# .NET MICROSERVICES

7. forduló

A kategória támogatója: DXC Technology

## Ismertető a feladatlaphoz

Az utolsó fordulókhoz érkeztünk, így megosztunk 1-2 fontos információt a továbbiakról:

- a versennyel kapcsolatos észrevételeket december 5-ig tudjátok velünk megosztani a szokásos helyen
- az utolsó fordulóhoz kapcsolódó megoldások november 30-án érhetők el
- a végeredményről tájékoztatás decemberben, részletek hamarosan

Sok sikert az utolsó fordulóhoz!

## 1. feladat 8 pont

A csapatod egy .NET 7 alkalmazást fejleszt. A management szeretné, ha költségcsökkentési okok miatt áttérnétek ARM alapú szerverekre. Mit csinálsz, hogy az alkalmazásod docker image futtatható legyen ARM architektúrán?

### Válasz

- ○ Semmit, figyelmen kívül hagyod a management kérését és továbbra is X64 szervereken futtatod az alkalmazást.
- ○ Semmit, a .NET platform és architektúra független ugyan úgy futtaható.
- ○ Semmit, a docker futtatókörnyezet elfedi az architektúrális különbségeket az alkalmazásod alatt.
- ○ A CI folyamathoz hozzáadsz egy ARM runnert és készítesz egy külön ARM imaget arm subfix taggel.
- ○ QEMU segítségével készítesz egy multi-arch image-et amely így mindkét architektúrán ugyanúgy futtatható.
- ○ A fentiek közül egyik sem.

## 2. feladat 2 pont

Melyik .NET típust használhatjuk a következő kódrészletben az aláhúzások helyén úgy, hogy csak egy dátumot, de ne időpontot tároljunk?

```
______ date = new ______(2023, 8, 19);
```

### Válasz

- DateOnly
- DateTime
- Date
- TimeOnly
- A fentiek közül egyik sem.

## 3. feladat 4 pont

Mi a Saga Pattern fő célja a mikroszolgáltatások tervezésében?

### Válasz

- A folyamatok egy központi, monolitikus alkalmazásban történő kezelésének megkönnyítése.
- Az alkalmazásokban lévő aszinkron hívások számának minimalizálása.
- Összetett tranzakciók kezelése, amelyek több lépésből állnak, és visszavonható vagy kompenzálható műveleteket alkalmaznak.
- Az alkalmazás adatbázis-összeköttetéseinek menedzselése.

## 4. feladat 5 pont

Egy message bus architektúrában mi a message routing szerepe?

### Válasz

- A message routing azt jelenti, hogy az üzeneteket közvetlenül specifikus subscriberekhez küldjük anélkül, hogy bármilyen közvetítőt használnánk.

- A message routing biztosítja, hogy az üzenetek csak egyetlen subscriberhez kerüljenek kézbesítésre, elkerülve mások felesleges terhelését és a redundanciát.
- A message routing meghatározza, hogy az üzenetek hogyan oszlanak meg különböző subscriber-ek között előre meghatározott szabályok alapján.
- A message routing felelős az üzenetek ideiglenes tárolásáért, amíg az subscriber-ek készen állnak azok fogadására.
- A fentiek közül egyik sem.

## 5. feladat 2 pont

Mi a célja a readiness probe-nak Kubernetesben?

### Válasz

- Biztosítja, hogy a microservice megfelelően fogadja a beérkező forgalmat.
- Monitorozza a microservice erőforrás szükségletét.
- Időközönként ellenőrzi a microservice állapotát és biztosítja, hogy az kész a kérések kiszolgálására.
- Monitorozza a microservice hálózati forgalmát és kiszűri a kártékony üzeneteket.

## 6. feladat 3 pont

Hogyan javítja az aszinkron API a kommunikáció teljesítményét és skálázhatóságát?

### Válasz

- Az aszinkron kommunikáció egy RESTful API-ban csökkenti az HTTP metódusok szükségességét, egyszerűsítve ezzel a kérés struktúráját.
- Az aszinkron kommunikáció biztosítja, hogy az összes API kérés sorrendben legyen feldolgozva, elkerülve a versenyhelyzeteket.
- Az aszinkron kommunikáció lehetővé teszi a szerver számára, hogy több kérést egyszerre kezeljen, javítva ezzel a válaszidőt és az erőforrások kihasználtságát.
- Az aszinkron kommunikáció korlátozza a gyorsítótárazási mechanizmusok használatát, biztosítva, hogy az adatokat mindig az eredeti forrásból szedje le.
- A REST API-val nem megoldható aszinkron kommunikáció.

A fentiek közül egyik sem.

## 7. feladat 12 pont

Jelöld be a helyes server kódrészletet!

### Válasz

```
var apiKey = "EXAMPLE_KEY";
var cityName = "Budapest";
using (var httpClient = new HttpClient())
{
    var APIURL = $"http://api.weatherapi.com/v1/current.json?key={apiKey}&q={c
    var response = await httpClient.GetAsync(APIURL);
    return await response.Content.ReadAsStringAsync();
}
```

```
var apiKey = "EXAMPLE_KEY";
var cityName = "Budapest";
using (var httpClient = new HttpClient())
{
    var APIURL = $"http://api.weatherapi.com/v1/current.json?key={apiKey}&q={c
    var response = await httpClient.GetAsync(APIURL);
    return response.Content.ReadAsStringSync();
}
```

```
var apiKey = "EXAMPLE_KEY";
var cityName = "Budapest";
using var httpClient = new HttpClient()
var APIURL = $"http://api.weatherapi.com/v1/current.json?key={apiKey}&q={cityN
var response = await httpClient.GetAsync(APIURL);
return response.Content.ReadAsStringASync();
```

```
var apiKey = "EXAMPLE_KEY";
var cityName = "Budapest";
using(var httpClient = new HttpClient())
{
    var APIURL = $"http://api.weatherapi.com/v1/current.json?key={apiKey}&q={c
```

```
        var response = await httpClient.GetAsync(APIURL);
        return response.Content.ReadAsStringSync();
    }
```

- ( ) A fentiek közül egyik sem.

## 8. feladat 4 pont

Mi lesz a következő kód kimenete?

```
namespace itm23;

internal static class TestApp
{
    public static void Main(string[] args)
    {
        IProcess process = new MyDerivedClass ();
        process.Execute();
        process.HelperMethod();
    }
}

public interface IProcess
{
    void Execute();
    void HelperMethod()
    {
        Console.WriteLine("Helper method in interface.");
    }
}

public class MyBaseClass
{
    protected MyBaseClass()
    {
        Console.WriteLine("Message from base class.");
    }
    public virtual void Execute()
    {
        Console.WriteLine("Execute method in base class.");
    }
}

public class MyDerivedClass  : MyBaseClass, IProcess
{
    public MyDerivedClass()
    {
        Console.WriteLine("Message from derived class.");
    }
    public override void Execute()
    {
        Console.WriteLine("Execute method in derived class.");
    }
}
```

## Válasz

- Message from derived class.
  Message from base class.
  Execute method in derived class.
  Helper method in interface.
- Message from base class.
  Message from derived class.
  Execute method in derived class.
  Helper method in interface.
- Message from derived class.
  Execute method in derived class.
  Helper method in interface.
- A kód nem fordul mert az ős osztály és az interface metódusai nem összeegyeztethetőek.
- Message from derived class.
  Execute method in derived class.
  Futás idejű hiba, "HelperMethod()" kétértelműsége miatt

Megoldások beküldése