Comércio Eletrônico
Projeto de Programação – Uso de WebServices em Sistemas de Comércio Eletrônico

Informações Gerais

- **Data de Entrega**: 16/08/2020.
- **Pontuação**: 20 pontos.
- O projeto pode ser realizado individualmente ou em duplas.
- Os códigos desenvolvidos devem ser enviados para o e-mail filipe.mutz@ifes.edu.br até a data limite. O e-mail conter os nomes dos integrantes do grupo e instruções de como executar os programas.
- O projeto pode ser implementado em qualquer linguagem. Devem ser usados bibliotecas e frameworks para facilitar o desenvolvimento.
- O aluno pode optar por criar um mini banco de dados contendo tabelas para armazenar produtos e compras ao invés de usar o banco de dados do framework de e-commerce. As pontuações das partes 1 e 2 cairão pela metade e a parte 4 não será pontuada.

Descrição do Projeto

Parte 1 [12 pontos]. O projeto consiste em prover uma interface RESTful parar o sistema de comércio eletrônico usado no projeto anterior. Devem ser criados os seguintes webservices:

- Adição, leitura, remoção e atualização de produtos.
  - **[2 pontos]** Ao realizar a operação de **GET** em http://localhost/webservice/produto sem argumentos deve ser retornada a lista de produtos.
  - **[2 pontos]** Ao realizar a operação de **GET** em http://localhost/webservice/produto passando como argumento o ID de um produto, devem ser retornadas as informações do produto. Os campos que devem ser retornados são o ID, o nome, a descrição, a quantidade em estoque, e o preço.
  - **[2 pontos]** A operação de **POST** em http://localhost/webservice/produto deve permitir o cadastro de um produto. As informações que devem ser enviadas são o nome, descrição, quantidade em estoque, valor, e link para a imagem na internet. Esta operação deve retornar o link do produto no formato esperado pela operação de GET (ex.: http://localhost/webservice/produto/310 se o produto inserido no banco de dados possui identificador 310).
  - **[2 pontos]** A operação de **DELETE** em http://localhost/webservice/produto passando como argumento o identificador de um produto deve remover o produto da base de dados.
  - **[2 pontos]** A operação de **PUT** em http://localhost/webservice/produto deve permitir a atualização dos dados de um produto. Devem ser enviados como argumentos o identificador do produto e os dados usados na inserção de produtos.

- Cálculo de estatísticas sobre as vendas.
  - **[2 pontos]** Ao realizar a operação de **GET** em http://localhost/webservice/stats devem ser retornados o número total de produtos vendidos, o valor total dos produtos vendidos e o valor médio dos produtos vendidos.

Parte 2 [2 pontos]. Os webservices devem retornar mensagens de erro nos seguintes casos:

- Na inserção e atualização, se o valor ou a quantidade forem negativos.

- Em qualquer operação que receba um identificador, se o identificador não existir na base de dados.

Parte 3 [3 pontos]. Para testar os webservices, deve ser criado um programa que permita fazer requisições para os endereços. O programa deve apresentar um menu com as seguintes opções:

1. **Listar produtos**. O primeiro webservice deve ser usado para recuperar a lista de produtos e exibi-la na tela com formatação adequada ao terminal (não exibir o JSON ou XML).
2. **Listar produto**. Deve ser solicitado o ID de um produto para o usuário. Em seguida, o segundo webservice deve ser usado para recuperar as informações do produto e exibi-las na tela.
3. **Inserir produto**. As informações do produto devem ser solicitadas ao usuário. Em seguida, o terceiro webservice deve ser usado para inserir os produtos na base de dados. O identificador do produto deve se exibido na tela. Ao entrar na loja virtual, o produto deve ser acessível.
4. **Atualizar produto**. O identificador e as informações do produto devem ser solicitados ao usuário. O quarto webservice deve ser usado para atualizar as informações na base de dados.
5. **Remover produto**. O identificador do produto deve ser solicitado ao usuário. O último webservice de produtos deve ser usado para remover o produto da base de dados.
6. **Obter estatísticas**. Esta opção deve usar o último webservice para recuperar as estatísticas de vendas e exibir estas informações na tela.

Se as requisições retornarem erros, a mensagem deve ser exibida na tela.

Parte 4 [3 pontos]. Descreva as tabelas do banco de dados e as queries usadas para implementar cada webservice.

Dicas

Tutoriais de como criar webservices RESTful em Python usando diferentes framewoks e bibliotecas:

- https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask
- https://www.codementor.io/@sagaragarwal94/building-a-basic-restful-api-in-python-58k02xsiq
- https://towardsdatascience.com/build-your-own-python-restful-web-service-840ed7766832

Tutoriais de como criar webservices RESTful usando Node.js e diferentes framewoks e bibliotecas:

- https://medium.com/xp-inc/https-medium-com-tiago-jlima-developer-criando-uma-api-restful-com-nodejs-e-express-9cc1a2c9d4d8
- https://medium.com/mackmobile/criando-um-web-service-restful-usando-node-js-7c00d8f16a4a

Documentação da biblioteca requests de Python que descreve como fazer requisições get, post, put e delete para o webservice (essas instruções são importantes para ajudar na solução da parte 3): https://requests.readthedocs.io/pt_BR/latest/user/quickstart.html.