SCC Project 1 Report

Afonso Proença

59158

Introduction

The Tukano project explores how cloud computing services can support scalable, high-performance web applications by leveraging Microsoft's Azure PaaS offerings. The primary goal is to migrate an existing social media application backend to Azure, aligning it with cloud engineering best practices.

This migration involves porting three main REST services—Users, Shorts, and Blobs—to Azure, leveraging Azure Blob Storage, Cosmos DB, and Redis for optimized storage and caching. The project targets two deployment scenarios: one for a continental user base and another for a globally distributed audience, to evaluate scalability and performance under various configurations.

Architecture

TuKano is structured as a three-tier architecture where the application layer hosts the REST services:

- Users: Manages user data.
- Shorts: Manages metadata and social interactions (likes, follows).
- Blobs: Manages multimedia blobs, such as video files.

In this setup:

- Azure Blob Storage replaces the traditional filesystem, allowing scalable storage of video files.
- Azure Cosmos DB supports the Shorts and Users services, providing a SQL (PostgreSQL) solution with high availability.
- Azure Redis Cache enhances performance, particularly for frequently accessed data, such as user feeds.

The database configuration is set in hibernate.cfg.xml, using Cosmos DB for both SQL and NoSQL options.

Source Code and Endpoints

The Java source code comprises:

- REST API classes (e.g., RestBlobs, RestShorts, RestUsers) within tukano.api.rest.
- Implementation classes within tukano.impl, which encapsulate the business logic for Users, Shorts, and Blobs.
- Utility classes for handling common operations, such as JSON, DB, and Hash.

These services are integrated in a single server without a discovery mechanism, directly referenced through method calls instead of REST.

All endpoints are documented in tukano.api.rest, and the provided Java interfaces model each service's abstract behavior. This setup allows future flexibility in using Azure-specific services while keeping the endpoints compatible with the existing client applications.

Testing

The REST endpoints were tested thoroughly in Postman, covering scenarios for uploading videos, managing user interactions, handling media blobs etc.

Challenges

- Azure Integration: Initial difficulties in adapting the application to Azure's cloud environment posed significant delays. Integrating Cosmos DB, Blob Storage, and Redis required careful configuration and troubleshooting.
- Debugging 500 Errors: Encountering HTTP 500 errors during endpoint testing on Postman proved challenging. The errors were often due to misconfigurations in Azure services or issues with token expiration for secure URL generation.

Limitations

Due to time constraints and challenges encountered in the Azure environment, the implementation was limited to support 13 values only. Efforts to extend beyond this limit encountered resource bottlenecks and unexpected errors. These issues require further investigation.

Conclusion

Tukano demonstrates how cloud-native design patterns and Azure services can enhance scalability and performance in a modern social media application. Despite the limited scope in this iteration, the groundwork has been laid for a fully scalable application, with lessons learned about Azure integration and endpoint testing.