# SCC Project 2 Report

Afonso Proença

59158

# Introduction

The second phase of the Tukano project focuses on migrating the platform from Azure PaaS to an IaaS environment using **Docker** and **Kubernetes**, deployed on the **Azure Kubernetes**

The primary goal is to replace PaaS services with container-based alternatives while maintaining the architecture and functionalities from the first phase.

Additionally, **secure user authentication** has been implemented using cookies and Redis for session management.

# 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.

Technologies Used:

- Docker: Containerization of each service.
- Kubernetes: Orchestration of services and infrastructure management.
- Azure AKS: Managed Kubernetes cluster in Azure.
- Redis Cache: Used for managing user sessions securely.
- PostgreSQL (with Hibernate): Persistent storage deployed within AKS.
- ThreadLocal Storage: Manages temporary cookie storage in server threads.

# 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.
- GET /login: Returns the login HTML page.
- □ POST /login: Validates user credentials, creates a session stored in Redis, and returns a secure cookie.

# Testing

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

Performance Testing: Conducted using Artillery scripts provided in the lab 6.

Metrics Evaluated:

Latency: Time taken to process requests.

Throughput: Number of requests handled per second.

Results:

Improved latency and throughput due to scalability provided by Kubernetes.

Session management overhead was minimized using Redis.

# Challenges

- Debugging 500 Errors: Encountering HTTP 500 errors during endpoint testing on Postman proved challenging.
- Kubernetes Configuration: Integrating PostgreSQL and Redis into AKS required additional YAML configuration and persistent volumes.
- Authentication Management: Ensuring secure and cookie-based sessions across services.
- Deployment Issues: Debugging container images and resolving compatibility issues with Kubernetes.

# Limitations

Due to time constraints and challenges encountered the implementation was limited to only implementing the mandatory stuff.

# Conclusion

The migration to an IaaS-based solution using Docker and Kubernetes successfully improved the scalability, modularity, and flexibility of the Tukano application. Redis was used effectively for session management, ensuring a secure user experience, while Kubernetes provided a robust environment for orchestration and performance optimization.