QUESTÃO 1

O padrão arquitetural Model-View-Controller (MVC) divide uma aplicação web em três componentes principais, permitindo a segregação de responsabilidades e facilitando a manutenibilidade do código. Baseando-se no texto abaixo, analise as afirmações e discuta sua veracidade.

"O padrão MVC é uma abordagem arquitetural que divide uma aplicação em três componentes interconectados. O 'Model' é responsável pela lógica de negócio e pelo acesso aos dados, mantendo os dados e o estado da aplicação. O 'View' é responsável pela apresentação e pela interface com o usuário. O 'Controller' atua como um intermediário, recebendo as entradas do usuário através da 'View', processando-as com a ajuda do 'Model' e, em seguida, exibindo os resultados apropriados de volta à 'View'."

Afirmativas:

I. No MVC, a 'View' é responsável pelo acesso e manutenção dos dados da aplicação.

II. O 'Controller' é o componente que faz a ponte entre o 'Model' e a 'View', garantindo que a lógica de negócio e a apresentação sejam separadas.

III. O 'Model', no padrão MVC, atua principalmente como uma interface gráfica para o usuário.

IV. Uma das vantagens do padrão MVC é a divisão clara de responsabilidades entre os componentes, o que facilita a manutenção e escalabilidade da aplicação.

Instruções: Em sua resposta, discuta a veracidade de cada uma das afirmativas, justificando sua análise com base no texto fornecido.

QUESTÃO 2

As APIs RESTful têm desempenhado um papel fundamental no desenvolvimento web moderno, permitindo a comunicação entre cliente e servidor de forma eficiente e padronizada. Baseando-se no texto abaixo, analise as afirmações e discuta sua veracidade.

"APIs RESTful são uma abordagem para desenvolvimento de serviços web que adota os princípios da arquitetura REST. Elas permitem a comunicação entre sistemas cliente-servidor utilizando padrões da web, como o protocolo HTTP e formatos de mensagem JSON ou XML. No modelo cliente-servidor, o cliente é responsável por solicitar recursos ou serviços, enquanto o servidor processa estas solicitações e retorna a resposta adequada. APIs RESTful usam métodos HTTP como GET, POST, PUT e DELETE para realizar operações CRUD (Create, Read, Update, Delete) em recursos."

Afirmativas:

I. APIs RESTful são exclusivas para comunicação usando o protocolo HTTP.

II. No modelo cliente-servidor, o servidor é o principal responsável por iniciar requisições e solicitar dados ao cliente.

III. O formato JSON é o único formato de mensagem permitido em APIs RESTful.

IV. APIs RESTful usam métodos HTTP específicos para representar operações comuns em recursos, facilitando a interpretação e integração entre sistemas diferentes.

Instruções: Em sua resposta, discuta a veracidade de cada uma das afirmativas, justificando sua análise com base no texto fornecido.

QUESTÃO 3

A organização em camadas é uma prática arquitetural essencial para separar as responsabilidades dentro de aplicações web robustas, especialmente ao trabalhar com APIs RESTful. Com base no texto a seguir, analise as afirmações e discuta sua veracidade.

Afirmativas:

I. A Camada Web é encarregada da lógica de negócios e também da persistência de dados.

II. A principal função da Camada Service é servir como um intermediário, tratando da lógica de negócios, sem se preocupar diretamente com detalhes de persistência ou protocolos HTTP.

III. A Camada Repository é a responsável por lidar com os detalhes específicos da conexão e operações em bancos de dados, como inserções, atualizações e consultas.

IV. É na Camada Repository que as requisições HTTP são inicialmente recebidas e tratadas antes de serem passadas para outras camadas.

Instruções: Em sua resposta, discuta a veracidade de cada uma das afirmativas, justificando sua análise com base no texto fornecido.

QUESTÃO 4

Dentro de uma API RESTful, a estruturação em camadas oferece uma organização clara e uma divisão de responsabilidades. São identificadas comumente as camadas: **Camada Web (Controladores)**, **Camada Service** e **Camada Repository**.

Com base nesse conhecimento:

1. Qual camada é a primeira a interagir diretamente com as requisições HTTP vindas de um cliente?
2. Em que camada se situa predominantemente a lógica de negócios da aplicação?
3. Qual camada é responsável por gerenciar as operações CRUD e se conectar diretamente com o banco de dados?
4. Como a **Camada Service** interage com as outras duas camadas mencionadas? Descreva brevemente o fluxo de uma requisição através destas camadas.

Instruções: Em sua resposta, forneça respostas claras e concisas para cada uma das perguntas, fundamentando-as com base em seus conhecimentos sobre a arquitetura de APIs RESTful.

QUESTÃO 5

Considere a seguinte classe `UserRepository` que possui métodos para manipulação de usuários em um banco de dados:

```python
from flask import abort, make_response, jsonify
from domain.user import User
from app import db
from sqlalchemy import exc

class UserRepository:

    @staticmethod
    def add_user(user):
        try:
            db.session.add(user)
            db.session.commit()
            return user
        except exc.IntegrityError:
           abort(make_response( jsonify({ "message" : "Erro ao criar usuario"}),400))

    @staticmethod
    def get_all_users():
        return User.query.all()
```

Até o momento, existem métodos para adicionar um novo usuário (`add_user`) e para buscar todos os usuários (`get_all_users`).

1. Implemente um método que atualize as informações de um usuário existente. Para isso, considere que o usuário tem um identificador único (`id`) e que o método deve receber, além do `id`, os novos dados do usuário para atualização.

2. Implemente um método que delete um usuário específico com base no seu identificador único (`id`).

3. Descreva uma situação em que o método `get_all_users` poderia apresentar problemas de performance e sugira uma possível solução.

Instruções: Em sua resposta, forneça a implementação dos métodos solicitados e fundamente suas decisões de design, especialmente em relação ao tratamento de exceções e à eficiência das operações.

QUESTÃO 6

Dado o serviço de gerenciamento de usuários abaixo:

```python
from domain.user import User
from dto.user_dto import UserDTO
from repository.user_repository import UserRepository

class UserService:

    @staticmethod
    def get_all_users():
         users =  UserRepository.get_all_users()
         if not users:
             return []
         return [UserDTO.from_model(user) for user in users]


    @staticmethod
    def add_user(user_dto):
        user = User(name=user_dto['name'], email=user_dto['email'])
        if not user:
            return
        return UserDTO.from_model(UserRepository.add_user(user))
```

Até o momento, a classe `UserService` possui métodos para adicionar um novo usuário (`add_user`) e para obter todos os usuários (`get_all_users`).

Item: Implemente uma regra de negócio no método `add_user` que permite adicionar ao sistema somente usuários com endereço de e-mail que termine com "@gmail.com". Justifique sua implementação e as possíveis implicações desta regra de negócio para a aplicação.

Instruções: Em sua resposta, forneça a implementação do método modificado e discuta suas escolhas de design e implementação, especialmente em relação ao tratamento de exceções e às regras de negócio.