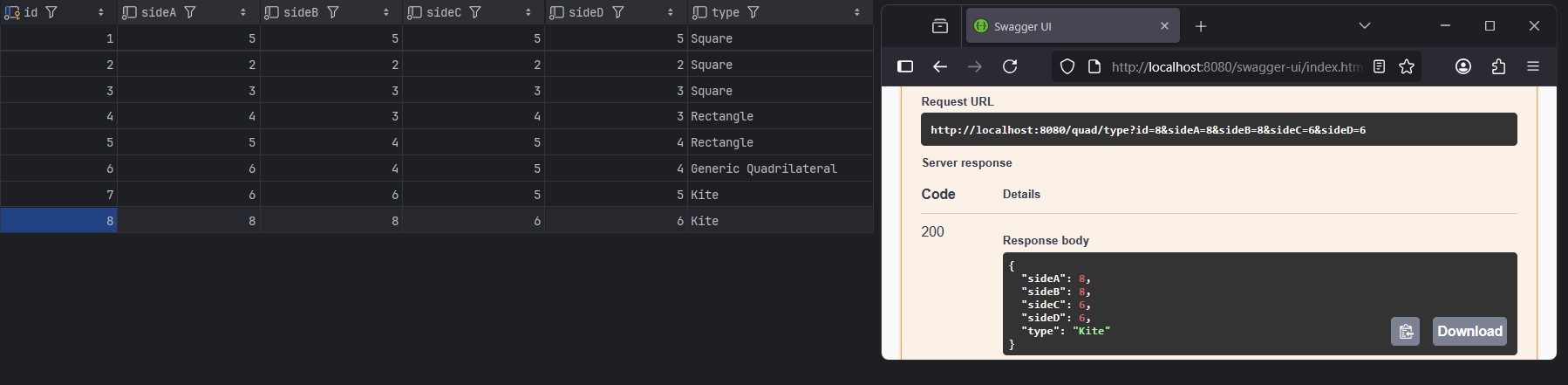
## Screenshot 3 – GET Stats

*Returns all types of quads in DB*

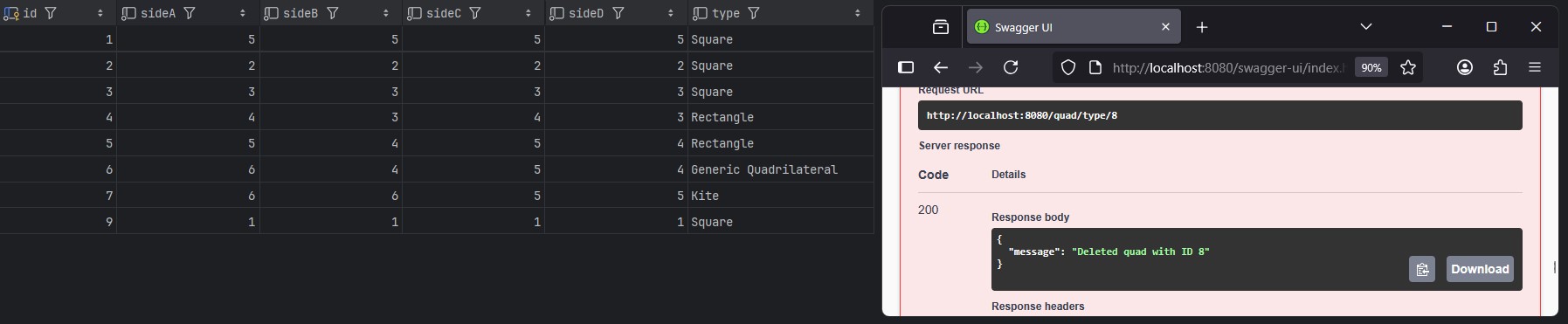
## Screenshot 4 – GET All

*Returns all quads submitted to DB*

## Screenshot 5 - PUT

*Updates quad by ID*

## Screenshot 6 - DELETE /type/id

*Deletes a record by ID*

## Refactored Files & Added Files

* **QuadServiceHibernateImpl.java**
  + New Hibernate-based implementation of QuadService using Spring Data JPA. Handles all database operations through a QuadRepository. Replaces manual SQL queries with concise repository calls and stream processing.
* **QuadRepository.java**
  + JPA repository interface that abstracts database operations on the Quadrilateral entity. Used by the Hibernate service.
* **QuadService.java**
  + Shared service interface now implemented by QuadInMemoryImpl, QuadServiceJdbcImpl, and QuadServiceHibernateImpl. Promotes modularity and testability across implementations.
* **Quadrilateral.java**
  + JPA entity representing a four-sided shape. Includes validation logic and type-detection logic. Annotated for Hibernate persistence.
* **QuadServiceIntegrationTest.java**
  + Extended to test both JDBC and Hibernate implementations by swapping service qualifiers. All major database operations are verified under the test profile.
* **application.properties**
  + Refers to the active MySQL database connection when using the dev profile. Hibernate DDL generation can be enabled if schema auto-creation is needed.
* **application-test.properties**
  + Defines the H2 in-memory environment used during testing. Ensures isolation from the production database.
* **schema.sql**
  + Retained for compatibility with the JDBC service or for initializing the schema during testing.

## Known Limitations & Future Enhancements

* The Hibernate version does not currently persist calculated shape types. Future enhancements may include storing the type as a derived column or via entity listeners.
* Controller logic relies on isInitialized() checks; these are now implemented in both JDBC and Hibernate services.
* Angle-based classification (e.g., rhombus vs. square) is not yet supported. This feature is planned in future iterations
* Additional user feedback and validation will be added to guard against invalid or duplicate input.
* The project currently lacks authentication and frontend filtering; these concerns will be addressed in a future sprint.

## Notes

* The project is maintained in a public GitHub repository for source control and demonstration purposes.
* The submission .zip includes all .java source files and a /Docs folder per course requirements.
* Screenshots demonstrating successful compilation, logs, and Swagger execution are included in this document.
* All assets are stored in:
  + Docs/Week4/assets/