How to Use a RESTful Microservice for Exchanging JSON Data with RDBMS on Azure

RESTful microservices and JSON are essential for scalable and efficient applications. Azure provides robust solutions for relational database management systems (RDBMS) such as MySQL, PostgreSQL, and SQL Server. This guide walks you through setting up a RESTful microservice to exchange JSON data with an RDBMS on Azure.

Prerequisites

1. Azure Account: Ensure you have an active Azure account.
2. RDBMS Instance on Azure: Set up a MySQL, PostgreSQL, or SQL Server instance on Azure.
3. Development Environment: Install development tools (e.g., Node.js, Python, or any language of your choice).

Step 1: Setting Up RDBMS on Azure

1. Create an RDBMS Instance:
   - Log in to the Azure portal.
   - Navigate to "Create a resource" > "Databases" > "Azure Database for MySQL/PostgreSQL/SQL Server".
   - Fill in details such as database name, server name, and configurations.
   - Set up the admin account and password.
   - Review and create the instance.

2. Configure the Database:
   - Go to the instance page once it's created.
   - Set up firewall rules for access from your development machine.
   - Create a database and necessary tables using the Azure portal or a database client (e.g., MySQL Workbench, pgAdmin, SQL Server Management Studio).

Step 2: Setting Up the Development Environment

1. Install Gilhari:

-Follow the installation instructions from the Gilhari documentation.

2. Set Up Your Project:

-Create a new project directory.
-Initialize your project according to Gilhari's setup guidelines.
-Install necessary dependencies for database interaction (e.g., mysql2, pg, mssql).

Step 3: Building the RESTful Microservice with Gilhari

1. Create the Basic Server:
-Set up a basic server using Gilhari.
-Example for setting up Gilhari:

2. Connect to the Database:
-Establish a connection to your Azure RDBMS.
-Example for MySQL in Gilhari:

3. Create CRUD Endpoints:
-Define endpoints for Create, Read, Update, and Delete (CRUD) operations.

Step 4: Testing the Microservice

1. Use Postman or Curl:
-Test your endpoints using tools like Postman or Curl.
-Ensure each endpoint performs the CRUD operations correctly.

2. Handle Errors and Edge Cases:
-Implement error handling for invalid requests or database connection issues.

Step 5: Deploying the Microservice on Azure

1. Create an App Service:
   - In the Azure portal, navigate to "Create a resource" > "App Services".
   - Fill in the necessary details and create the service.

2. Deploy Your Code:
   - Use GitHub Actions, Azure DevOps, or any CI/CD tool to automate deployment.
   - Alternatively, manually deploy using FTP, ZIP deployment, or Visual Studio Code extensions.

3. Configure Environment Variables:
   - Set up database connection strings and other configuration settings as environment variables in the Azure App Service.

Conclusion

By following these steps, you can create a RESTful microservice using Gilhari that facilitates easy JSON data exchange with an RDBMS on Azure. This setup leverages Azure's scalability and reliability while adhering to modern web development best practices. Now, you can focus on building more complex and feature-rich applications.

Further reading:

https://www.softwaretree.com/v1/products/gilhari/gilhari-restfulAPI.php

https://warped-rocket-9192.postman.co/workspace/Team-Workspace~a872b0cc-7dae-40af-872c-aec9a1b2d643/collection/5675780-eade11a5-d5d7-4475-8037-8f8461238d6b?action=share&creator=5675780

https://www.softwaretree.com/v1/products/gilhari/gilhari-usage-tips.php

https://www.softwaretree.com/v1/products/gilhari/download-gilhari.php