Exercícios de Revisão Bimestral 1

Prof. Luiz Gustavo D. de O. Véras
Desenvolvimento Web Back-end (BRADWBK)
Tecnologia em Análise e Desenvolvimento de Sistemas

Questão 1.

Leia as afirmações a seguir sobre as áreas de front-end e back-end no desenvolvimento de software:

I. O back-end é frequentemente chamado de “lado do servidor” e o front-end de “lado do cliente”.
II. O profissional de back-end é responsável por cuidar exclusivamente do design e da experiência do usuário.
III. O back-end processa solicitações do front-end, interage com bancos de dados e APIs, e retorna as respostas.
IV. As tecnologias de front-end costumam evoluir e mudar mais rapidamente do que as de back-end.
V. O front-end se preocupa principalmente com a segurança, a estabilidade e a performance do sistema.

Com base nas afirmações acima, assinale a alternativa CORRETA:

A) Apenas as afirmações I, III e IV são verdadeiras.
B) Apenas as afirmações II e V são verdadeiras.
C) Apenas as afirmações I, II, IV e V são verdadeiras.
D) Todas as afirmações são verdadeiras.
E) Apenas as afirmações I, III, IV e V são verdadeiras.

Questão 2.

Quando se desenvolve uma interface de serviço, denominada API (*Aplication Programming Interface*), ocultam-se os detalhes de implementação de seus usuários e clientes. Entretanto, algumas características ainda são percepitíveis e influenciam diretamente na decisão de se utilizar o serviço da API em um sistema.

Dentro desse contexto, descreva a definição correta dos conceitos relacionados abaixo:

Confiabilidade:

Escalabilidade

Manutenabilidade

Questão 3.

As APIs podem ser organizadas em diferentes paradigmas, dependendo da forma como a comunicação entre o cliente e o servidor é estabelecida. O primeiro paradigma pode ser denominado requisição-resposta, dentre os quais podemos citar as APIs do tipo RPC (Remote-Procedure Call), REST e GraphQL como exemplos desse grupo. No segundo paradigma temos as APIs baseadas em eventos, representadas por APIS do tipo

WebSockets, HTTP Streaming e Webhooks. Discorra sobre as características gerais entre as tecnologias de APIs em cada paradigma, caracterizando-os as semelhança gerais em um paradigma e apresentando as diferenças gerais entres esses paradgimas de APIs (baseadas em requisição-resposta e baseadas em eventos.)

Resposta

Questão 4.

Existem diversos padrões de API para a Web, cada qual com suas características e restrições específicas. Dentre as bastante utilizadas, temos as APIs que seguem a arquitetura REST (Representational State Transfer) e a GraphQL. Analise cada uma das afirmações a seguir e indique a qual tipo de API ela está associada: REST ou GraphQL.

1. Introduz uma linguagem de consulta para APIs:

- [ ] REST
- [ ] GraphQL

2. Utiliza endpoints distintos para cada recurso (ex: `/usuarios`, `/produtos`):

- [ ] REST
- [ ] GraphQL

3. Permite que o cliente defina exatamente quais campos deseja receber na resposta:

- [ ] REST
- [ ] GraphQL

4. Segue os princípios do protocolo HTTP, utilizando métodos como GET, POST, PUT e DELETE:

- [ ] REST
- [ ] GraphQL

5. Reduz o número de requisições ao servidor ao permitir consultas aninhadas em uma única chamada:

- [ ] REST
- [ ] GraphQL

6. Pode resultar em *overfetching* (dados demais por requisição) ou *underfetching* (poucos dados por requisição), dependendo da estrutura do endpoint:

- [ ] REST
- [ ] GraphQL

7. Toda a comunicação é feita por meio de um único endpoint:

- [ ] REST
- [ ] GraphQL

8. É mais adequada quando se deseja seguir a semântica padrão do protocolo HTTP:

- [ ] REST
- [ ] GraphQL

9. As respostas seguem um formato definido pelo servidor, e não pelo cliente:

- [ ] REST
- [ ] GraphQL

Questão 5.

Uma escola está desenvolvendo um sistema online para gerenciar seus alunos e turmas. O sistema deve permitir:

- Cadastrar novos alunos e novas turmas;
- Listar todos os alunos ou buscar apenas os alunos de uma determinada turma;
- Atualizar os dados de um aluno (como nome, e-mail ou turma);
- Remover um aluno do sistema.

Com base nesse cenário, descreva como você modelaria uma API RESTful que atenda a esses requisitos. Sua resposta deve incluir:

a. Os principais endpoints da API (com caminhos da URL);

b. O método HTTP adequado para cada operação (GET, POST, PUT, DELETE);

c. Exemplos de uso de parâmetros de rota (*path params*) e de *query strings*;

d. Uma explicação breve de como cada operação funcionaria.

Resposta

Questão 6.

O protocolo HTTP (Hypertext Transfer Protocol) é um protocolo de comunicação, situado na camada de aplicação, segundo o modelo OSI. Apesar de ser um protocolo simples, do tipo solicitação-resposta que roda sobre TCP, muito utilizado para as páginas da Internet, tem sido cada vez mais utilizado para comunicação entre máquinas. Em sua definição está, entre outras regras, os códigos de status da resposta a uma requisição, também conhecidos como códigos de status HTTP.

Com relação aos códigos de status de resposta do protocolo HTTP, descreva a finalidade dos subgrupos de códigos de status HTTP. Dê pelo menos dois exemplos de status padrão.

Códigos iniciados com 1 (1xx):

Resposta

Códigos iniciados com 2 (2xx):

Resposta

Códigos iniciados com 3 (3xx):

Resposta

Códigos iniciados com 4 (4xx):

Resposta

Códigos iniciados com 5 (5xx):

Resposta

Questão 7.

No contexto de APIs RESTful, qual código de status HTTP indica que uma requisição foi bem-sucedida e resultou na criação de um novo recurso?

A) 200 OK
B) 201 Created
C) 400 Bad Request
D) 404 Not Found
E) 500 Internal Server Error

Questão 8.

Uma API REST foi desenvolvida utilizando Spring Boot para fornecer informações sobre dispositivos que acessam o sistema. O endpoint a seguir retorna dados personalizados com base em parâmetros fornecidos pela requisição HTTP.

Analise o trecho do código abaixo:

```java
@RestController
@RequestMapping("/api")
public class InfoController {

    @GetMapping("/info/{device}")
    public ResponseEntity<DeviceData> getDeviceInfo(
            @PathVariable("device") String device,
            @RequestParam(name = "type", required = false) String type,
            @RequestHeader("User-Agent") String userAgent) {

        if (userAgent == null || userAgent.isBlank()) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                    .body(null);
        }

        DeviceData data = new DeviceData();
        data.setDevice(device);
        data.setType(type != null ? type : "unknown");
        data.setUserAgent(userAgent);

        return ResponseEntity.ok()
                .header("Cache-Control", "no-cache")
                .body(data);
    }
}
```

Qual será o status de resposta quando forem feitas as seguintes requisições? Justifique sucintamente cada resposta.

I.

```http
GET /api/info/laptop?type=personal HTTP/1.1
User-Agent: Mozilla/5.0
```

Resposta: _____
Justificativa:_____

II.

```
GET /api/info/laptop HTTP/1.1
User-Agent:
```

Resposta: _____
Justificativa:_____

III.

```
GET /api/info/laptop HTTP/1.1
```

Resposta: _____
Justificativa:_____

IV.

```
POST /api/info/laptop HTTP/1.1
User-Agent: Mozilla/5.0
```

Resposta: _____
Justificativa:_____

V.

```
GET /api/infos/laptop HTTP/1.1
User-Agent: Mozilla/5.0
```

Resposta: _____
Justificativa:_____

Questão 9.

GraphQL (Graph Query Language) é uma linguagem muito útil para busca de dados.

Sobre a sintaxe da linguagem GraphQL, assinale a afirmativa correta.

A) Ela possui um esquema de tipos escalares e enumerações, sem suporte a tipos abstratos.
B) A indentação por meio de espaços em branco é a maneira de agrupar os campos dos

objetos definidos para consulta.
C) Ela define um conjunto de operações para consultar dados, mas não oferece meios para modificá-los no lado do servidor.
D) As consultas são estruturadas hierarquicamente e têm o mesmo formato dos dados que elas retornam.
E) O formato XML deve ser usado para serialização das consultas e respostas.

Questão 10.

O GraphQL e o REST são dois padrões para APIs Web muito utilizandos para o desenvolvimento de sistemas. Entretanto, eles apresentam diferenças fundamentais na forma de interação com as aplicações clientes e na solicitação de representação de dados e consultas. Preencha a tabela abaixo, elencando pelos menos duas vantagens e duas desvantagens de se utilizar o GraphQL em comparação à uma API Restful.

Vantagem do GraphQL sobre REST	Desvantagem do GraphQL ao REST

Questão 11.

Considere a seguinte **consulta GraphQL** feita a uma API de um sistema de uma universiade online:

```
query {
  course(id: "CS101") {
    title
    instructor {
      name
    }
    lessons {
      title
      duration
    }
  }
}
```

Com base nessa consulta, qual das seguintes respostas está corretamente formatada e de acordo com o esperado para essa consulta?

A)

```
{
  "data": {
    "course": {
      "title": "Lógica de Programação",
      "instructor": {
        "name": "Prof. Ana"
      },
      "lessons": [
        {
          "title": "Introdução a Variáveis",
          "duration": 40
        },
        {
          "title": "Operadores Lógicos",
          "duration": 50
        }
      ]
    }
  }
}
```

B)

```
{
  "data": {
    "course": {
      "titulo": "Lógica de Programação",
      "instructor": {
        "nome": "Prof. Ana"
      },
      "aulas": [
        {
          "titulo": "Introdução a Variáveis",
          "duracao": 40
        }
      ]
    }
  }
}
```

C)

```
{
  "course": {
    "title": "Lógica de Programação",
    "instructor": {
      "name": "Prof. Ana"
    },
    "lessons": {
      "title": "Introdução a Variáveis",
      "duration": 40
    }
  }
}
```

D)

```
{
  "data": {
    "course": {
      "title": "Lógica de Programação",
      "instructor": "Prof. Ana",
      "lessons": [
        {
          "title": "Introdução a Variáveis",
          "duration": 40
        }
      ]
    }
  }
}
```

E)

```
{
  "data": {
    "course": {
      "title": "Lógica de Programação",
      "instructor": {
        "name": "Prof. Ana"
      },
      "lessons":
        {
          "title": "Introdução a Variáveis",
          "duration": "40 minutos"
        }
    }
  }
}
```

Questão 12.

Você está desenvolvendo uma API GraphQL para uma plataforma de leitura de livros digitais. A entidade principal é **Livro**, que possui os seguintes atributos:

- `id`: identificador único do livro (tipo: ID)
- `titulo`: título do livro (tipo: String)
- `autor`: nome do autor (tipo: String)
- `paginas`: número de páginas (tipo: Int)

A seguir estão as consultas aplicadas numa requisição ao servidor GraphQL.

```
query {
  livro(id: "1") {
    titulo
    autor
  }
}
```

```
mutation AdicionarLivro($titulo: String!, $autor: String!, $paginas: Int!) {
  adicionarLivro(titulo: $titulo, autor: $autor, paginas: $paginas) {
    id
    titulo
  }
}
```

Com base na descrição da entidade, na **consulta** e na **mutação** acima, **escreva o schema GraphQL completo** que inclui:

- O tipo `Livro`
- O tipo `Query` com o campo `livro`
- O tipo `Mutation` com o campo `adicionarLivro`

Questão 13.

Implemente a seguinte API REST em um projeto Spring MVC.

Verbo	Rota	Corpo Requisição (JSON)	Corpo Resposta (JSON)	Status Sucesso	Status Erro	Descrição
GET	`/produtos/{id}`	Não possui corpo de requisição.	`{ "id": 1, "nome": "Smartphone X", "marca": "TechCorp", "preco": 999.99, "estoque": 50 }`	`200 OK`	`404 Not Found`	Retorna os dados de um produto específico com base no ID.
GET	`/produtos`	Não possui corpo de requisição.	`[ { "id": 1, "nome": "Smartphone X", "marca": "TechCorp", "preco": 999.99, "estoque": 50 }, { "id": 2, "nome": "Fone de Ouvido Pro", "marca": "AudioPlus", "preco": 199.99, "estoque": 120 } ]`	`200 OK`	`500 Internal Server Error`	Retorna uma lista de todos os produtos disponíveis.
POST	`/produtos`	`{ "nome": "Smartwatch Y", "marca": "FitTech", "preco": 349.99, "estoque": 100 }`	`{ "id": 3, "nome": "Smartwatch Y", "marca": "FitTech", "preco": 349.99, "estoque": 100 }`	`201 Created`	`400 Bad Request`	Cria um novo produto no sistema.
PUT	`/produtos/{id}`	`{ "nome": "Smartphone X Pro", "marca": "TechCorp", "preco": 1199.99, "estoque": 45 }`	`{ "id": 1, "nome": "Smartphone X Pro", "marca": "TechCorp", "preco": 1199.99, "estoque": 45 }`	`200 OK`	`400 Bad Request`, `404 Not Found`	Atualiza os dados de um produto existente.
DELETE	`/produtos/{id}`	Não possui corpo de requisição.	`{ "message": "Produto removido com sucesso" }`	`200 OK`	`404 Not Found`	Remove um produto específico do sistema.