**Project documentation for Systemy rozproszone i przetwarzanie równoległe**

**“Application MiniTwitter based on gRPC”**

**Created by: Yevhenii Palamarchuk (126931)**

**2023**

**Introduction**

**Background**  
At the start of this project, I decided to develop a MiniTwitter application with gRPC. As we moved into the project, it became evident that gRPC (gRPC Remote Procedure Calls) and Python would be the cornerstone technologies to achieve this objective. It's essential to note that, at the project's inception, I did not have any prior experience with gRPC, and only limited familiarity with Python was present.

**Objectives**

Mine primary objectives were twofold: to master the intricacies of gRPC, a high-performance, open-source RPC (Remote Procedure Call) framework, and to leverage Python for the server-client communication aspect of our MiniTwitter application. While Python is widely known for its simplicity and versatility, mine unfamiliarity with gRPC posed an initial challenge. The project aimed not only to implement the basic functionality of MiniTwitter but also to significantly enhance my skills in these technologies.

**Team members**  
As the only person in my group, I needed to do all the work for programming and documentation. The main cornerstone was lack of knowledge about gRPC and limited skills in Python, so I decided to work on this week-to-week to improve my understanding of those two major contributors to my project.

**Week 1:**  
 Work on gRPC, get basic knowledge of what is it.  
**Week 2:**  
 I found how can I implement everything and how to tie it together.

**Week 3:**

I created a basic code and created a GitHub repository.

**Week 4:**

My laptop stopped to work so I have got new one (20 Nov 2023) and I had to set everything from zero.

**Week 5:**

Final review, trying to optimize demo.

**Week 6:**  
 Creation of documentation.

**Project Architecture**

**System Overview**

The architecture of MiniTwitter is designed to facilitate efficient and scalable communication between users through the implementation of gRPC, a modern RPC framework. The system comprises two main components: the MiniTwitter server and the MiniTwitter client.

**MiniTwitter Server**

The MiniTwitter server, implemented in Python, serves as the central hub for handling incoming requests from clients. It leverages the gRPC framework to expose remote procedures that clients can invoke. The server maintains a list of messages, allowing clients to send and retrieve messages as needed. The server's architecture is built on top of the MiniTwitterServicer class, responsible for handling gRPC service requests.

**MiniTwitter Client**

The MiniTwitter client, also implemented in Python, communicates with the server through gRPC, sending messages and requesting message retrieval. Clients can customize and send messages, incorporating features such as hashtags for categorization. The client-server communication is asynchronous, allowing for efficient handling of multiple concurrent requests.

**Technologies Used**

The MiniTwitter project employs several key technologies to achieve its objectives.

**gRPC**

gRPC serves as the primary communication protocol, enabling the efficient exchange of data between the MiniTwitter server and clients. It utilizes Protocol Buffers (protobuf) for message serialization and provides features such as bidirectional streaming, making it well-suited for real-time communication.

**Protocol Buffers**

Protocol Buffers are used for defining the message formats exchanged between the server and clients. This serialization format offers a concise and efficient way to structure data, enhancing the speed and interoperability of communication.

**Python**

Python is the chosen programming language for implementing both the MiniTwitter server and client. Its simplicity, readability, and extensive libraries make it an ideal choice for rapid development and ease of collaboration among team members.

**Implementation Details**

**Server Implementation**

**MiniTwitterServicer Class**

The backbone of the MiniTwitter server is the MiniTwitterServicer class. This class is responsible for handling gRPC service requests, defining the behavior for the sendMessage and getMessages operations. Messages sent by clients are processed, and the server maintains an in-memory list of messages to fulfill retrieval requests.  
  
A computer screen shot of a program

Description automatically generated

Figure : MiniTwitterServicer

**gRPC Server Setup**

The server is set up using the grpc.server class with a ThreadPoolExecutor to handle concurrent requests efficiently.

A computer screen shot of a program

Description automatically generated

Figure : gRPC Server Setup

**Client Implementation**

**Sending Messages**

The MiniTwitter client communicates with the server using the MiniTwitterStub class. The client can send messages by creating instances of the Message protocol buffer and invoking the sendMessage operation.

A screenshot of a computer code

Description automatically generated

Figure : Client sending messages

**Retrieving Messages**

To retrieve messages, the client can use the getMessages operation, providing a GetMessagesRequest with the desired number of messages to retrieve.

A computer screen shot of text

Description automatically generated

Figure : Retrieve Messages

**Conclusion**

**Summary of Achievements**

The development of MiniTwitter has been an interesting experience of exploration and learning, marked by significant achievements. Key accomplishments include the successful implementation of a gRPC-based communication system, allowing clients to send and retrieve messages in real-time. The use of Protocol Buffers has enhanced the efficiency of data serialization, contributing to the overall performance of the system.

I navigated the challenges of mastering gRPC and Python, demonstrating adaptability and a commitment to acquiring new skills. The implementation of additional features, such as hashtag support and message categorization, showcases the project's evolution beyond its initial scope.

**Lessons Learned**

Throughout the project, I had encountered challenges that provided valuable learning opportunities. Overcoming initial unfamiliarity with gRPC and Python required a collaborative and iterative approach. The importance of clear communication, consistent coding practices, and version control became evident, contributing to a smoother development process.

The testing phase highlighted the significance of comprehensive testing strategies, including unit testing and integration testing, to ensure the robustness of the system. Feedback received during the intermediary presentation provided valuable insights that influenced subsequent enhancements and optimizations.

**Future Work**

As the project concludes, several avenues for future work present themselves. Further refinement of the MiniTwitter application could involve the implementation of user authentication, message persistence, and expanded support for multimedia content. Additionally, exploration of containerization and deployment strategies could enhance the scalability and deployment flexibility of the system.

In conclusion, the MiniTwitter project stands as a good call for my dedication, adaptability, and collaborative spirit. The acquired skills and experiences will undoubtedly contribute to future projects and endeavors.