Trabalho Prático Individual: Desenvolvimento de Serviços Web Multitecnologia

Objetivo:

Desenvolver um sistema cliente-servidor demonstrando a implementação e integração de múltiplas tecnologias de serviços web, incluindo funcionalidades de exportação e importação de dados nos formatos XML e JSON.

Requisitos:

1. **Servidor (Ubuntu):**
    - Implementar um servidor com as seguintes tecnologias:
        - **SOAP** (com validação XSD).
        - **REST** (com validação JSON Schema e consultas JSONPath).
        - **GraphQL** (queries e mutations).
        - **gRPC** (serviços unários e streaming).
    - Linguagem: Python ou Node.js.
    - Implementação:
        - Único processo ou múltiplos containers Docker.
        - Caso utilize Docker:
            - Criar Dockerfiles para cada serviço.
            - Utilizar docker-compose.yml para orquestração.
            - Compartilhar dados entre serviços via volumes Docker (JSON ou XML).
    - Armazenar dados persistentemente em arquivos JSON ou XML (sem uso de SGBD).
    - Disponibilizar endpoints para operações CRUD num recurso à escolha (ex.: utilizadores, produtos).
    - Disponibilizar funcionalidades específicas para exportação e importação de dados em XML e JSON.
    - Garantir testabilidade de todos os endpoints através do Postman.
2. **Cliente (Python ou JavaScript):**
    - Desenvolver um cliente interagindo com o servidor através de todas as tecnologias mencionadas.
    - - Demonstrar claramente funcionalidades de exportação e importação de dados em XML e JSON.
    - Sugestões de implementação:

        - **Python (desktop):** Utilizar requests, zeep, grpc, etc.
        - **JavaScript (web):** Node.js + Express, com fetch, Apollo Client, etc.
        - Outra configuração, mediante aprovação prévia do professor.

3. **Armazenamento e Validação de Dados:**
    - SOAP: XML validado por XSD.
    - REST e GraphQL: JSON validado por JSON Schema e consultas via JSONPath.
    - gRPC: Validação diretamente no código servidor.
4. **Tema:**
    - Livre escolha (ex.: sistema de gestão de tarefas, catálogo de produtos, etc).
5. **Entrega:**
    - Criar um repositório GitHub com:
        - Código fonte do servidor e cliente (bem estruturado e documentado).
        - Dockerfiles e docker-compose.yml (quando aplicável).
        - Documentação:
            - Descrição detalhada dos endpoints/serviços.
            - README.md com instruções claras para execução, exemplos de chamadas (Postman) e esquemas de validação.
            - Vídeo de demonstração (até 8 minutos), para todos os alunos.
    - Estrutura sugerida do repositório:
        - /servidor
        - /cliente
        - /documentacao
        - docker-compose.yml (quando aplicável)
    - **Acesso ao Repositório GitHub:**
        - Adicionar o professor como **colaborador com permissões de leitura** no repositório GitHub assim que possível, para acompanhamento do progresso através dos commits.
        - Realizar commits frequentes com mensagens claras e informativas.
6. **Apresentação:**
    - Demonstração presencial (8 minutos) para 15 alunos selecionados aleatoriamente.

    - Restantes alunos deverão submeter o vídeo de demonstração no repositório.

7. **Avaliação:**
    - **Funcionalidade (60%)**: Implementação correta e completa das tecnologias.
    - **Organização (30%)**: Estrutura e documentação clara e eficiente do código.
    - **Apresentação (10%)**: Clareza, objetividade e qualidade da demonstração.

Observações Adicionais:

- WebSockets, armazenamento, segurança em APIs e outros temas da UC serão abordados no trabalho em grupo, a realizar.
- Escolher apenas um formato principal de dados persistentes (JSON ou XML), garantindo capacidade de conversão entre formatos (por exemplo, para fins de exportação e importação).
- Enviar o link do repositório GitHub por email o mais cedo possível.
- O trabalho deverá estar concluído e depositado até ao final do dia 17 de abril.
- As apresentações presenciais serão realizadas no dia 22 de abril.