**Module 4 Activity 5: Object Relational Mapping (ORM)**

Saurabh Kale

IFT 458/554: Middleware Programming & Database Security

Dinesh Sthapit

Oct 6th, 2023

* What is ORM?

An [ORM](https://www.prisma.io/dataguide/intro/database-glossary#orm), or Object Relational Mapper, is a piece of software designed to translate between the data representations used by databases and those used in object-oriented programming. Basically, these two ways of working with data don't naturally fit together, so an ORM attempts to bridge the gap between the two systems' data designs.

Reference-https://www.prisma.io/dataguide/types/relational/what-is-an-orm#what-is-an-orm

* What are the benefits of using ORM?

Object-Relational Mapping (ORM) can be a valuable tool when managing complex state in object-oriented programming, especially when dealing with intricate inheritance relationships and schema migrations. ORMs simplify project development and data structure changes. However, they can introduce debugging challenges due to their abstraction level and occasionally inaccurate translations between the database and application. Assess your project's needs and resource allocation carefully to determine if an ORM aligns with your goals and adds value to your database-backed application development.

Reference-https://www.prisma.io/dataguide/types/relational/what-is-an-orm#what-is-an-orm

* What technology is PRIMA ORM replacing in your previous assignments?

Prisma's primary objective is to enhance the productivity of application developers in their database-related tasks. It accomplishes this goal through various means, including encouraging developers to think in terms of objects instead of dealing with the complexities of mapping relational data. Prisma prioritizes queries over classes, reducing the intricacies of model objects. It serves as a central source of truth for both database and application models, enforcing valuable constraints to prevent common mistakes and anti-patterns. Prisma offers an abstraction that simplifies the right actions, promoting a "pit of success" approach. It also provides type-safe database queries that can be validated during compilation, minimizing errors. With less boilerplate code, developers can focus on the critical aspects of their application, benefiting from auto-completion in code editors instead of constantly referring to documentation.

Reference-https://www.prisma.io/docs/concepts/overview/why-prisma#:~:text=Prisma's%20main%20goal%20is%20to,to%20avoid%20complex%20model%20objects

**Prisma Sample Project-**

**Feed path-**

**A screenshot of a computer

Description automatically generated**

**Users-**A screenshot of a computer

Description automatically generated

**/post/:id-**

A screenshot of a computer

Description automatically generated

**Searchstring-**

A screenshot of a computer

Description automatically generated

**Drafts-**

A screenshot of a computer

Description automatically generated

Users-

A screenshot of a computer

Description automatically generated

Post Create New Post-

A screenshot of a computer

Description automatically generated

Create New User-

A screenshot of a computer

Description automatically generated

Change the value of publish using PUT-

A screenshot of a computer

Description automatically generated

Increase the Post count by 1 PUT-

A screenshot of a computer

Description automatically generated

Delete a post-

A screenshot of a computer

Description automatically generated

**User Assignment-**

**Get All users-**

A screenshot of a computer

Description automatically generated

**POST User-**

**A screenshot of a computer

Description automatically generated**

**Get specific User-**

**A screenshot of a computer

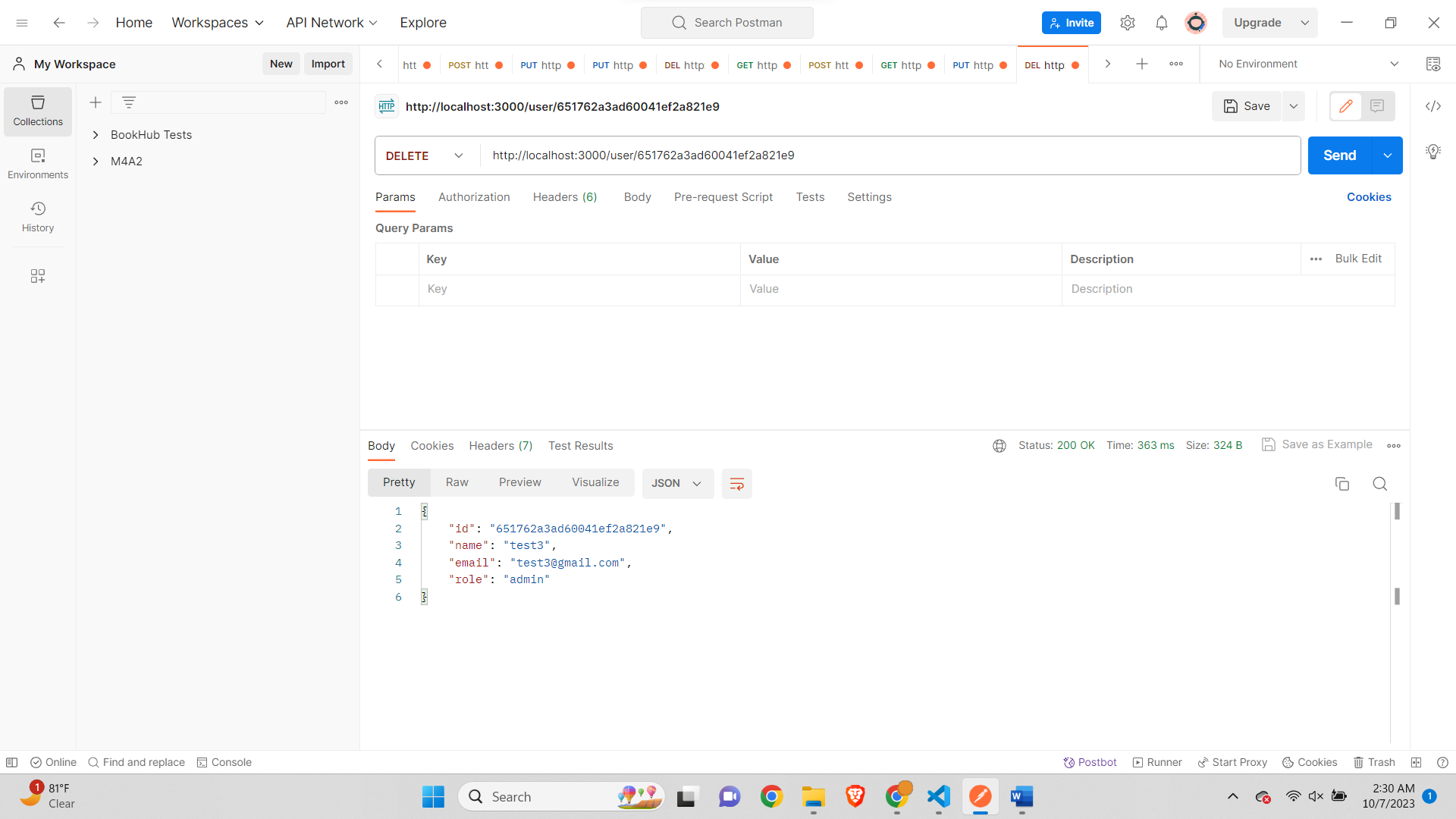
Description automatically generated**

**Update User-**

**A screenshot of a computer

Description automatically generated**

**DELETE Specific User-**

****