**1. FELADAT**

**REBOARDING 2.0**

**A legjobb megoldást keressük a dolgozók biztonságos irodai vissza áramoltatására!**

A koronavírus miatti védekezés részeként az Accenture iroda is teljesen elnéptelenedett, hiszen hónapok óta mindenki otthonról dolgozik.  
Most, hogy a vírus terjedése lassult, ismételten igény lenne az iroda használatára. Mivel a munkatársak biztonsága kiemelten fontos, a HR útmutatása alapján megfelelő óvintézkedések mellett, csökkentett létszámmal és több hullámban lehet ismételten benépesíteni az irodát.  
A HR egy olyan alkalmazás (mobil applikáció) megvalósítását kéri, ami lehetővé teszi adott napra az igények jelzését, és a kapacitás függvényében az esetleges várólista kezelését.  
  
Az architektúra csapat ezért egy olyan JVM alapú JSON REST back-end megvalósítása mellett döntött, ami a következő funkciókat támogatja (ezzel szolgálva ki a beléptető rendszer és a mobil applikáció kéréseit):  
  
**Kapacitás kezelése**A rendszer a 250 fő befogadására alkalmas irodából csak a HR által előzőleg beállított %-ban megadott kapacitás mértékéig engedélyezi a foglalást.  
Kezdetben 10% kapacitással fog indulni a rendszer, később ezt lépésenként 20, 30, 50 majd 100%-ra fogják növelni. A várható használat során az egyes munkatársak csak rövid ideig fognak az irodában tartózkodni, ezért ha valaki elmegy, helyette egy újabb kolléga mehet be.  
  
**Várólista kezelése**Amennyiben valaki a napi kapacitáson felül kíván foglalni, akkor várólistára kerül. A várólistás pozíció az adott napon belül fog csak frissülni, amikor valamely pozícióban előtte szereplő munkatárs elhagyja az irodát.  
  
**Az API a következő endpoint-okkal rendelkezik:**Register: Irodai helyfoglalási igény jelzése az applikációból:  
A felhasználó azonosítója alapján történik az adott napra az igény jelzését követően.  
A rendszer elmenti az igényt, és amennyiben az adott napon van még szabad hely, akkor azonnal elfogadja az igényt, ellenkező esetben a kérés várólistára kerül, ekkor a várólistában elfoglalt pozíciót kell visszaadni.  
  
**Status: A felhasználó listapozíciójának lekérése:**A mobil alkalmazás periodikusan, a felhasználó azonosítója alapján kéri le az aktuális várólista pozíciót, így a felhasználó dönthet hogy mikor indul el otthonról az irodába.  
  
**Entry: A beléptetéshez (a kártyaolvasó terminálok használják):**A felhasználó azonosítója alapján a rendszer az adott napon vagy engedélyezi a belépést, vagy megtagadja, ha az irodában levő jelenlegi szabad kapacitás kisebb, mint a felhasználó várólistában elfoglalt jelenlegi pozíciója.  
A belépések sorrendje nem kötelezően a foglalás sorrendjében kell történjen, de a rendszer meg kell akadályozza az alacsonyabb sorszámmal történt foglalások elől a helyek "elhappolását". Ez azt jelenti, hogy, ha most két szabad hely is van az irodában, és a 2-es sorszámú várakozó előbb érkezik, mint az 1-es, akkor a 2-es bemehet, de a 3-as már nem (hiszen akkor a később érkező 1-es már nem tudna bemenni).  
  
**Exit: A kilépéshez (a kártyaolvasó terminálok használják):**Amikor valaki elhagyja az irodát, akkor a rendszer frissíti a szabad kapacitást, így ezt követően a beléptetés már az eggyel nagyobb sorszámú várólistás pozícióval is beengedi az érkező kollégát.  
  
**A megvalósítás során törekedni kell a következőkre:**

* **JSON üzenet formátumok**
* **HTTP válasz kódok megfelelő használata**
* **Beszédes, angol nyelvű változó, függvény és osztály elnevezések**

**Az elkészült modulokat megfelelő tesztekkel kell ellátni (ajánlottan unit teszt és API teszt), amelyek segítségével ellenőrizhető a helyes működés. Minimum elvárás megfelelő számú hívás helyes kezelésének demonstrálása (pl.: postman scriptek segítségével)  
Agilis szoftverfejlesztésnek megfelelő szintű kommentezettség (javadoc)  
  
Előny az értékelésnél:**

* Kódolási szabályok követése (clean code)
* Tisztán microservice alapú megoldás
* Átgondolt szoftver architektúra használata
* Egyszerű és innovatív megoldások használata
* Skálázhatóság, felhő alapú telepítés támogatása (docker, kubernetes)
* TDD vagy BDD használata a teszteléshez (pl.: cucumber)
* Szabványos és elterjedt technológiák használata (pl.: spring-boot, swagger)

Amennyiben a megvalósítás során adatbázis használata mellett döntenek, akkor elegendő valamely in-memory (pl.: H2) adatbázis, vagy bármilyen (docker) container alapú adatbázis használata (ebben az esetben kérjük a docker indításához szükséges konfiguráció elküldését is).  
  
**Feladat leadás:  
A megoldások beadása GitHub/GitLab repository megosztásával a JunctionApp felületén történik. Leadási határidő: 06.16 23.59.**