**Questões**

**Objetivo:**

A partir dos conceitos e práticas sobre API apresentados em aula, usando a IDE VisualStudio 2022, crie um projeto ASP.NET Core API no estilo GATEWAY DE PAGAMENTO (por exemplo, CIELO e GETNET).

Durante a prova, todo tipo de recurso para consulta estará disponível.

**Requisitos técnicos:**

1 – **(1 ponto)** Middleware de Log Serilog:

* O middleware Serilog deve ser adicionado ao projeto para gerenciamento de logs.
* O Serilog deve ser configurado para armazenar os logs em um arquivo local, com um limite de retenção de um dia.
* Nas demais partes do projeto, erros capturados pelo bloco catch dos tratamentos de exceções devem ser registrados no log.

2 - Utilização de Classes de Serviço, ViewModels e Entidades:

* O projeto deve ser organizado com classes de serviço, DTOs e entidades para uma melhor estruturação do código. Analise com critério, quais recursos são necessários e como usá-los.
* Para acesso ao banco de dados MySQL, utilize ADO.NET.

3 - **(0.5 ponto)** Injeção de Dependência

* Analise sua implementação do projeto e utilize injeção de dependência sempre que possível para evitar o alto nível de acoplamento de código.

4 - **(0.5 ponto)** Documente a API com o Swagger (TODOS OS ENDPOINTS).

**Requisitos da API:**

Além dos requisitos técnicos, a projeto de API deve implementar os seguintes endpoints (EP):

**(0.5 ponto)** **GET /cartoes/{cartao}/obter-bandeira**

* Este endpoint recebe o número do cartão de crédito e retorna sua bandeira (VISA, MASTERCARD, ELO...) de acordo com a regra de negócio fictícia dada a seguir, que considera os primeiros 4 dígitos do número e o 8º do cartão (BIN):
  + **1111**-XXX-X**1**X-XXX: VISA
  + **2222**-XXX-X**2**X-XXX: MASTERCARD
  + **3333**-XXX-X**3**X-XXX: ELO
* O EP deverá o código HTTP 200 +bandeira do cartão ou HTTP 404 se bandeiras desconhecida.

**(0.5 ponto)** **GET /cartoes/{cartao}/valido:**

* Este endpoint recebe o número do cartão de crédito e deve retornar um valor booleano indicando se o cartão é válido. Isso deve ser feito verificando sua existência e validade do cartão na tabela "CARTAO" do banco de dados.

**(1.5 ponto)** **POST /pagamentos/calcular-parcelas:**

* Este endpoint permite calcular o valor das parcelas de um pagamento com base no valor total da transação, a taxa de juros e o número de parcelas desejado. Para tal, necessário enviar no corpo da requisição HTTP do EP, o Valor Total (decimal), Taxa de Juros (decimal) e Quantidade de Parcelas (int). Para facilitar, a taxa de juros deverá ser aplicada sobre o valor total (Valor Total \* Taxa de Juros).
* O EP deverá retornar uma Lista de Parcelas calculadas. Cada item deve possuir o Número da Parcela e seu Valor, exemplo: [{“parcela”: 1, “valor” 10.00}, {“parcela”: 2, “valor” 20.00}]

**(3 pontos)** **POST /pagamentos:**

* Este EP inicia o processo de pagamento, recebendo os detalhes do pagamento no corpo da requisição HTTP: (valor [decimal], número do cartão de crédito [string], CVV [int] e quantidade de parcelas [int]).
* A situação do pagamento/transação financeiras dentro da implementação deve ser representada por um ENUM com as seguintes entradas: PENDENTE (1), CONFIRMADO (2) e CANCELADO (3).
* Após a VALIDAÇÃO (cartão válido), o pagamento deve ser registrado no banco de dados com a situação "pendente/1".
* O EP deve retornar o código HTTP 201 para indicar que o pagamento foi criado com sucesso e também o ID gerado, caso contrário, o HTTP código para BadRequest.

**(1.5 ponto)** **GET /pagamentos/{id}/situacao:**

* Consulta a situação de um pagamento com base no seu ID e retorna apenas a situação do pagamento.

**(0.5 ponto)** **PUT /pagamentos/{id}/confirmar:**

* Este EP confirma o pagamento do ID informado, trocando a situação para "confirmado/2" (ENUM).
* Transações com situação Cancelada não podem ser confirmadas (implemente a validação necessária).
* O EP deve retornar o código HTTP 200 para indicar que o pagamento foi cancelado com sucesso, caso contrário, o HTTP código para BadRequest.

**(0.5 ponto)** **PUT /pagamentos/{id}/cancelar:**

* Este EP cancela um pagamento, apenas se ainda não tenha sido confirmado. Troca a situação para "cancelado/3" (ENUM).
* O EP deve retornar o código HTTP 200 para indicar que o pagamento foi cancelado com sucesso, caso contrário, o HTTP código para BadRequest.

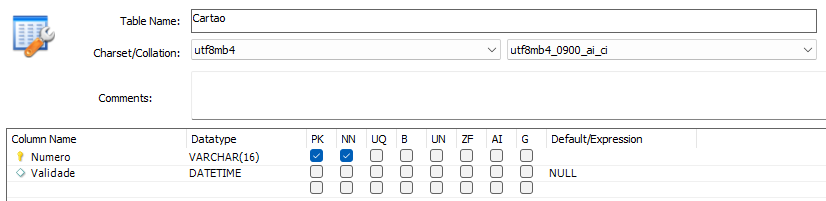
**Requisitos para correção da prova:**

Para a correção da prova, a aplicação deverá estar compilando corretamente, os endpoints devem funcionar conforme descrito nos requisitos e as rotas devem ser exatamente a mesma apresentadas no enunciado. O código-fonte/implementação será avaliado para verificar se os requisitos foram implementados tecnicamente de forma correta. Controllers esperados:

* CartoesController
* PagamentosController

**Tabelas do Banco de Dados:**

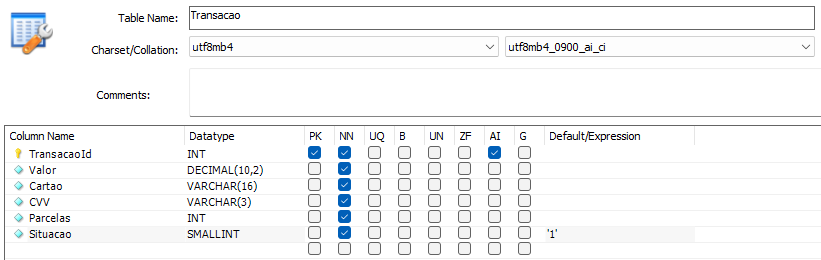
TABELA CARTAO

****

Dados para teste

|  |  |
| --- | --- |
| 1111255516996632 | 2025-01-02 00:00:00 |
| 1111386413243452 | 2025-01-02 00:00:00 |
| 2222322324544323 | 2025-01-02 00:00:00 |
| 2222662825298242 | 2025-01-02 00:00:00 |
| 3333235335483493 | 2025-01-02 00:00:00 |
| 3333932532432452 | 2023-01-02 00:00:00 |
|  |  |

TABELA TRANSACAO

****

SCRIPT PARA GERAR O BANCO DE DADOS

**CREATE TABLE `Cartao` (**

**`Numero` varchar(16) NOT NULL,**

**`Validade` datetime DEFAULT NULL,**

**PRIMARY KEY (`Numero`)**

**)**

**CREATE TABLE `Transacao` (**

**`TransacaoId` int NOT NULL AUTO\_INCREMENT,**

**`Valor` decimal(10,2) NOT NULL,**

**`Cartao` varchar(16) NOT NULL,**

**`CVV` varchar(3) NOT NULL,**

**`Parcelas` int NOT NULL DEFAULT '1',**

**`Situacao` smallint NOT NULL,**

**PRIMARY KEY (`TransacaoId`)**

**)**