Zadanie 1:

Twoim zadaniem jest stworzenie mikroserwisu w C#, który będzie generował dozwolone akcje dla karty użytkownika. Mikroserwis powinien udostępniać punkt końcowy API, który przyjmuje identyfikator użytkownika, numer karty i zwraca dozwolone akcje dla tej karty.

Wymagania:

- Stwórz mikroserwis przy użyciu C# i .NET 8.
- Serwis powinien być w stanie przetworzyć dane wejściowe i zwrócić dane w formacie JSON.
- Zadbaj o obsługę przypadków brzegowych i błędów, aby usługa była niezawodna.
- Jakość kodu, czytelność i możliwość rozszerzenia funkcjonalności są kluczowe
- Aplikacja powinna zostać zacommitowana do repozytorium na GitHubie

Tabela dozwolonych akcji w zależności od typu i statusu karty:

ALLOWED ACTION	CARD KIND			CARD STATUS						
	PREPAID	DEBIT	CREDIT	ORDERED	INACTIVE	ACTIVE	RESTRICTED	BLOCKED	EXPIRED	CLOSED
ACTION1	TAK	TAK	TAK	NIE	NIE	TAK	NIE	NIE	NIE	NIE
ACTION2	TAK	TAK	TAK	NIE	TAK	NIE	NIE	NIE	NIE	NIE
ACTION3	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK
ACTION4	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK
ACTION5	NIE	NIE	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK
ACTION6	TAK	TAK	TAK	TAK - ale jak nie ma pin to NIE	TAK - ale jak nie ma pin to NIE	TAK - ale jak nie ma pin to NIE	NIE	TAK - jeżeli pin nadany	NIE	NIE
ACTION7	TAK	TAK	TAK	TAK - jeżeli brak pin	TAK - jeżeli brak pin	TAK - jeżeli brak pin	NIE	TAK - jeżeli pin nadany	NIE	NIE
ACTION8	TAK	TAK	TAK	TAK	TAK	TAK	NIE	TAK	NIE	NIE
ACTION9	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK	TAK
ACTION10	TAK	TAK	TAK	TAK	TAK	TAK	NIE	NIE	NIE	NIE
ACTION11	TAK	TAK	TAK	NIE	TAK	TAK	NIE	NIE	NIE	NIE
ACTION12	TAK	TAK	TAK	TAK	TAK	TAK	NIE	NIE	NIE	NIE
ACTION13	TAK	TAK	TAK	TAK	TAK	TAK	NIE	NIE	NIE	NIE

Przykład:

1. Dla karty PREPAID w statusie CLOSED aplikacja powinna zwrócić akcje: ACTION3, ACTION4, ACTION9.
2. Dla karty CREDIT w statusie BLOCKED aplikacja powinna zwrócić akcje: ACTION3, ACTION4, ACTION5, ACTION6 (jeżeli pin nadany), ACTION7 (jeżeli pin nadany), ACTION8, ACTION9

Wykorzystaj przykładowy kod aby zrealizować zadanie:

```
public enum CardType
{
   Prepaid,
   Debit,
   Credit
}

public enum CardStatus
{
   Ordered,
   Inactive,
   Active,
   Restricted,
   Blocked,
   Expired,
   Closed
}
```

```
public record CardDetails(string CardNumber, CardType CardType, CardStatus CardStatus, bool IsPinSet);


public class CardService
{
  private readonly Dictionary<string, Dictionary<string, CardDetails>> _userCards = CreateSampleUserCards();

  public async Task<CardDetails?> GetCardDetails(string userId, string cardNumber)
  {
    // At this point, we would typically make an HTTP call to an external service
    // to fetch the data. For this example we use generated sample data.
    await Task.Delay(1000);

    if (!_userCards.TryGetValue(userId, out var cards)
      || !cards.TryGetValue(cardNumber, out var cardDetails))
    {
      return null;
    }

    return cardDetails;
  }

  private static Dictionary<string, Dictionary<string, CardDetails>> CreateSampleUserCards()
  {
    var userCards = new Dictionary<string, Dictionary<string, CardDetails>>();
    for (var i = 1; i <= 3; i++)
    {
      var cards = new Dictionary<string, CardDetails>();
      var cardIndex = 1;
      foreach (CardType cardType in Enum.GetValues(typeof(CardType)))
      {
        foreach (CardStatus cardStatus in Enum.GetValues(typeof(CardStatus)))
        {
          var cardNumber = $"Card{i}{cardIndex}";
          cards.Add(cardNumber,
            new CardDetails(
              CardNumber: cardNumber,
              CardType: cardType,
              CardStatus: cardStatus,
              IsPinSet: cardIndex % 2 == 0));
          cardIndex++;
        }
      }
      var userId = $"User{i}";
      userCards.Add(userId, cards);
    }
    return userCards;
  }
}
```