Probléma-analízis enterprise rendszerekben (AdNovum)
1. forduló

Ismertető a feladathoz

Egy egészségügyi szervezet betegségek terjedésének megfigyelésével és előrejelzésével foglalkozik. Egy veszélyes vírus visszaszorítására a szervezet vezetői úgy döntenek, hogy speciális informatikai rendszereket is bevetnek. Rövid időn belül számos komponens éles üzembe is kerül, azonban jó néhány probléma csak ezekben a rendkívüli időkben derül ki. A szervezet Téged kér fel a felmerülő tervezési, biztonsági és teljesítmény problémák kivizsgálására. Elvállalod?

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 15 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 1 pont

Az alábbiak közül melyek relációs adatbázisok?

- ( ) Key-value store
- ( ) Wide-column store
- (•) SQL database
- ( ) Document database

Magyarázat a megoldáshoz

A SQL adatbázis relációs, ezért a helyes megoldás.

A másik három különféle NoSQL adatbázis típus, ezért azok nem helyes megoldások.

2. feladat 0 / 2 pont

Az alábbiak közül melyik valós keyword a Java nyelvben?

- ( ) repeat
- ( ) select
- (•) final
- ( ) alias

Magyarázat a megoldáshoz

3. feladat 0 / 4 pont

Melyik célkitűzést segít megvalósítani, ha Docker-t használunk?

- [ ] Egy fizikai (nem virtuális) szerveren futó adatfeldolgozó alkalmazás teljesítményét szeretnénk növelni.
- [x] Sok fejlesztőre lesz szükség az alkalmazások elkészítéséhez, ezért könnyen reprodukálható fejlesztői környezetet szeretnénk kialakítani.
- [ ] Az egészségüggyel kapcsolatos adatok megőrzése elsődleges szempont, megőrzésüket szeretnénk jobban bebiztosítani.
- [x] A kontaktkövető alkalmazás backend rendszerét microservices architektúrával szeretnénk megvalósítani.

Magyarázat a megoldáshoz

Egy fizikai (nem virtuális) szerveren futó adatfeldolgozó alkalmazás teljesítményét szeretnénk növelni: Bár a Docker konténerek kis teljesítménybeli többletet jelentenek, a teljesítményt önmagában ez a technológia nem javítja.

Sok fejlesztőre lesz szükség az alkalmazások elkészítéséhez, ezért könnyen reprodukálható fejlesztői környezetet szeretnénk kialakítani.: A fejlesztéshez szükséges eszközöket nem kell egyesével telepíteni, hanem például docker konténerek formájában is el lehet indítani.

Az egészségüggyel kapcsolatos adatok megőrzése elsődleges szempont, megőrzésüket szeretnénk jobban bebiztosítani.: A Docker konténerekben tárolt adatok a konténer élettartama után elvesznek. Ez kikerülhet külső volume-k használatával, de a Docker nem ad plusz funkciót adatok hatékonyabb megőrzéséhez.

A kontaktkövető alkalmazás backend rendszerét microservices architektúrával szeretnénk megvalósítani.: Alkalmazások saját függőségeikkel egységbe zárva telepíthetőek, kis overheaddel.

4. feladat 0 / 2 pont

Az alábbiak közül melyik adatstruktúrát válasszuk Java-ban, ha azonos elemek többszöri tárolását is szeretnénk támogatni?

- [ ] Stream
- [ ] Set
- [x] List
- [x] tömb

Magyarázat a megoldáshoz

A Stream és a Set megoldások nem helyesek: a Stream nem tárolásra való (doc), a Set pedig duplikált elemek közül csak egy kerül tárolásra.

Vissza a kategóriáimhoz

Probléma-analízis enterprise rendszerekben (AdNovum)
2. forduló

Ismertető a feladathoz

Egy egészségügyi szervezet betegségek terjedésének megfigyelésével és előrejelzésével foglalkozik. Egy veszélyes vírus visszaszorítására a szervezet vezetői úgy döntenek, hogy speciális informatikai rendszereket is bevetnek. Rövid időn belül számos komponens éles üzembe is kerül, azonban jó néhány probléma csak ezekben a rendkívüli időkben derül ki. A szervezet Téged kér fel a felmerülő tervezési, biztonsági és teljesítmény problémák kivizsgálására. Elvállalod?

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 15 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 1 pont

Az alábbiak közül mely fogalmak kapcsolatosak a biztonsági mentésekkel?

- (●) daily / weekly / monthly
- ( ) presentation / application / data
- ( ) physical / data link / network / transport

Magyarázat a megoldáshoz

A helyes megoldás az daily / weekly / monthly, ezek a fogalmak a biztonsági mentés ütemezésére vonatkoznak.

A többi megoldás más területekről származik:

presentation / application / data: ezt szoftver architektúrában, a 3 rétegű tervezésnél szokás alkalmazni

physical / data link / network / transport: ez az OSI hálózati modell alsó 4 rétege

2. feladat 0 / 3 pont

Az alábbiak közül melyik paranccsal listázatóak a futó Docker konténerek?

- ( ) docker images
- ( ) docker run
- (●) docker ps
- ( ) docker containers

Magyarázat a megoldáshoz

A docker containers parancs nem létezik, két megoldás pedig másra való:

docker images: docker imagek listázása

docker run: docker container futtatása

3. feladat 0 / 4 pont

A contact tracking alkalmazásnak a világszerte elérhetőnek kell lennie, ezért arra számítunk, hogy egyidejűleg nagy számú felhasználó is igénybe veheti. Hogy ennek a követelménynek megfeleljünk, úgy döntöttünk, hogy a backend rendszert elosztott módon valósítjuk meg; a rendszert több komponens fogja alkotni és egy komponensből igény szerint többet is el tudunk indítani.

Válaszd ki az alábbi tulajdonságok közül, milyen előnyöket remélhetünk ezáltal!

- [x] a rendszer robosztusabb lesz kiesések, üzemszünetekkel szemben
- [x] a rendszer komponenseit egyszerűbb lesz tesztekkel ellenőrizni
- [ ] a rendszert egészében egyszerűbb lesz end-to-end tesztekkel ellenőrizni
- [ ] a hatékonyság növekedni fog, hiszen kevesebb hálózati forgalomra lesz szükség

Magyarázat a megoldáshoz

a rendszer robosztusabb lesz kiesések, üzemszünetekkel szemben: Igen, hiszen egy kieső komponens helyét át tudja venni egy másik ugyan olyan.

a rendszer komponenseit egyszerűbb lesz tesztekkel ellenőrizni: Igen, mivel a rendszer alapvetően moduláris van tervezve, az egyes modulok mérete várhatóan kisebb lesz, és jól definiált interfészeken keresztül fognak kommunikálni.

a rendszert egészében egyszerűbb lesz end-to-end tesztekkel ellenőrizni: Nem, end-to-end teszt futásához szükséges a teljes rendszer és az őket összekötő infrastruktúra is, amihez számos komponens egyidejű működése szükséges.

a hatékonyság növekedni fog, hiszen kevesebb hálózati forgalomra lesz szükség: Nem, a komponensek együttműködéséhez hálózati kommunikáció szükséges.

4. feladat 0 / 5 pont

Milyen problémát okozhat, ha a rendszerünket Denial of Service támadás éri?

- [x] Elfogyasztja a rendelkezésre álló hálózati kapcsolatokat
- [ ] Eltéríti a felhasználóinkat, és a mi alkalmazásunk helyett egy másik szerverhez csatlakoznak.
- [x] Kimeríti a hálózat sávszélességét
- [ ] Szenzitív adatok szivárognak ki a rendszerünkből
- [x] Túlterheli az alkalmazás erőforrás igényes moduljait

Magyarázat a megoldáshoz

A helyes megoldások arra irányulnak, hogy a támadás a rendszer valamely erőforrását merítik ki.

Az Eltéríti a felhasználóinkat, és a mi alkalmazásunk helyett egy másik szerverhez csatlakoznak. illetve a Túlterheli az alkalmazás erőforrás igényes moduljait válasz ilyen módon nem merül fel egy tipikus Denial of Service támadás során.

Vissza a kategóriáimhoz

Probléma-analízis enterprise rendszerekben (AdNovum)
3. forduló

Ismertető a feladathoz

Egy egészségügyi szervezet betegségek terjedésének megfigyelésével és előrejelzésével foglalkozik. Egy veszélyes vírus visszaszorítására a szervezet vezetői úgy döntenek, hogy speciális informatikai rendszereket is bevetnek. Rövid időn belül számos komponens éles üzembe is kerül, azonban jó néhány probléma csak ezekben a rendkívüli időkben derül ki. A szervezet Téged kér fel a felmerülő tervezési, biztonsági és teljesítmény problémák kivizsgálására. Elvállalod?

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 15 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 1 pont

A rendszer egyik komoponensével felhasználók egészségügyi adatait fogjuk kezelni. Az adatok jól struktúráltak és integritásuk kulcsfontosságú. A rendszert egy webalkalmazással fogjuk megvalósítani, amelyen keresztül az arra jogosult felhasználók léphetnek be és menedzselhetik ezeket az adatokat (tipikus CRUD műveletek, összetett keresés különböző szempontok alapján).

Milyen adatbáziskezelő-rendszert lenne a legcélszerűbb alkalmaznunk ehhez a rendszerhez?

- ○ dokumentum
- ◉ relációs
- ○ key-value
- ○ fájl tároló

Magyarázat a megoldáshoz

Struktúrált adatoknál, ahol a konzisztencia az első számú szempont érdemes NoSQL megoldás helyett relációs adatbázist választani.

A fájl tároló nem nyújt segítséget az adatok strukturált kezelésében, és összetett szempontrendszer szerinti lekérdezésében.

2. feladat 0 / 8 pont

A következő DDL által leírt táblát szeretnénk mappelni JPA-val az InformationBooth osztályra Hibernate segítségével.

```
/* SQL DDL script:
CREATE TABLE info_booth (
    id int,
    name varchar(255),
    country_id int,
    PRIMARY KEY (id)
    FOREIGN KEY (country_id) REFERENCES country(id)
);
*/

public class InformationBooth {
    private Long id;
    private String name;
    private Country country;

    // ... getterek és setterek ...
}
```

A rendszerben már adott egy country adatbázis tábla, amelyet a Country osztályra képeztünk le. Az adott osztályon kívül mást nem módosíthatunk. Milyen annotációk hozzáadására van szükség ehhez?

- ○ Semmire
- ○ Entity, Table, Id, JoinColumn, OneToMany
- ◉ Entity, Table, Id, JoinColumn, ManyToOne
- ○ Entity, Id, JoinColumn, OneToMany
- ○ Table, Id, JoinColumn, ManyToOne

Magyarázat a megoldáshoz

3. feladat 0 / 8 pont

Az alábbiak közül melyek igazak a RESTful HTTP GET műveletre?

- ☑ a paraméterek az URL-ben közlekednek
- ☐ nagy mennyiségű adat továbbítható a HTTP kérés törzsében
- ☐ adatok módosítására szolgál
- ☑ idempotens művelet
- ☑ jól gyorsítótárazható

Magyarázat a megoldáshoz

a paraméterek az URL-ben közlekednek, és nagy mennyiségű adat továbbítható a HTTP kérés törzsében: adatok a GET műveleteknél az URL-ben közlekednek. a HTTP törzsben például POST esetén közlekedik adat

adatok módosítására szolgál: A GET adatok, erőforrások állapotának lekérésére szolgál

idempotens művelet: https://hu.wikipedia.org/wiki/Idempotencia

jól gyorsítótárazható: a kérés az erőforrás aktuális állapotának megfelelő választ ad vissza, ezért ismételt lekérdezésnél is ugyanazt kapnád feltéve hogy nem változott.

Vissza a kategóriáimhoz

Probléma-analízis enterprise rendszerekben (AdNovum)
4. forduló

Ismertető a feladathoz

Egy egészségügyi szervezet betegségek terjedésének megfigyelésével és előrejelzésével foglalkozik. Egy veszélyes vírus visszaszorítására a szervezet vezetői úgy döntenek, hogy speciális informatikai rendszereket is bevetnek. Rövid időn belül számos komponens éles üzembe is kerül, azonban jó néhány probléma csak ezekben a rendkívüli időkben derül ki. A szervezet Téged kér fel a felmerülő tervezési, biztonsági és teljesítmény problémák kivizsgálására. Elvállalod?

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 15 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 1 pont

Az alábbi paranccsal a health/stat Docker imageből készített konténert futtatjuk.

```
docker run --rm -v $(pwd):/work health/stat
```

A program a rendelkezésre álló egészségügyi adatokból készít statisztikát. A folymat sokáig tart, így nem futtathatjuk le többször. Ennek során az alábbi fájlok jönnek létre a konténer fájlrendszerén:
- /work/hello.txt
- /opt/world.txt

Miután a folyamat befejeződött, az alábbi állítások közül melyik igaz?

- ( ) Mindkét fájl elérhető lesz az aktuális könyvtárban.
- (•) A /work/hello.txt elérhető az aktuális könyvtárban, a másik törlésre került.
- ( ) A /opt/world.txt elérhető az aktuális könyvtárban, a másik törlésre került.
- ( ) Egyik fájl sem érhető el az aktuális könyvtárban, de mindkettőhöz hozzáférhetünk a konténer fájlrendszerén.
- ( ) A /work/hello.txt elérhető az aktuális könyvtárban, a másikhoz hozzáférhetünk a konténer fájlrendszerén.
- ( ) A /opt/world.txt elérhető az aktuális könyvtárban, a másikhoz hozzáférhetünk a konténer fájlrendszerén.

Magyarázat a megoldáshoz

A -v kapcsoló után megadott mount miatt a konténer /work könyvtára a gazda rendszer aktuális könyvtárára lesz leképezve, így az ott létrehozott állományok megmaradnak és elérhetőek a gazda rendszeren.

Az --rm kapcsoló miatt a konténerhez tartozó névtelen volumek törlésre kerülnek a futás után, így a többi fájl, köztük a /opt könytár tartalma aművelet után már nem hozzáférhető.

2. feladat 0 / 8 pont

A következő kódrészlet azt az API végpont definíciót és adatbázis lekérdezést mutatja be, amit a rendszer egy publikusan elérhető weboldalán elhelyezett kereső használ. A weboldalon egy szabad szöveges keresőfeltétel alapján jelennek meg a mindenki számára látogatható információs pontok. A rendszer az információs pontok nyilvánosan hozzáférhető adatait kezeli a világ minden tájáról.

```
@RestController
public class SearchController {
    @Autowired
    InformationBoothRepository informationBoothRepository;
    @GetMapping("/search")
    public List<InformationBooth> search(@RequestParam(value = "name") S
        return informationBoothRepository.searchByName(name);
    }
}

public interface InformationBoothRepository extends JpaRepository<Inform
    @Query("SELECT ib FROM InformationBooth ib WHERE ib.name LIKE CONCAT
    List<InformationBooth> searchByName(@Param("name") String name);
}
```

Milyen problémába ütközhetünk ezzel a megoldással?

- [ ] SQL injection
- [x] adatbázis túlterhelés
- [x] magas hálózati forgalmat generál
- [ ] a megoldás nem felel meg a GDPR -nek

Magyarázat a megoldáshoz

a) Nem, a kód paraméterezett JPQL-t használ, ami védelmet nyújt az SQL injection ellen.

b) Igen, ha üres stringet küld be a felhasználó, az egész adatbázis tábla megjelenik az eredményhalmazban.

c) Igen, ha üres stringet küld be a felhasználó, a felhasználónak kiküldjük az összes információs .

d) Nem, a rendszer nem kezel felhasználói adatot.

3. feladat 0 / 4 pont

A következők közül mely intézkedések segítenek egy brutforce támadás megállításában?

- [x] Rate limiting
- [x] Tűzfal
- [ ] Alkalmazás memóriájának limitálása
- [x] Feldolgozandó kérések aszinkron queue-ba helyezése

Magyarázat a megoldáshoz

A memória limitálása csak az alkalmazás korábbi túlterhelődéséhez fog vezetni. Queue-ba való aszinkron feldolgozással mi választjuk meg a feldolgozás ütemét, és ezzel az egyébként kiszolgálhatatlan kéréseket is kiszolgálhatjuk amint fölszabadulnak erőforrások a későbbiekben.

Vissza a kategóriáimhoz

Probléma-analízis enterprise rendszerekben (AdNovum)
5. forduló

Ismertető a feladathoz

Egy egészségügyi szervezet betegségek terjedésének megfigyelésével és előrejelzésével foglalkozik. Egy veszélyes vírus visszaszorítására a szervezet vezetői úgy döntenek, hogy speciális informatikai rendszereket is bevetnek. Rövid időn belül számos komponens éles üzembe is kerül, azonban jó néhány probléma csak ezekben a rendkívüli időkben derül ki. A szervezet Téged kér fel a felmerülő tervezési, biztonsági és teljesítmény problémák kivizsgálására. Elvállalod?

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 15 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 1 pont

A kontaktkövető mobil app érzékeny adatokat küld a szerverenk. Ezért az alkalmazásban certificate pinning-et implementáltunk, ami a következők ellen véd:

- [ ] rosszindulatú user valótlan adatokat tölt fel
- [x] egy rosszhiszemű internetszolgáltató, aki a DNS-t kontrollálja
- [x] szofisztikált támadó, aki egy root CA-t kontrollál
- [ ] szofisztikált támadó, aki képes a felhasználó eszközére malware-t telepíteni

Magyarázat a megoldáshoz

rosszindulatú user valótlan adatokat tölt fel: Nem. Egy kellő tudású felhasználó az eszköz birtokában a legtöbb kliens oldali védelmi megoldást meg tudja kerülni. Továbba a certificate pinning nem akadályozza meg a felhasználót, hogy valótlan adatokat adjon be.

egy rosszhiszemű internetszolgáltató aki a DNS-t kontrollálja: Igen. Mivel a kliens egy konkrét szerver tanúsítványt vár, a DNS spoofolása fel fog tűnni neki.

szofisztikált támadó aki egy root CA-t kontrollálnak: Igen. Pont ez a lényege a certificate pinning-nek, hogy a kliens csak egy bizonyos tausítványt (vagy CA-t) tart megbízhatónak, tehát egy random root CA tanusítványát nem fogja elfogadni.

szofisztikált támadó aki képes a felhasználó eszközére malware-t telepíteni: Nem. Az eszköz feletti teljes irányítás és root jogosultságok megszerzése a kliens oldali alkalmazás szintű védőmechanizmusok kiiktatására ad lehetőséget.

2. feladat 0 / 20 pont

A 8GB memóriájú szerverünkön 10 darab Spring-Boot alapú Java alkalmazás fut docker containerekben azonos erőforrás szükségletekkel. A docker daemont futtató operációs rendszer 800 MB RAM-ot igényel.

Az alábbiak közül mennyi legyen az egyes Java alkalmazások maximum Heap mérete ahhoz hogy optimálisan kihasználjuk a memóriát és az OOM errort is nagy esélyel elkerüljük?

- [ ] 800 MB
- [x] 720 MB
- [x] 710 MB
- [x] 500 MB

Magyarázat a megoldáshoz

800 MB az oprendszernek -> 7200MB marad, osztva 10 = 720MB, ezzel a 800 MB ki van zárva. A Java alkalmazásoknak nem csupán heap memóriára van szüksége hanem metaspace, native space, stack memory-ra is.

3. feladat 0 / 10 pont

Az alábbi osztályok egy egyszerű számológépet valósítanak meg.

```
public class Calculator{
    private Map<String, Operation> operations;

    public Calculator() {
        this.operations = new HashMap<>();
        this.operations.put("+", new Addition());
        this.operations.put("-", new Subtraction());
    }

    public int perform(int a, int b, String operation) {
        return operations.get(operation).perform(a, b);
    }
}

public interface Operation {
    int perform(int a, int b);
    String render(int a, int b);
}

public class Addition implements Operation {
    public int perform(int a, int b) { return a + b; }
    public String render(int a, int b) { return a + "+" + b; }
}

public class Subtraction implements Operation {
    private Addition addition = new Addition();
    public int perform(int a, int b) { return addition.perform(a, (-1) *
    public String render(int a, int b) { throw new UnsupportedOperationE
}
```

Szeretnénk továbbfejleszteni, azonban előtte szeretnénk megérteni, milyen lehetséges tervezési problémák vannak a jelenlegi megvalósítással.

A SOLID objektum-orientált tervezési alapelvek közül melyeket sérti a fenti kód?

- [x] Single responsibility principle
- [x] Open/closed principle
- [x] Liskov substitution principle
- [x] Interface segregation principle
- [x] Dependency inversion principle

Magyarázat a megoldáshoz

- Single responsibility principle: egy művelet több mindent is csinál: számol, és a megjelenítéséért is felelős
- Open/closed principle: újabb művelet hozzáadásánál módosítanunk kell a Calculator osztályt.
- Liskov substitution principle: a kivonás csak típus szinten támogatja a render metódust
- Interface segregation principle: az Operation interface több, nem szorosan összetartozó metódust is tartalmaz (a render nincs használva a fent vázolt használati esetben)
- Dependency inversion principle: a függőségek jelenleg nem befecskendezettek, hanem az osztályokba vannak hard-codeolva.

Vissza a kategóriáimhoz

Probléma-analízis enterprise rendszerekben (AdNovum)
6. forduló

Ismertető a feladathoz

Egy egészségügyi szervezet betegségek terjedésének megfigyelésével és előrejelzésével foglalkozik. Egy veszélyes vírus visszaszorítására a szervezet vezetői úgy döntenek, hogy speciális informatikai rendszereket is bevetnek. Rövid időn belül számos komponens éles üzembe is kerül, azonban jó néhány probléma csak ezekben a rendkívüli időkben derül ki. A szervezet Téged kér fel a felmerülő tervezési, biztonsági és teljesítmény problémák kivizsgálására. Elvállalod?

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 15 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 1 pont

Mi fog történni: Van egy alkalmazásunk amiben a db connection pool mérete 10 és 1 db metódust publikál, ami két egymásba ágyazott tranzakciót indít a meghívásakor (ezzel max 2 db connectiont használva egyidejűleg).

----->metódus eleje--->

----->db kapcsolat 1 eleje--->

----->db kapcsolat 2 eleje--->

----->db kapcsolat 2 vége--->

----->db kapcsolat 1 vége--->

----->metódus vége--->

A metódus futási ideje 1mp és a db connection timeout 5 másodperc. Mi történhet legrosszabb esetben amikor elérjük a 6 valamint később a 10 párhuzamos hívást másodpercenként?

- Semmi, a rendszer elbírja a terhelést.
- 6 kérés működik, viszont 10 kérésnél holtpont alakul ki, amiből nem lehet visszatérni.
- 6 kérés működik, viszont 10 kérésnél holtpont alakul ki, amiből 5 másodperc után visszaáll a rendszer.
- 6 kérésnél erőforrás kiéheztetés alakul ki, amiből 1 másodperc után visszaáll a rendszer, 10 kérésnél holtpont alakul ki, amiből nem lehet visszatérni.
- 6 kérésnél erőforrás kiéheztetés alakul ki, amiből 5 másodperc után visszaáll a rendszer, 10 kérésnél holtpont alakul ki, amiből nem lehet visszatérni.
- (●) 6 kérésnél erőforrás kiéheztetés alakul ki, amiből 1 másodperc után visszaáll a rendszer, 10 kérésnél holtpont alakul ki, amiből 5 másodperc után lehet visszatérni.
- 6 kérésnél erőforrás kiéheztetés alakul ki, amiből 1 másodperc után visszaáll a rendszer, 10 kérésnél holtpont alakul ki, amiből 1 másodperc után lehet visszatérni.
- A rendszerben már 6 kérésnél deadlock alakul ki, amiből 5 másodperc visszatérni.

Magyarázat a megoldáshoz

5 kérésnél 10 kapcsolat fogyhat el maximum. Ebben a pillanatban a 6. hívásnak max 1 másodpercet kell várnia hogy fölszabadoljon egy kapcsolat. 10 hívásnál előfordulhat hogy minden hívás csupán az első kapcsolatot szerzi meg és egymásra várnak a 2. kapcsolat fölszabadulásához ezért holtpont alakul ki ami 5 másodperc után a connection timeoutkor feloldódik.

2. feladat 0 / 6 pont

A rendszer információs pont kereső rendszere különböző hibák következtében gyakran leáll. A rendszer összetett keresési kifejezések alapján végzett számítások alapján jelenít meg találati listát a felhasználóknak. Sajnos a jelenlegi kialakítás mellett akár órák is eltelhetnek, mire a rendszer üzemeltetői észreveszik a problémát újraindítják a rendszert. Szerencsére az alkalmazásnak nem kell sok idő az inicializáláshoz, így újraindítás után szinte azonnal használatba vehetik a felhasználók.

Az alábbiak közül melyekkel érhetjük el, hogy a modul magasabb rendelkezésre állással működjön, és a fenti problémát kiküszöböljük?

- [x] monitorozzuk az infrastruktúra és az alkalmazás működését, probléma esetén automata riasztást küldünk
- [x] a komponenst képessé tesszük automatikus horizontális skálázódásra, és terhelés elosztón keresztül tesszük elérhetővé a többi modul számára
- [x] hiba esetén a felhasználókat áttereljük egy standby szerverre
- [ ] hiba esetén a felhasználókat áttereljük statikus HTML oldalra

Magyarázat a megoldáshoz

A "hiba esetén a felhasználókat áttereljük statikus HTML oldalra" is segít a felhasználói élmény növelésében hiba esetén, viszont egy dinamikus fórum modulnál a statikus HTML oldal nem elég.

3. feladat 0 / 20 pont

Az alkalmazásunk az RFC 7519 szabványnak megfelelő JWT token használ a felhasználók azonosítására. Az rendszer az alábbi kódrészlettel illusztrált módon autentikálja és autorizálja a felhasználókat:

```
Algorithm algo = Algorithm.RSA256(publicKey, privateKey);

String authenticate(String username, String password) {
    User user = userService.getUser(username, password);
    return JWT.create()
            .withIssuer("stay-healthy")
            .withSubject(user.getEmail())
            .sign(algo);
}

String authorize(String token) {
    JWTVerifier jwtVerifier= JWT.require(algo)
            .withIssuer("stay-healthy")
            .build();
    DecodedJWT jwt = jwtVerifier.verify(token);
    String currentUserEmail = jwt.getSubject();
    logger.info("Hello, " + currentUserEmail);
    return currentUserEmail;
}
```

A rendszer felhasználói natív Android és iOS alkalmazások, amelyek a bejelentkezés után a memóriájukban tárolják a felhasználó tokenjét.

Milyen lehetőségeink vannak, ha egy adott felhasználót szeretnénk biztonságosan kijelentkeztetni?

- A rendszert kiegészítjük egy "logout" metódussal, ami egy token vár paraméterként és érvényteleníti azt. Visszatérési értéke az új, érvénytelenített token amit visszaküldünk a felhasználónak.
- (●) Az authenticate metódust módosítjuk, hogy rövid élettartamú tokeneket állítson elő, és kiegészítjük egy "refresh" metódussal, ami egy érvényes token alapján egy frisset ad vissza. A rendszerből való kijelentkezés úgy történik, hogy a felhasználó nem újítja meg a tokenét, és az aktuális tokent törli a saját memóriájából.
- A rendszert kiegészítjük egy "logout" metódussal, ami egy új "algo" objektumot hoz létre egy másik aláírókulccsal. Ha egy felhasználó ki szeretne jelentkezni, meghívja ezt a metódust.
- A kliens törli a tokenjét a saját memóriájából.

Magyarázat a megoldáshoz

A helyes válasz támaszkodik arra, hogy a JWT token nem módosítható és önleíró. A rövid élettartammal és a folyamatos megújítással valósítja meg biztonságosan azt, hogy a nem aktív felhasználói munkamenetek érvénytelenítésre kerüljenek.

A többi megoldás különböző okok miatt nem megfelelő:

A rendszert kiegészítjük egy "logout" metódussal, ami egy token vár paraméterként és érvényteleníti azt. Visszatérési értéke az új, érvénytelenített token amit visszaküldünk a felhasználónak.: A JWT token nem módosítható.

A rendszert kiegészítjük egy "logout" metódussal, ami egy új "algo" objektumot hoz létre egy másik aláírókulccsal. Ha egy felhasználó ki szeretne jelentkezni, meghívja ezt a metódust.: Ezzel a metódussal ha valaki ki szeretne lépni, az összes többi felhasználó tokenjét is érvénytelenítenénk.

A kliens törli a tokenjét a saját memóriájából.: Ettől a token még érvényes marad a szerver oldalon, illetve ha illetéktelenek kezébe jutott, azok korlátlan ideig felhasználhatják, mivel nem állítunk be lejáratot.

Vissza a kategóriáimhoz

Probléma-analízis enterprise rendszerekben (AdNovum)
7. forduló

Ismertető a feladathoz

Egy egészségügyi szervezet betegségek terjedésének megfigyelésével és előrejelzésével foglalkozik. Egy veszélyes vírus visszaszorítására a szervezet vezetői úgy döntenek, hogy speciális informatikai rendszereket is bevetnek. Rövid időn belül számos komponens éles üzembe is kerül, azonban jó néhány probléma csak ezekben a rendkívüli időkben derül ki. A szervezet Téged kér fel a felmerülő tervezési, biztonsági és teljesítmény problémák kivizsgálására. Elvállalod?

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 15 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 1 pont

A kontaktkövető mobil appunk a backend rendszer REST végpontját HTTPS-en keresztül éri el. Tekintve az alkalmazásba épített biztonsági megoldásokat, nem kell aggódnunk, hogy egy esetleges támadó a titkosított üzenetek tartalmát vissza tudná fejteni.

Milyen információkhoz juthat hozzá egy támadó ha a titkosított hálózati kommunikációt figyeli meg?

- [x] A backend szerver domain neve.
- [ ] A kliens által küldött HTTP query paramétereket.
- [ ] A kliens által küldött cookie-k.
- [x] Ha a felhasználó nagy mennyiségű adatot tölt fel.

Magyarázat a megoldáshoz

A backend szerver domain neve: Igen. Az szerver IP címe látható HTTPS kommunikáció közben. A domain név a letöltők esetben egyértelmű az IP címből. Ha az adott szerver több domain nevet szolgál ki HTTPS-en keresztül, egy SNI rekordot küld a handshake fázisban. Továbbá kapcsolat felépítése előtt a kliens DNS lekérései tipikusan nincsenek titkosítva.

A kliens által küldött HTTP query paramétereket.: Nem. Az URL (ami a GET paramétereket tartalmazza) ügymint az üzenettest (POST paraméterek) titkosítva vannak.

A kliens által küldött cookie-k.: Nem. A cookiek a HTTP fejlécben titkosítva vannak.

Ha a felhasználó nagy mennyiségű adatot tölt fel. : Igen. Bár a kommunikáció tartalma titkosítva van, a küldött és fogadott adatok mennyisége megfigyelhető dekódolás nélkül, így a két eset jól különválasztható.

2. feladat 0 / 5 pont

A következők közül melyek okoznak N+1 problémát?

- [ ] orderids = select id from order; select * from order_item where order_id in (orderids);select * from order inner join order_item on order.id = order_item.order_id;
- [ ] select * from order inner join order_item on order.id = order_item.order_id;
- [x] orderids = select id from order;
  for each (orderid in orderids) {
  select * from order_item where order_id = orderId;
  }

Magyarázat a megoldáshoz

N darab lekérést végzünk el a kezdeti lista lekérés mellé: N+1

3. feladat 0 / 20 pont

A következő két osztály egy Spring Boot-ra épülő alkalmazásban található, ami a Spring Data komponenst használja, hogy az általa használt adatokat adatbázisba perzisztálja.

```
@RestController
@Transactional
public class UserController {
    @Autowired private UserService userService;

    @PostMapping("/update")
    public void updateUser(Long userId, String firstName, String lastNam
        userService.updateProfile(userId, firstName, lastName);
        userService.setAvatar(userId, avatar);
    }
}

@Service
@Transactional(propagation = Propagation.REQUIRES_NEW)
public class UserService {

    @Autowired
    private UserRepository userRepository;

    public void updateProfile(Long userId, String firstName, String last
        User user = userRepository.findById(userId).get();
        user.setFirstName(firstName);
        user.setLastName(lastName);
        try {
            calculateTimestamp(userId);
        }
        catch (Exception e) {
            //nothing to do
        }
        userRepository.save(user);
    }

    @Transactional(propagation = Propagation.REQUIRES_NEW)
    private void calculateTimestamp(Long userId) {
        User user = userRepository.findById(userId).get();

        LocalDateTime ldt = LocalDateTime.now();
        String dateStamp = DateTimeFormatter.ofPattern("yyyy-MM-dd", Loc
        user.setLastUpdated(dateStamp);

        // Boom!
        throw new SimulationException("Oh-oh!");
    }

    public void setAvatar(Long userId, String avatar) {
        if (avatar == null || avatar.isEmpty()) {
            throw new IllegalArgumentException("avatar is not specified"
        }
        userRepository.findById(userId).get().setAvatar(avatar);
    }
}
```

A rendszer dependency injectionhoz a Spring alapértelmezett AOP működésére támaszkodva proxy objektumokkal dolgozik.

Tegyük fel, hogy a rendszerünkben van egy felhasználó az alábbi tulajdonságokkal:

- id=1
- firstName=John
- lastName=Doe
- avatar=<egy kép base64 kódolásban>
- lastUpdated=2019-03-01

A rendszer egyik adminisztrátora a következő adatokkal hívja meg az UserController-ben definiált végpontot:

- id=1
- firstName=Jane
- lastName=Doe
- avatar=null

Az alábbi állítások közül mely lesz igaz az 1-es azonosítójú felhasználóra a művelet után?

- [ ] firstName=John
- [x] firstName=Jane
- [x] lastName=Doe
- [x] avatar=<egy kép base64 kódolásban>
- [ ] avatar=null
- [x] lastUpdated=<aktuális dátum yyyy-MM-dd formátumban>
- [ ] lastUpdated=2019-03-01
- [ ] lastUpdated=null

Magyarázat a megoldáshoz

Az updateProfile és a setAvatar külön tranzakcióban fut. Az utóbbi meghiúsul, hiszen paraméternek null-t adtunk át, így az avatar értéke nem változik.

Ettől függetlenül az updateProfile elvégzi firstName és a lastName módosítását hiszen a különálló tranzakcióját nem hiúsítja meg az avatar módosítás sikertelensége.

A lastUpdated metódus noha meg van jelölve, hogy külön tranzakcióban fusson (@Transactional(propagation = Propagation.REQUIRES_NEW)), az annotáció nem érvényesül, mert a hívás az objektumon belülről érkezik, nem pedig kintről, a proxy-n keresztül.

4. feladat 0 / 15 pont

A tervezett szolgáltatással kapcsolatban fontos szempont, hogy ellenálló legyen Denial of Service támadás ellen. Milyen módszerekkel csökkenthetjük az esélyét, hogy egy ilyen támadás szolgáltatáskiesést okozzon?

- [x] statikus fájlok kiszolgálása Content Delivery Network segítségével
- [ ] minden hálózati kommunikációt HTTPS protokollon valósítunk meg
- [x] olyan architektúrát alkalmazunk, ami a terhelés függvényében képes dinamikusan skálázódni
- [x] kártékony forgalom szűrése a tűzfalnál

Magyarázat a megoldáshoz

statikus fájlok kiszolgálása Content Delivery Network segítségével: Igen. A CDN célja statikus fájlok hatékony kiszolgálása, alkalmazása jelentős terheléstől mentesítheti a szolgáltatásunkat.

minden hálózati kommunikációt HTTPS protokollon valósítunk meg: Nem. A HTTPS, bár biztonsági szempontból előnyös, alkalmazása nem segít ebben a helyzetben.

olyan architektúrát alkalmazunk, ami a terhelés függvényében képes dinamikusan skálázódni: Igen. Ha megállítani nem is sikerül teljesen terheléses támadást, ha az alkalmazásunk el tudja nyelni a fennmaradó részét, azzal megelőzhető a szolgáltatáskiesés.

kártékony forgalom szűrése a tűzfalnál: Igen, például ha nem elosztott támadásról van szó, amely egy IP címről érkezik.

Vissza a kategóriáimhoz