.NET MICROSERVICES

3. forduló

A kategória támogatója: DXC Technology

Ismertető a feladatlaphoz

Kérjük, hogy a feladatlap indítása előtt mindenképp olvasd el az alábbi útmutatót:

> Amennyiben olyan kategóriában játszol, ahol van csatolmány, de hibába ütközöl a letöltésnél, ott valószínűleg a vírusirtó korlátoz, annak ideiglenes kikapcsolása megoldhatja a problémát. (Körülbelül minden 3000. letöltésnél fordul ez elő.)

> Helyezéseket a 4. forduló után mutatunk, százalékos formában: adott kategóriában a TOP 20-40-60%-hoz tartozol.
>
> A feltűnően rövid idő alatt megoldott feladatlapok kizárást vonnak maguk után, bármilyen más gyanús esetben fenntartjuk a jogot a forduló érvénytelenítésére!

Jó versenyzést kívánunk!

1. feladat 3 pont

Válaszd ki a helyes állítást a .NET Core microservice-ekre vonatkozóan!

Válasz

- .NET-ben a microservicek mindig HTTP-n keresztül kommunikálnak egymással a clusteren belül.
- Az API Gateway-ek felelősek a microservice-eken belüli kommunikáció felügyeletéért.
- A microservice architechtúra leegyszerűsíti az alkalmazás skálázhatóságát és kitelepítésének folyamatát.
- Asszinkron kommunikációra kizárólag message bus használható.
- A fentiek közül egyik sem.

2. feladat 1 pont

Microservice fejlesztés esetén mi az előnye a "Database per Service" tervezési mintának?

Válasz

- Csökkenti a deployment során a komplexitást. A minta alkalmazásával a microservice-ek ugyanazt az adatbázist használják.
- Elősegíti a konzisztens adat előállítását, mivel az összes microservice ugyan azt az adatbázist írja és olvassa.
- Elősegíti az adat elérést, mivel a microservice-ek olvashatják egymás tábláit.
- Segíti a skálázhatóságot és az izolációt, mivel minden microservice saját adatbázissal rendelkezik.

3. feladat 1 pont

Mi a microservices architektúra egyik fő előnye a szoftverfejlesztésben?

Válasz

- A microservices architektúra szoros összekapcsolást biztosít különböző komponensek között, ami jobb teljesítményhez vezet.
- A microservices architektúra egyszerűsíti a telepítést, mivel az egész alkalmazást egyetlen monolitikus egységként csomagolja.
- A microservices architektúra lehetővé teszi, hogy a csapatok függetlenül fejleszthessék, telepíthessék és skálázhassák a különböző komponenseket.

- [ ] A microservices architektúra megszünteti a szolgáltatások közötti hálózati kommunikáció szükségességét, javítva ezzel a biztonságot.

4. feladat 4 pont

Egy RESTful API-t használsz, szeretnéd az egyes azonosítóval rendelkező felhasználó emailcímét megváltoztatni. A GET /users/1 hívásra a következő választ kaptad.

```
{
    "id": 1,
    "username": "john_doe",
    "email": "john@doe.com"
}
```

A következő hívások közül melyekkel változtathatod meg helyesen az email címet?

Válaszok

- [ ] POST /users/1

```
{
    "id": 1,
    "username": "john_doe",
    "email": "new@doe.com"
}
```

- [ ] POST /users/1

```
{
  "username": "john_doe",
  "email": "new@doe.com"
}
```

- [ ] POST /users/1

```
{
    "email": "new@doe.com"
}
```

- [ ] PATCH /users/1

```
{
    "id": 1,
    "username": "john_doe",
    "email": "new@doe.com"
}
```

- [ ] PUT /users/1

```
{
  "username": "john_doe",
  "email": "new@doe.com"
}
```

- [ ] PUT /users/1

```
{
  "email": "new@doe.com"
}
```

- [ ] PATCH /users/1

```
{
  "username": "john_doe",
  "email": "new@doe.com"
}
```

- [ ] PATCH /users/1

```
{
    "email": "new@doe.com"
}
```

5. feladat 6 pont

A cégnél ahol dolgozol, éppen december van, minden kolléga megkapja az év végi bónuszát. Egyelőre csak a managerek tudják, hogy ki mennyit fog kapni, de az egyik kávézás során megtudtál annyi infót, hogy pontosan 3 ember lesz, aki nálad több bónuszt kap idén, mindenki más kevesebbet. Egy belső hiba miatt már előre bekerültek

az adatbázisba a dolgozók bónuszai, amihez van hozzáférésed is. A Main függvény kikommentelt sorába az alábbi lekérdezések melyikét kell behelyettesítened, hogy megkapd a bónuszod pontos összegét?

```
public class Employee
{
      public int Id { get; set; }
      public decimal Bonus { get; set; }
}

class Program
{
      static List<Employee> GetMockEmployees()
      {
          return new List<Employee>
          {
              new Employee { Id = 1,  Bonus = 60000 },
              new Employee { Id = 2, Bonus = 75000 },
              new Employee { Id = 3,  Bonus = 90000 },
              new Employee { Id = 4,  Bonus = 80000 },
              new Employee { Id = 5, Bonus = 95000 },
              new Employee { Id = 6,  Bonus = 72000 }
          };
      }

      static void Main(string[] args)
      {
          var employees = GetMockEmployees();
          // ** A te queryd **
      }
 }
```

Válasz

- [ ]
```
var yourBonus = employees.OrderByDescending(emp => emp.Bonus).Skip(1).Take(3).Last().Bonus;
```

- [ ]
```
var yourBonus = employees.OrderByDescending(emp => emp.Bonus).Take(3).Last().Bonus;
```

```
var yourBonus = employees.OrderBy(emp => emp.Bonus).Skip(2).Take(2).Last().Bonus;
```

```
var yourBonus = employees.OrderByDescending(emp => emp.Bonus).Take(4).First().Bonus;
```

6. feladat 2 pont

Melyik tervezési minta felel azért, hogy a parancsokat és lekérdezéseket elkülönítetten kezeljük az alkalmazásban?

Válasz

- MVVM
- CQS
- Dependency Injection
- ORM

Megoldások beküldése