**Project Proposal**

**Basic Architecture of gRPC microservice**

**Introduction:**

The purpose of this proposal is to outline the development of a gRPC (Google Remote Procedure Call) Microservice API for a comprehensive product catalog. The proposed API will empower customers to effortlessly browse and access product information. This project aims to demonstrate the efficacy of gRPC technology and its interaction with the internet, highlighting its reliability and performance.

Objectives:

The primary objectives of this project are as follows:

a. Develop a gRPC-based Microservice API for a product catalog.

Showcase the reliability and performance advantages of gRPC technology.

d. Demonstrate the seamless interaction between the API and the internet infrastructure.

The scope of this project will include the following key components:

a. Development of a gRPC Microservice API architecture.

b. Integration with a database system for storing and retrieving product information.

c. Implementation of search functionality for efficient product discovery.

d. Deployment of the gRPC Microservice API on a cloud platform for public access.

Methodology:

To achieve the stated objectives, the following methodology will be employed:

a. Requirement Gathering: Conduct in-depth analysis and discussions with stakeholders to gather functional and non-functional requirements.

b. Architecture Design: Design a scalable and fault-tolerant gRPC Microservice API architecture that meets the project's objectives.

c. Implementation: Develop the gRPC Microservice API using a suitable programming language and framework.

d. Database Integration: Integrate the API with a reliable database system to store and retrieve product information.

e. Search Functionality: Implement an efficient search mechanism to enable customers to discover products quickly.

f. Performance Optimization: Employ optimization techniques to ensure the API's reliability and high performance.

g. Deployment: Deploy the gRPC Microservice API on a cloud platform for easy accessibility and testing.

Deliverables:

The deliverables for this project will include the following:

a. Fully functional gRPC Microservice API for the product catalog.

b. Comprehensive documentation detailing the API's functionalities, architecture, and usage guidelines.

c. Test suite ensuring the reliability and correctness of the API.

d. Deployment guide for easy deployment on a cloud platform.

e. Presentation showcasing the benefits of gRPC technology and its interaction with the internet.

Timeline:

The estimated timeline for the completion of this project is as follows:

Phase 1: Requirements Gathering and Architecture Design - 2 weeks

Phase 2: Implementation and Integration - 4 weeks

Phase 3: Performance Optimization and Testing - 2 weeks

Phase 4: Documentation and Deployment - 1 week

Budget:

The budget for this project will cover the following aspects:

a. Testing infrastructure and tools.

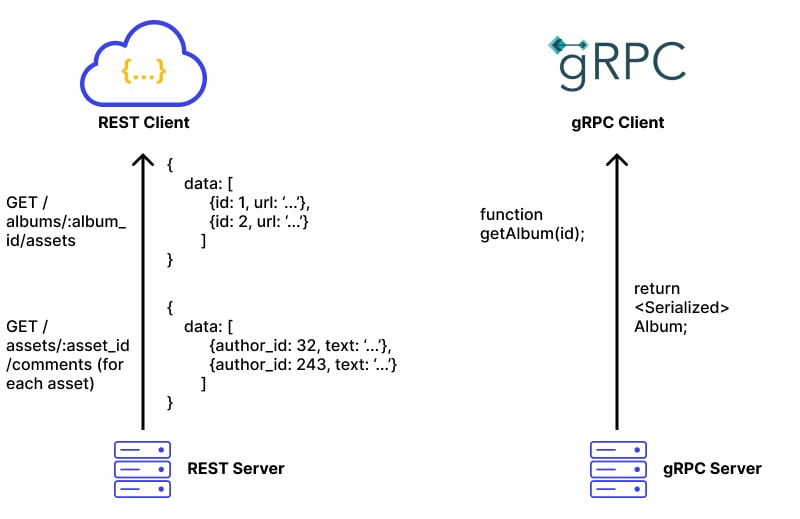
b. Documentation and presentation materials.

The development of a gRPC Microservice API for a product catalog aims to provide customers with a reliable and high-performance browsing experience. By showcasing the functionality and interaction of gRPC technology with the internet, this project will serve as a valuable demonstration of its capabilities. We are confident that this endeavor will contribute to the advancement of API development and promote the adoption of gRPC technology.

Performance: gRPC outperforms REST in terms of speed and efficiency. It achieves this by using a binary protocol (Protocol Buffers) for payload serialization instead of the text-based formats (such as JSON or XML) used in REST. Binary serialization results in smaller message sizes, reducing bandwidth usage and improving response times. Additionally, gRPC employs HTTP/2 as the underlying protocol, enabling multiplexing, header compression, and server push, further enhancing performance.

Strongly Typed Contracts: gRPC utilizes Protocol Buffers (protobuf) as the interface definition language (IDL) for defining the contract between the client and server. Protobuf allows for the definition of strongly typed messages and services, enabling better type validation and reducing the chances of errors during communication. In contrast, REST APIs typically rely on unstructured data formats like JSON, where data types are loosely defined.

Code Generation: One of the significant advantages of gRPC is its code generation capability. The protobuf-based contracts can be used to generate client and server code in various programming languages, providing a strongly typed API and eliminating the need for manual serialization/deserialization of data. This automation improves developer productivity, reduces human error, and ensures consistency between client and server implementations.



**Objectives:** To show how basic gRPC microservice works

To show how clean code for microservices looks like

To investigate how contract really works

**Key features:**

Registration of user to interact with some function.

Login to take a token to get an access to some features.

Find function is to find a product from database.

**Target Audience:**

None. Because I’m just showing basic interaction between three microservices using grpc

**Competitors:** Every person in my class

**Technology Stack**:

GoLang

PostgreSQL

gRPC

RabbitMQ

Docker

In summary, the development of a gRPC Microservice API for a product catalog is relevant due to its alignment with technological advancements, industry demand for efficient APIs, the focus on performance optimization, the opportunity to showcase cutting-edge technology, and the potential for personal and professional growth. This project aims to address these relevant aspects while providing a valuable solution for customers and businesses in need of a robust and user-friendly product catalog API.