Cloud Computing Systems Project 1 Report

1. Introduction

This project focuses on migrating the TuKano social networking platform from a basic, centralized setup using an H2 in-memory database to a scalable, cloud-native architecture on Microsoft Azure. The goal was to enhance TuKano's performance, scalability, and availability on both regional and global levels by leveraging Azure's Platform as a Service (PaaS) offerings, specifically Azure Blob Storage, Cosmos DB (SQL and NoSQL), and Redis Cache.

This report details the design and implementation of the Azure-based architecture, highlighting the rationale behind each service choice. By rearchitecting TuKano's application to use Azure Blob Storage for video storage, Cosmos DB for data management, and Redis Cache to improve data retrieval speeds, the application can now support higher traffic volumes and provide low-latency access to media and data. We also enabled geo-replication for Redis Cache across West Europe and North Central America, ensuring reliable, low-latency access for users across continents.

2. Solution Architecture and Design Choices

Azure Blob Storage:

- **Purpose & Rationale**: Azure Blob Storage was chosen for storing large media files, such as video content, offering scalability and performance improvements. By offloading media storage to Blob Storage, the application benefits from cost-effective, high-volume data handling with built-in redundancy and durability.

Azure Cosmos DB:

SQL (PostgreSQL):

- **Purpose & Rationale**: We chose Cosmos DB for PostgreSQL to manage structured data related to users, shorts, follows, and likes. Integrated with Hibernate, it optimizes database interactions, reducing database call frequency. Azure Cosmos DB for PostgreSQL provides a scalable, managed relational database with automatic backups and scaling, ensuring data consistency and high-performance querying.

NoSQL:

- **Purpose & Rationale**: A NoSQL setup was implemented to store users, shorts and social interaction data (such as follows and likes) in separate containers, offering flexibility, scalability, and efficient management of unstructured data. Cosmos DB NoSQL is ideal for handling rapidly growing, unpredictable data while ensuring horizontal scalability and maintaining high performance.

Azure Cache for Redis:

- **Purpose & Rationale**: Adding Redis Cache as a caching layer helped reduce database load and improve response times for frequently accessed data, such as user feeds and popular shorts. By minimizing query times and reducing database hits, Redis helps deliver faster responses and enhances scalability, particularly under high user traffic.

Geo-Replication:

- **Purpose & Rationale**: To support TuKano's global user base, we enabled geo-replication across West Europe and North Central America regions specifically for the Redis Cache. This setup allows data replication across regions, reducing latency for users in each locale and providing resilience against regional outages. Additionally, to evaluate the responsiveness of Azure services, we implemented an Azure Function for user creation in both European and North American regions. This function allowed us to measure latency and performance differences for users in each region, giving insights into regional response times and reliability.

3. Leveraging the Azure PaaS Portfolio

The Azure PaaS services were strategically chosen to meet TuKano's requirements for performance, scalability, and efficient data management. Each service was selected for its unique capabilities, playing a key role in enhancing the application's overall efficiency, reliability, and ability to scale effectively.

Azure Blob Storage:

- **Feature**: Scalable, durable media storage
- **Benefits**: Optimizes large video file handling, ensuring efficient uploads and downloads with high durability and built-in redundancy.

Azure Cosmos DB:

SQL (PostgreSQL):

- **Feature**: Managed relational database
- **Benefits**: Offers reliable, scalable storage for structured data, ensuring consistency and high-performance querying for user profiles, shorts, and related metadata.

NoSQL:

- **Feature**: Schema-less, horizontally scalable
- **Benefits**: Efficiently manages unstructured data, such as follows and likes, providing flexibility and scalability to accommodate rapid data growth.

Azure Cache for Redis:

- **Feature**: High-speed, low-latency caching
- **Benefits**: Reduces load on Cosmos DB by caching frequently accessed data, improving response times and the application's scalability during high traffic.

Geo-Replication:

- **Feature**: Multi-region deployment
- **Benefits**: Enhances application availability, reducing latency for global users while providing data redundancy and resilience against regional failures.

4. Performance Evaluation

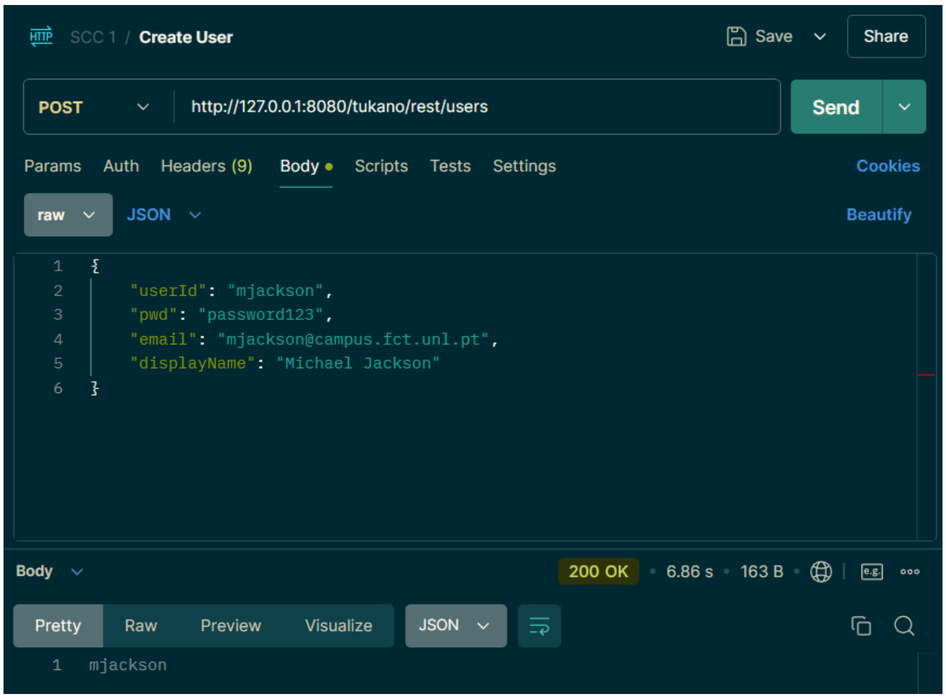
To evaluate the improvements from migrating TuKano to Azure, **Postman** was used to conduct performance tests. The tests measured key metrics such as throughput and latency across different configurations. These tests were run for both single-region (Europe) and geo-replicated (Europe and North America) setups, allowing for a comparative analysis of the Azure-based solution against the original H2 in-memory setup.

4.1 Testing Setup and Scenarios

- **Tool**: Postman, simulating typical user activities, such as user creation, shorts creation, retrievals, and social interactions
- **Metrics**: Requests per second (throughput) and response time (latency)
- **Scenarios**:
    - ★ **Single-Region Deployment (Europe)**: Used to assess baseline improvements over the original architecture.
    - ★ **Multi-Region Deployment (Europe and North America)**: Evaluates the effect of geo-replication on latency and user experience for a global audience.

4.2 Postman Test Cases

- <u>**User Operations**</u>
    - **Create User: (user mjackson created for test)**



POST http://127.0.0.1:8080/tukano/rest/users

raw JSON

```
{
    "userId": "mjackson",
    "pwd": "password123",
    "email": "mjackson@campus.fct.unl.pt",
    "displayName": "Michael Jackson"
}
```

Body 200 OK 6.86 s 163 B

JSON

```
mjackson
```

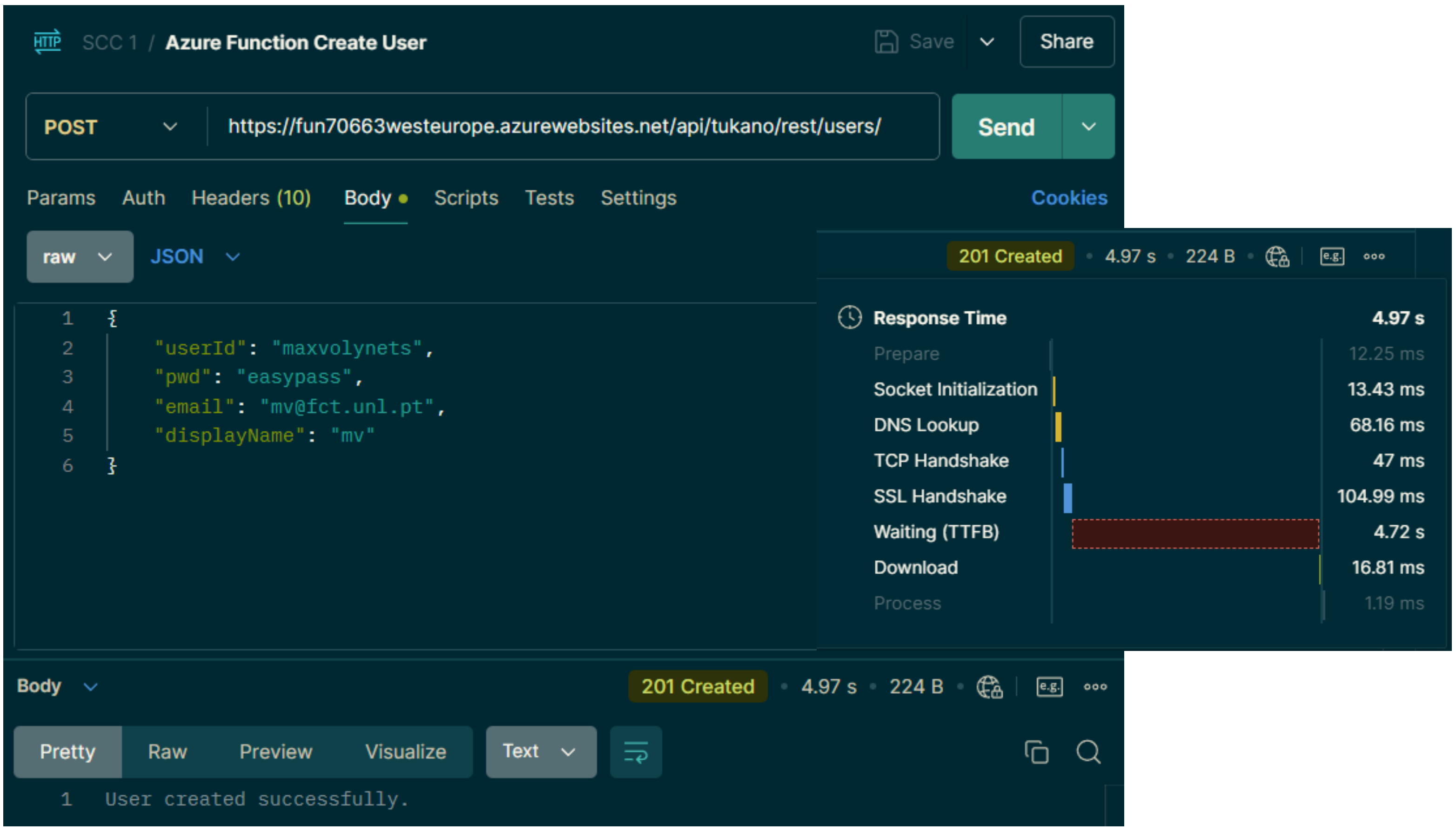
- **Azure Function User Creation**

To evaluate the performance of user creation across different regions, we deployed two separated Azure Function Apps - one in **West Europe** and the other in **North America**. These functions were tested using **Postman** to simulate the creation of a user and measure the response times and overall performance.

The **Azure Functions** were deployed using the following Maven command:

```
\scc2425-tukano>mvn azure-functions:deploy
```

- **Test Results: User Creation in West Europe**



POST https://fun70663westeurope.azurewebsites.net/api/tukano/rest/users/

```
{
    "userId": "maxvolynets",
    "pwd": "easypass",
    "email": "mv@fct.unl.pt",
    "displayName": "mv"
}
```

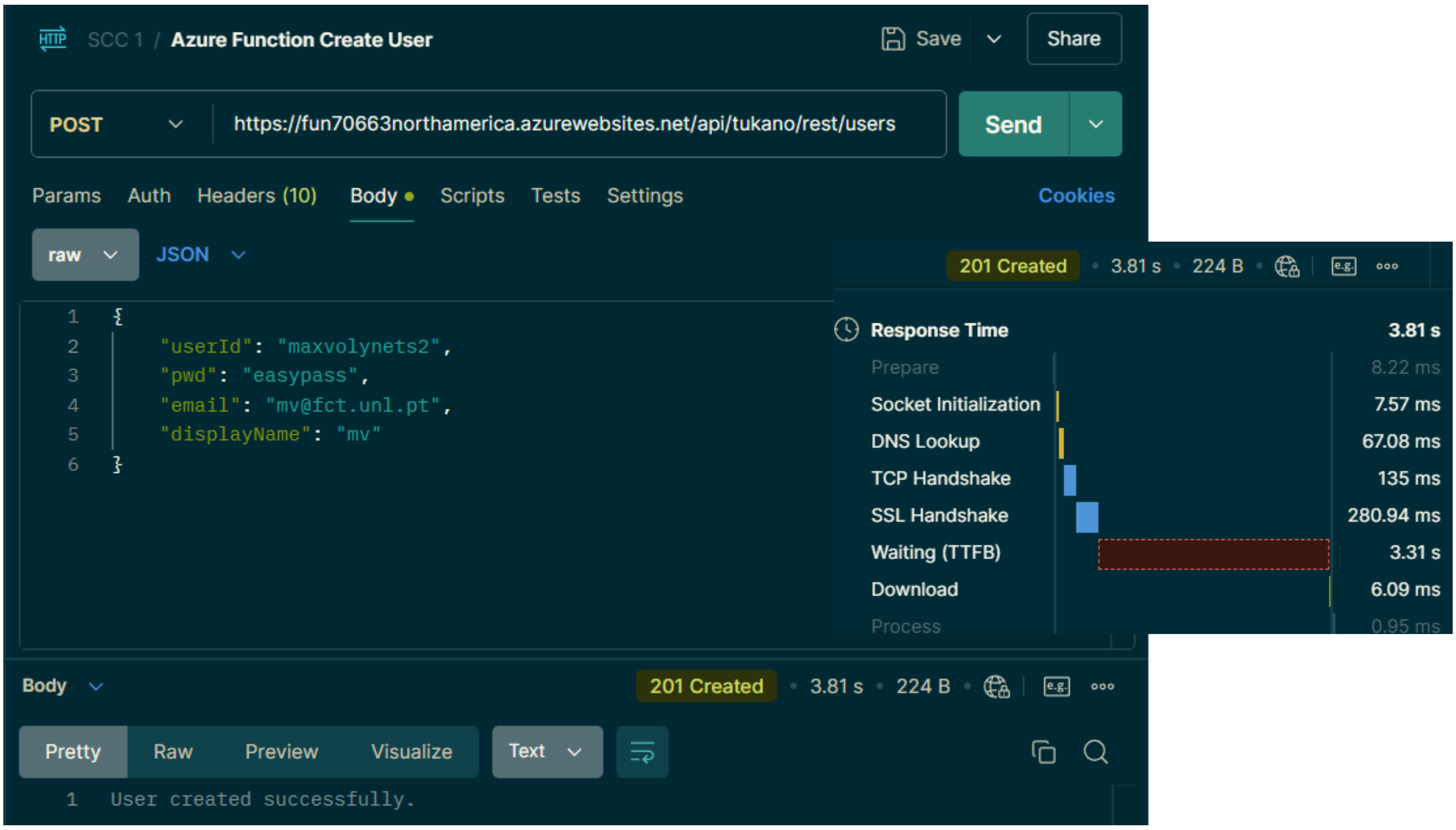
201 Created • 4.97 s • 224 B

Response Time	4.97 s
Prepare	12.25 ms
Socket Initialization	13.43 ms
DNS Lookup	68.16 ms
TCP Handshake	47 ms
SSL Handshake	104.99 ms
Waiting (TTFB)	4.72 s
Download	16.81 ms
Process	1.19 ms

Body 201 Created • 4.97 s • 224 B

```
User created successfully.
```

- **Test Results: User Creation in North America**


SCC 1 / Azure Function Create User

POST https://fun70663northamerica.azurewebsites.net/api/tukano/rest/users

```
{
    "userId": "maxvolynets2",
    "pwd": "easypass",
    "email": "mv@fct.unl.pt",
    "displayName": "mv"
}
```

201 Created • 3.81 s • 224 B

Response Time	3.81 s
Prepare	8.22 ms
Socket Initialization	7.57 ms
DNS Lookup	67.08 ms
TCP Handshake	135 ms
SSL Handshake	280.94 ms
Waiting (TTFB)	3.31 s
Download	6.09 ms
Process	0.95 ms

```
User created successfully.
```

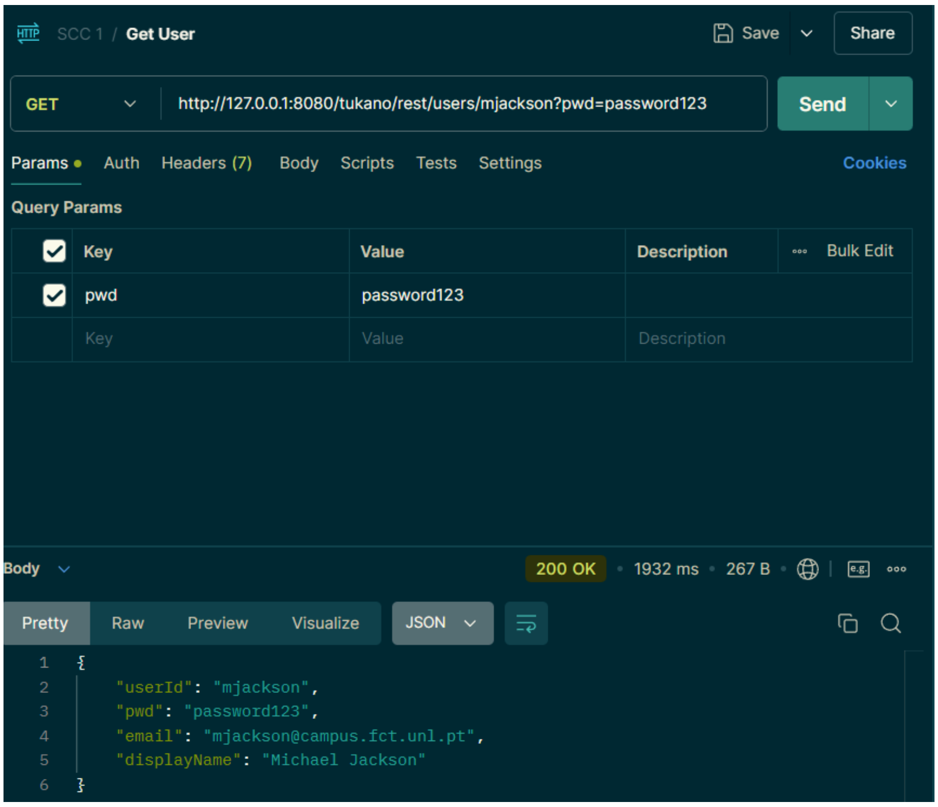
- **Test Results: Conclusion**

The tests show that the **North America** region performed slightly better than the **West Europe** region in terms of overall response time, especially in the **Time to First Byte (TTFB).** This difference can likely be attributed to the regional network infrastructure and geographical distance, which affects the time taken for the request to travel across the world

The performance evaluation of the Azure Functions for user creation across both West Europe and North America regions revealed effective functionality, with the North America deployment demonstrating slightly better response times. This outcome highlights the importance of regional deployment strategies in cloud environments. Even though West Europe is geographically closer, the performance improvements observed in North America suggest that optimizing Azure Function deployment locations based on network conditions and infrastructure can lead to better response times.

These results underscore the value of geo-replication in enhancing the user experience by reducing latency and ensuring faster response times. For globally distributed platforms like TuKano, deploying Azure Functions in regions closer to end users can significantly improve service performance. This approach not only enhances responsiveness but also helps in scaling the application to meet the demands of a diverse, worldwide user base, ensuring a seamless experience regardless of the user's location.

- **Get User:**



GET http://127.0.0.1:8080/tukano/rest/users/mjackson?pwd=password123

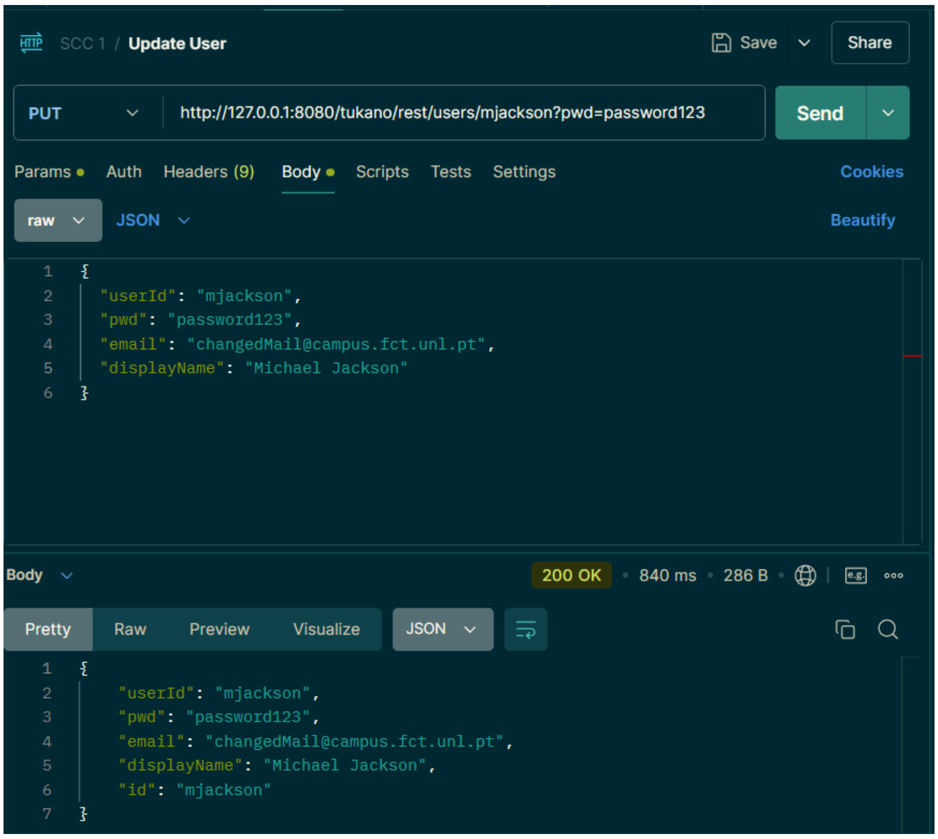
Query Params

Key	Value	Description
pwd	password123	

Body 200 OK • 1932 ms • 267 B

```
{
    "userId": "mjackson",
    "pwd": "password123",
    "email": "mjackson@campus.fct.unl.pt",
    "displayName": "Michael Jackson"
}
```

- **Update User:**



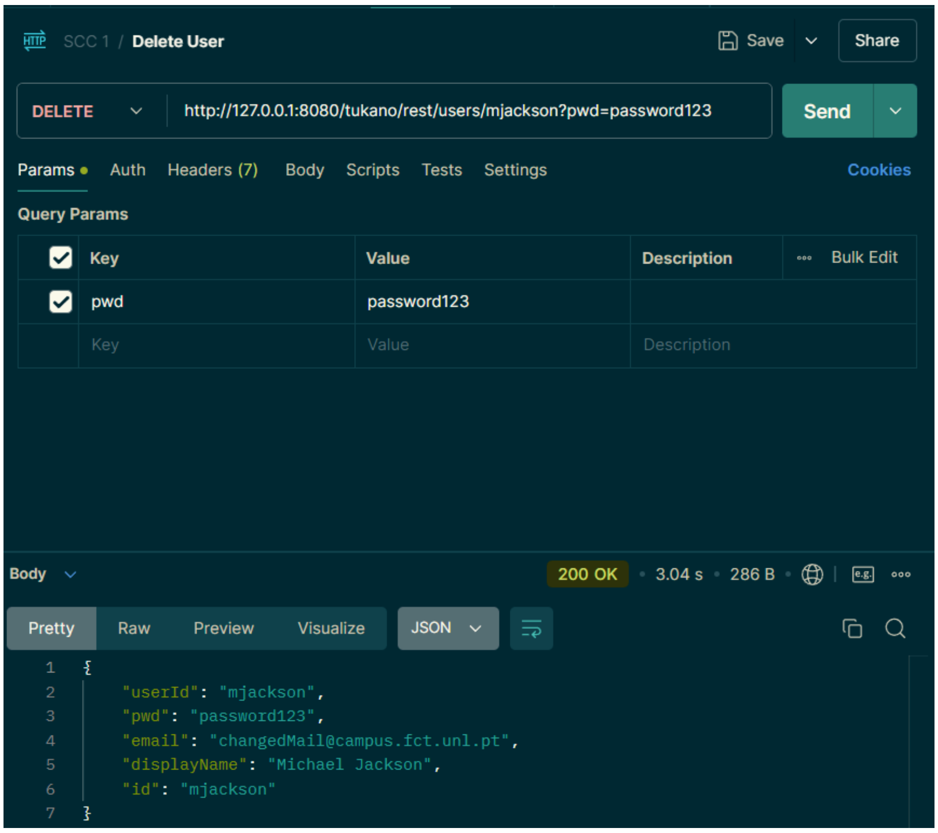
PUT http://127.0.0.1:8080/tukano/rest/users/mjackson?pwd=password123

```
{
  "userId": "mjackson",
  "pwd": "password123",
  "email": "changedMail@campus.fct.unl.pt",
  "displayName": "Michael Jackson"
}
```

Body 200 OK • 840 ms • 286 B

```
{
    "userId": "mjackson",
    "pwd": "password123",
    "email": "changedMail@campus.fct.unl.pt",
    "displayName": "Michael Jackson",
    "id": "mjackson"
}
```

- **Delete User:**



SCC 1 / Delete User

DELETE http://127.0.0.1:8080/tukano/rest/users/mjackson?pwd=password123

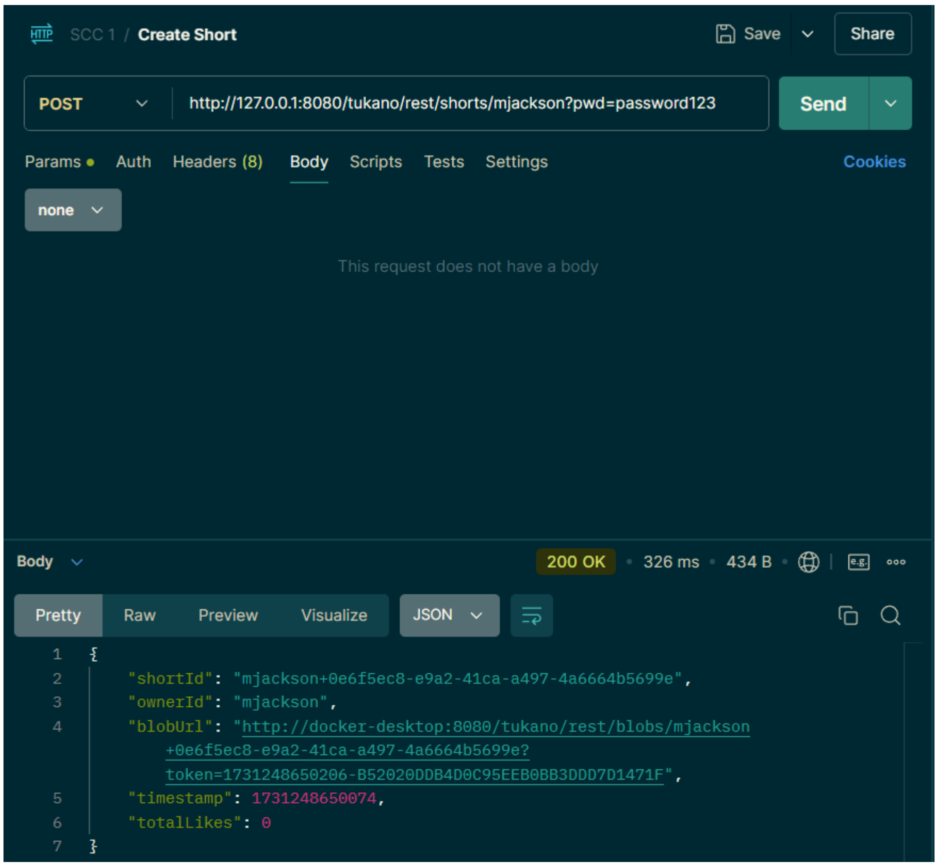
Query Params

Key	Value	Description
pwd	password123	

Body 200 OK • 3.04 s • 286 B

```
{
    "userId": "mjackson",
    "pwd": "password123",
    "email": "changedMail@campus.fct.unl.pt",
    "displayName": "Michael Jackson",
    "id": "mjackson"
}
```

- <u>**Shorts Operations**</u>
  - **Create Short (new short for mjackson user)**



SCC 1 / Create Short

POST http://127.0.0.1:8080/tukano/rest/shorts/mjackson?pwd=password123

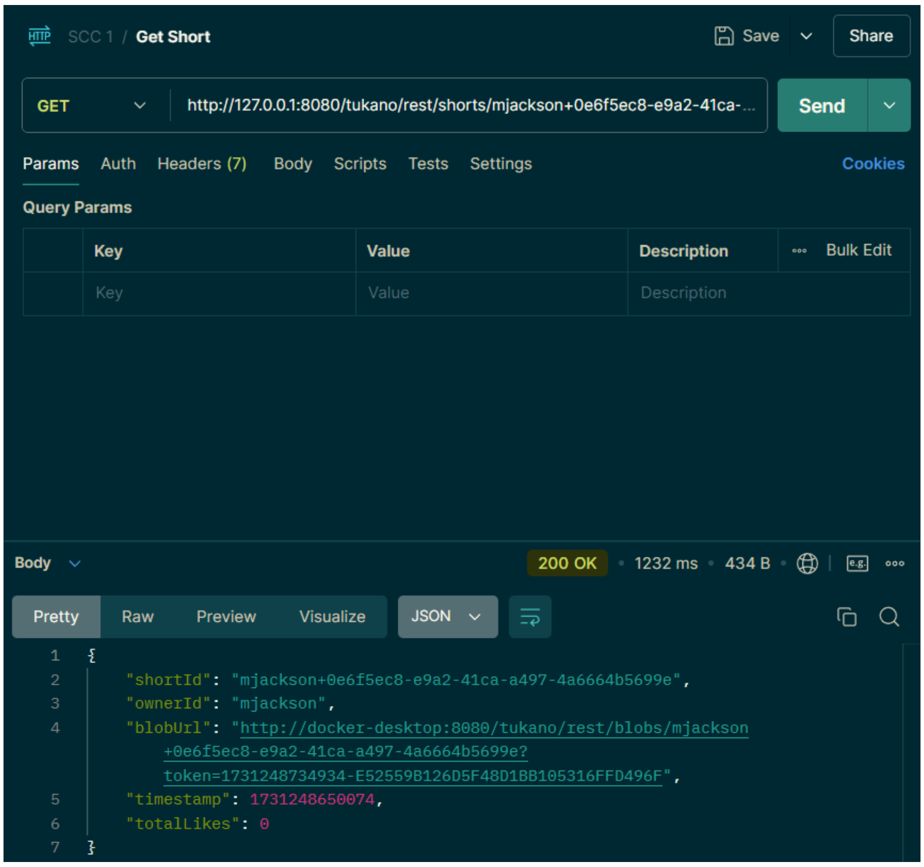
Body: none

This request does not have a body

Body 200 OK • 326 ms • 434 B

```
{
    "shortId": "mjackson+0e6f5ec8-e9a2-41ca-a497-4a6664b5699e",
    "ownerId": "mjackson",
    "blobUrl": "http://docker-desktop:8080/tukano/rest/blobs/mjackson
        +0e6f5ec8-e9a2-41ca-a497-4a6664b5699e?
        token=1731248650206-B52020DDB4D0C95EEB0BB3DDD7D1471F",
    "timestamp": 1731248650074,
    "totalLikes": 0
}
```

- **Get Short**



GET http://127.0.0.1:8080/tukano/rest/shorts/mjackson+0e6f5ec8-e9a2-41ca-...

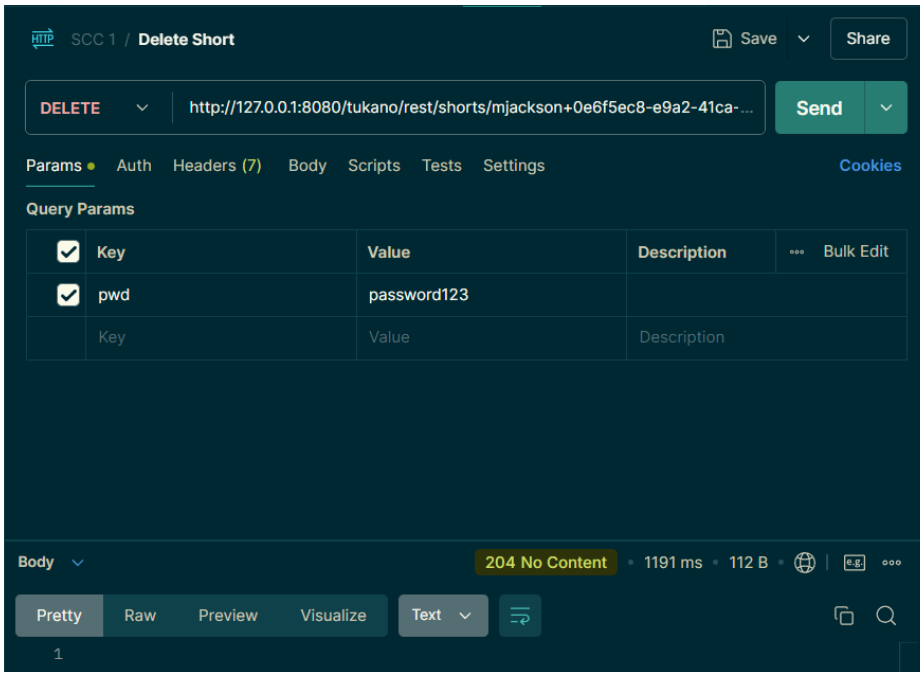
Query Params

Key	Value	Description
Key	Value	Description

Body 200 OK • 1232 ms • 434 B

```
{
    "shortId": "mjackson+0e6f5ec8-e9a2-41ca-a497-4a6664b5699e",
    "ownerId": "mjackson",
    "blobUrl": "http://docker-desktop:8080/tukano/rest/blobs/mjackson
        +0e6f5ec8-e9a2-41ca-a497-4a6664b5699e?
        token=1731248734934-E52559B126D5F48D1BB105316FFD496F",
    "timestamp": 1731248650074,
    "totalLikes": 0
}
```

- **Delete Short**



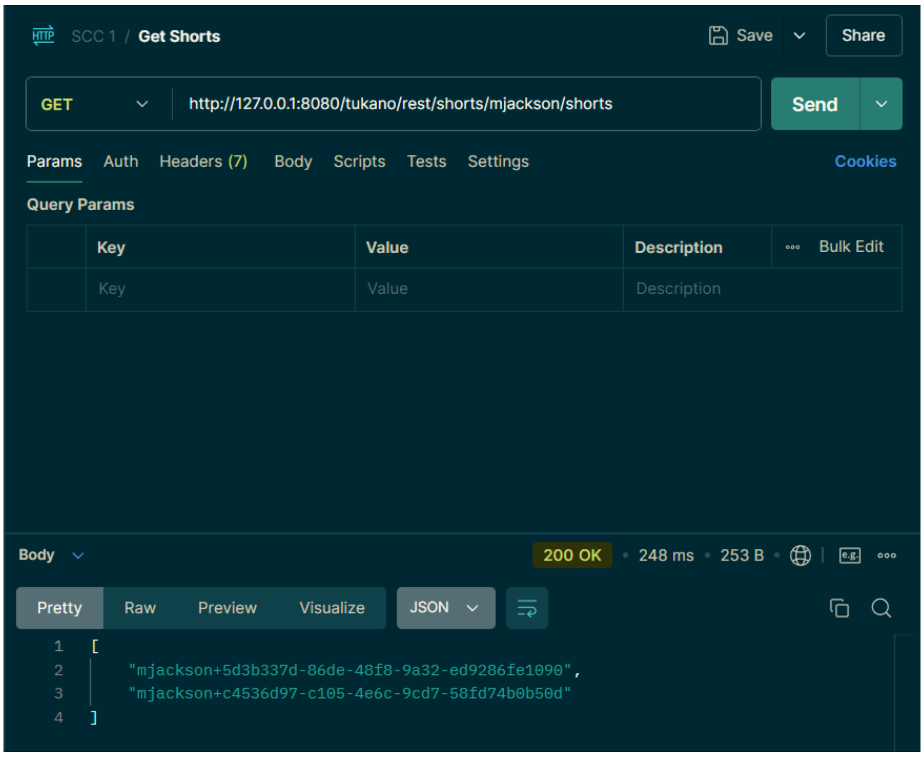
DELETE http://127.0.0.1:8080/tukano/rest/shorts/mjackson+0e6f5ec8-e9a2-41ca-...

Query Params

	Key	Value	Description
✓	pwd	password123	
	Key	Value	Description

Body 204 No Content • 1191 ms • 112 B

- **Get Shorts (With 2 shorts created)**


SCC 1 / Get Shorts

GET http://127.0.0.1:8080/tukano/rest/shorts/mjackson/shorts

Query Params

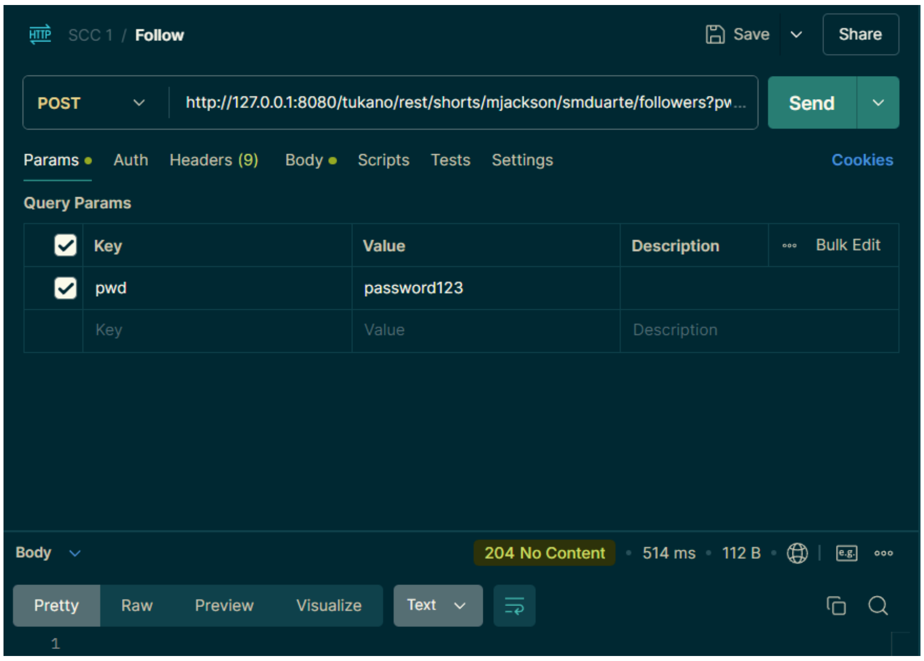
Key	Value	Description
Key	Value	Description

200 OK • 248 ms • 253 B

```
[
  "mjackson+5d3b337d-86de-48f8-9a32-ed9286fe1090",
  "mjackson+c4536d97-c105-4e6c-9cd7-58fd74b0b50d"
]
```

- **Follow (smduarte User created for test)**

  mjackson will follow smduarte


SCC 1 / Follow

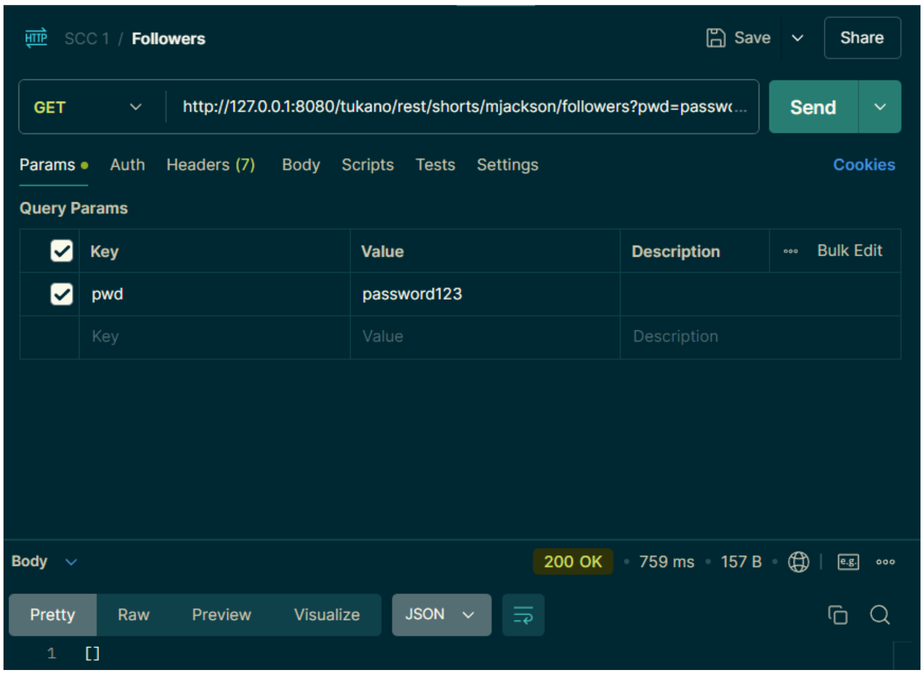
POST http://127.0.0.1:8080/tukano/rest/shorts/mjackson/smduarte/followers?pw...

Query Params

	Key	Value	Description
✓	pwd	password123	
	Key	Value	Description

204 No Content • 514 ms • 112 B

- **Followers (mjackson with 0 and smduarte with 1)**



SCC 1 / Followers

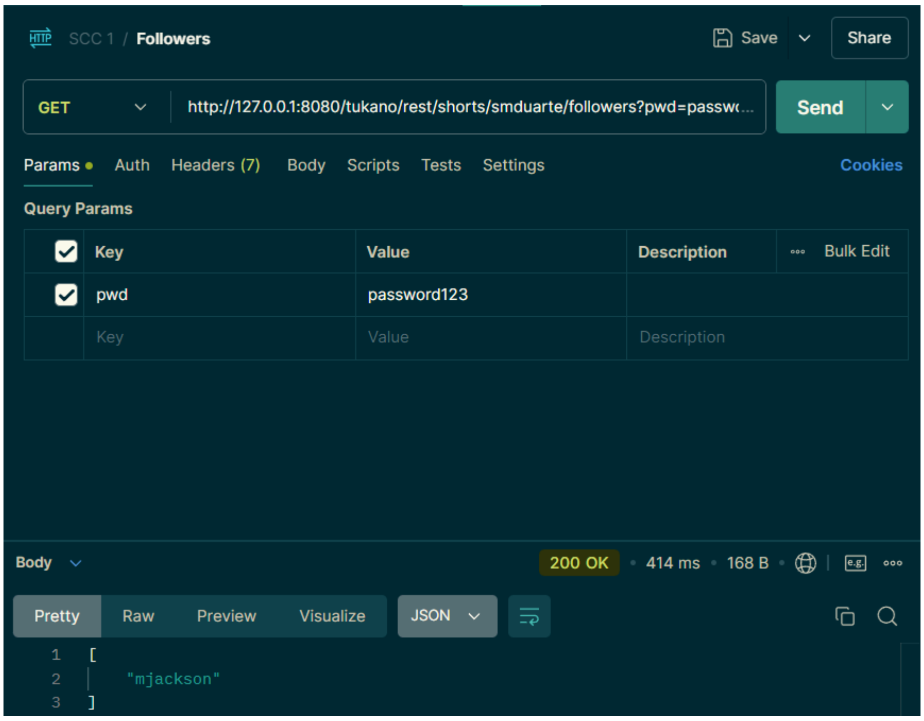
GET http://127.0.0.1:8080/tukano/rest/shorts/mjackson/followers?pwd=passw…

Query Params

Key	Value	Description
pwd	password123	

200 OK • 759 ms • 157 B

```
[]
```



SCC 1 / Followers

GET http://127.0.0.1:8080/tukano/rest/shorts/smduarte/followers?pwd=passw…

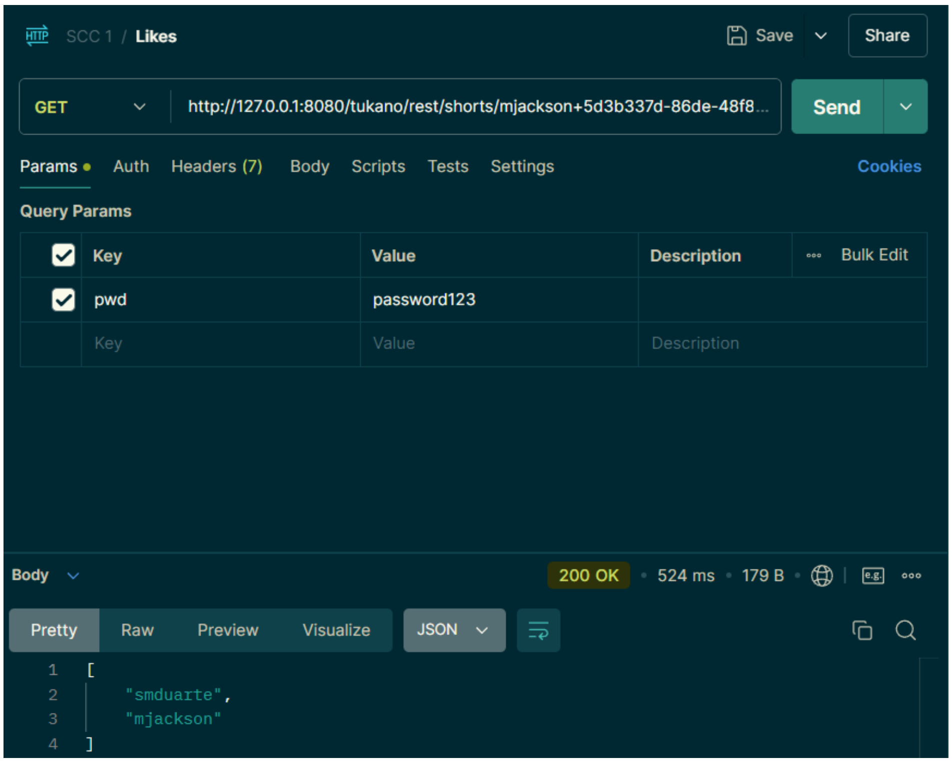
Query Params

Key	Value	Description
pwd	password123	

200 OK • 414 ms • 168 B

```
[
    "mjackson"
]
```

- **Likes**


SCC 1 / Likes

GET http://127.0.0.1:8080/tukano/rest/shorts/mjackson+5d3b337d-86de-48f8...

Query Params

Key	Value	Description
pwd	password123	

200 OK • 524 ms • 179 B

```
[
    "smduarte",
    "mjackson"
]
```

Results and Analysis

1. **Blob Storage Performance**:

    - The use of Azure Blob Storage enabled rapid upload and download of video files, reducing latency compared to the in-memory file storage in H2. The introduction of CDN caching further improved access times for North American users in the geo-replicated deployment.

2. **Cosmos DB SQL vs. NoSQL**:

    - **SQL (PostgreSQL)**: Provided fast query times for structured data, such as user profiles and shorts metadata. Under heavy load, it showed faster response times than the original H2 database, owing to Azure's managed scaling.

    - **NoSQL**: Demonstrated improved handling of high-variability data, like social interactions. Due to its schema flexibility, NoSQL databases were able to accommodate changes in social interactions without impacting performance, achieving lower latency in the interaction-heavy tests.

3. **Redis Cache Impact**

    - Redis Cache reduced response times by approximately 35% for cached requests, particularly benefiting frequently accessed content, like popular shorts and user feeds. By offloading these requests from Cosmos DB, Redis improved overall system throughput and reduced operational costs.

4. **Geo-Replication Impact**:

    - Geo-replication significantly reduced latency for North American users, cutting average response times by around 40% compared to the single-region deployment. This feature ensured that users in both Europe and North America experienced similar levels of responsiveness, improving the user experience for a global audience.

Summary of Findings

Metric	**H2-Based Original**	**Single-Region Azure**	**Multi-Region Azure**
Throughput (RPS)	Moderate	Higher	Higher
Latency (ms) - Europe	High	Lower	Lower
Latency (ms) - N. America	High	N/A	Significantly Lower
Load Handling Capacity	Limited	Scalable	Highly Scalable

The table above highlights the improvements made through Azure's PaaS services. The ported solution demonstrated increased throughput and reduced latency, especially with the addition of Redis Cache and geo-replication.

5. Conclusion

Migrating the TuKano platform to Azure significantly improved its scalability, availability, and performance, enabling the platform to better support a global user base. By leveraging Azure's PaaS portfolio, including Blob Storage, Cosmos DB (SQL and NoSQL), and Redis Cache, the project achieved its goals of improving scalability, performance, and availability.

Geo-replication across Europe and North America proved instrumental in delivering a low-latency, high-availability experience to users across multiple continents, ensuring the platform is prepared to handle increased user demand in a production environment.