Alkalmazott Informatikai Tanszék Tanév: 2018/2019
II. félév

Tantárgyi tematika és ütemterv
az **Informatikai Rendszerek Integrálása** c. tárgyhoz
Mérnök Informatikus MSc szakos hallgatók számára

A tárgy előadója és gyakorlatvezetője: Dr. Nehéz Károly
Tankör: Mérnök Informatikus MSC, II. évf.
Az előadások helye és ideje: órarend szerint

Tantárgyi követelmények:
Az aláírás megszerzésének feltétele a féléves zárthelyi és féléves feladat elégséges jegyre történő megírása. A vizsga szóbeli anyaga a tematika azon része, mely az előadásokon elhangzott. A tárgyat szóbeli és írásbeli vizsga zárja.

Féléves ütemterv

	Előadás	Gyakorlat
1	Informatikai rendszerek integrálásának alapfogalmai, osztályozása.	Eclipse J2EE platform bemutatása
2	TCP socket alapú integráció	Jboss alkalmazáskiszolgáló telepítése és konfigurációja
3	RPC (remote process call) alapú integráció	Java Socket TCP/UDP
4	Object request browser (ORB) alapú integráció	JAVA RMI
5	Üzenetkezelő rendszerek alkalmazása az integráció során	Java tools (JConsole), Dead lock
6	Webszolgáltatások (JAX-WS/JAX-RS) alkalmazása az integrációs folyamatban	Servlet, EJB 3.x
7	SOA – Service Oriented Architecture	Webszolgáltatás mintapélda
8	Loose coupling (laza összekpcsolás), Interface vs. Payload	JMX mintapélda
9	File alapú adatmegosztás	JMS, MDB példák
10	Adatbázis alapú adatmegosztás, Microsoft specifikumok	JTA példák
11	A Corba integrációs modell	Összetett alkalmazás példa
12	Enterprise Service Bus (ESB) alapjai és alkalmazása	JBOSS ESB bemutatás 1.
13	Felhő technológia az integrációs folyamatban	JBOSS ESB bemutatás 2.
14	Elővizsga, Pótlások	Feladat bemutatás

Kötelező és ajánlott irodalom:

1. Raffai Mária: Információrendszerek fejlesztése és menedzselése. Novadat kiadó. Budapest, 2003.
2. Juhász Sándor: Vállalti Információs Rendszerek műszaki alapjai, Szak Kiadó, Budapest 2011.
3. Tóth Tibor: *Minőségmenedzsment és Informatika.* Műszaki könyvkiadó, Budapest, 1999.
4. Kovács László: OLAP rendszerek. *Elektronikus* jegyzet, www-db.iit.uni-miskolc.hu
5. William Wake: Extreme Programming Explored. Addison-Wesley Professional; 1st edition, 2001.
6. D. S. Linthicum: Enterprise Application Integration. Addison Wesley, 1999.
7. G. Hohpe, B. Woolf: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, 2003.
8. D. Chappel: Enterprise Service Bus: Theory in Practice. O'Reilly Media, 2004.
9. T. Erl: Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall Ptr, 2005.

Miskolc, 2019.02.01.

……..
Dr. Nehéz Károly
egyetemi docens

Miskolci Egyetem
Alkalmazott Informatikai Tanszék

Név:.....……
Tanulókör:.....
Neptun kód:…….....

Információs rendszerek integrálása
Minta Zárthelyi feladat

1. Röviden jellemezze a TCP socket alapú kommunikációt. Mit jelent az on-demand csatorna. Miért az a leggyorsabb átvitel, ha UDP alapú socketet használunk?

2. Mit jelent az ORB kifejezés? Jellemezze röviden.

3. Mire szolgál a UDDI és WSDL a webszolgáltatások esetén?

4. Mit jelent a data enrichment az ESB-vel megvalósított integráció során?

5. Hogyan definiáljuk a szoftver integráció fogalmát?

6. Jellemezze a Loose coupling (laza összekapcsolás) fogalmát.

7. Mit jelent az Interface és Payload szemantika?

8. Jellemezze a file alapú adatmegosztást?

9. Mi a feladata a halott levél csatornának az üzenetküldő rendszerkben?

10. A felhő technológiában mire szolgál a Node Controller (NC)?

Féléves feladatok általános követelményei
Architektúra

A feladatok tipikusan olyan egyszerű alkalmazás integrációk, amelyek a JBoss vagy Wildfly alkalmazás szerver, vagy sima socket kommunikációval vagy django framework vagy angularjs segítségével oldahatóak meg.

Feladatok beadása

A feladatot a félév végén kell leadni, személyesen bemutatva. Lehet saját laptopon is vagy a labor gépein. Csak email-ben elküldött megoldásokat nem fogadunk el.

Beadási határidő

Az utolsó tanítási héttel bezárólag minden gyakorlaton. Ez után pótlás lehetséges.

Feladatok
1.

Készítsen egy alkalmazást, amely 2 kliensből áll. Az első kliens a '/queue/colorQueue' üzenetsorra pont-pont csatlakozással véletlenszerűen RED, GREEN és BLUE paraméterrel ellátott üzeneteket küld 1 másodpercenként. Készítsen három MDB-t (üzenet vezérelt bean) amelyek filterrel a 'RED', 'GREEN' és a 'BLUE' paraméterrel ellátott üzeneteket kapják kizárólag. Minden 10 megkapott üzenet után az MDB-k a '/queue/colorStatistics' sorra küldenek egy üzenetet, ami azt jelzi, hogy 10 (adott színű) üzenetet feldolgoztak. Készítsen egy második klienst, ami a '/queue/colorStatistics' sorrol olvassa a statisztikát és a konzolba kiírja hogy pl. '10 'RED' messages has been processed'

2.

Készítsen egy alkalmazást, amely 3 kliensből áll. Az első kliens a '/queue/colorQueue' üzenetsorra pont-pont csatlakozással véletlenszerűen RED, GREEN és BLUE paraméterrel ellátott üzeneteket küld 1 másodpercenként. Készítsen három MDB-t amelyek filterrel a 'RED', 'GREEN' és a 'BLUE' paraméterrel ellátott üzeneteket kapják kizárólag. Az MDB-k véletlenszerűen átlagosan 10 ből 3 szor, rollback-elik az üzenetet, ami így a halott levél csatornára kerül. Minden 10 sikeresen megkapott (nem rollback-elt) üzenet után az MDB-k a '/queue/colorStatistics' sorra küldenek egy üzenetet, ami azt jelzi, hogy 10 (adott színű) üzenetet feldolgoztak. Készítsen egy második klienst, ami a '/queue/colorStatistics' sorrol olvassa a statisztikát és a konzolba kiírja hogy pl. '10 'RED' messages has been processed'. A harmadik kliens a '/queue/DLQ' halott levél csatornáról a konzolon jelzi, ha egy üzenetet nem dolgoztak fel.

3.

Készítsen alkalmazást, amely az 1. feladat alapján működik, annyi különbséggel, hogy a kliens egy webszolgáltatás, amely SOAP-on keresztül küldi a véletlenszerű színeket a a JBoss webszolgáltatásnak, és a webszolgáltatás küldi tovább '/queue/colorQueue' üzenetsorra pont-pont csatlakozással az üzeneteket. Az ez utáni teendők megegyeznek az 1.-es feladatban leírtakkal. A lényeges különbség az, hogy a kliens nem kapcsolódik közvetlenül az üzenetsorra, hanem a JBoss ban futó szolgáltatás küldi tovább az üzenetet a sorra.

4.

Készítsen alkalmazást, amely az 2. feladat alapján működik, annyi különbséggel, hogy a kliens egy webszolgáltatás, amely SOAP-on keresztül küldi a véletlenszerű színeket a a JBoss webszolgáltatásnak, és a webszolgáltatás küldi tovább '/queue/colorQueue' üzenetsorra pont-pont csatlakozással az üzeneteket. Az ez utáni teendők megegyeznek az 2.-es feladatban leírtakkal. A lényeges különbség az, hogy a kliens nem kapcsolódik közvetlenül az üzenetsorra, hanem a JBoss ban futó szolgáltatás küldi tovább az üzenetet a sorra.

5.

Készítsen alkalmazást amelynek egyik kliense a '/queue/colorQueue' üzenetsorra véletlen szöveges üzeneteket küld. ('RED', 'GREEN', 'BLUE') szöveggel 0.5 másodpercenként. Készítsen 3 további klienst amelyek folyamatosan 'hallgatják' a '/queue/colorQueue' üzenetsort, ha valamelyik üzenetet olvas a sorról, akkor véletlenszerűen (2000-9000) milliszekundum ideig várakozik, majd utána újra hallgatja az üzenetsort. (A véletlenszerű feldolgozási idő miatt -e közben egy másik üzenetet a következő kliens fogja feldolgozni). A 3 kliensnek ne legyen konzol outputja, azaz ne írjon ki semmit a konzolra. Minden üzenet feldolgozása után a '/queue/processedColors' sorra küldjenek egy üzenetet, ami azt jelzi,

hogy feldolgozták a feladatot. Készítsen egy klienst, ami a '/queue/processedColors' sort olvassa és a megjelenő üzeneteket kiírja az outputra. pl: 'Client 2, processed a 'BLUE' message'. Ennél a feladatnál nem kell a JBOSS-ban komponenst létrehozni.

6.

Készítsen egy képzeletbeli prepaid mobilfeltöltő alkalmazást amely egy tcp alapú socket szerver lesz egy adott porton fogadja a kliensen kéréseit. A kliensek atm automaták, amelyek a következő protokoll szerint működnek: 1.) a kliens egy tesztüzenetben elküldi a feltölteni kívánt telefonszámot, 2.) ha a szám feltölthető, akkor OK egyébként ERROR üzenettel megszakítja a tranzakciót. OK üzenet esetén egy tranzakció azonosítót kapunk, amely a következő 3. lépésben azonosítja a folyamatot. 3.) a kliens a tényleges feltöltést kezdeményezi a korábban kapott tranzakció azonosítóval, de újra el kell küldeni a telefonszámot és a feltöltési összeget.

A lehetséges feltöltési összegek: 3000,5000,10000,15000 Készítsen egy lekérdező klienst, amely egy adott telefonszámhoz tartozó korábbi tranzakciókat listázza. Használjon adatbázist a tranzakciók tárolására a szerver alkalmazásban.

Indítson 3 klienst, amelyek egy előre adott telefonszám halmazt használnak és véletlenszerűen teszt és feltöltés tranzakciókat indítanak.

7.

Készítsen REST API-t django frameworkben, amely egy videokölcsönző adatbázisát használja. Készítsen egy angularJS klienst, amely filmeket tud hozzáadni az adatbázishoz. Készítsen, a kölcsönzéshez használható funkciót, amely a filmeket kikölcsönzött, illetve visszavitt státuszra állítja. Listázza a kikölcsönzött és kölcsönözhető filmeket.

8.

A 7. feladat alapján készítsen egy könyvtári rendszert könyvek kölcsönzéséhez.

9.

A 7. feladat alapján készítsen egy autókölcsönző rendszert autók kölcsönzéséhez.

10.

A 7. feladat alapján készítsen egy betegnyilvántartó rendszert betegek adatinak kezeléséhez. Az adatfelvitel után készítsen a vizitek menedzseléséhez alkalmas modult.

11.

Készítsen el egy szabadon választott feladatot innen: **http://ait2.iit.uni-miskolc.hu/oktatas/doku.php?id=tanszek:oktatas:informatikai_rendszerek_epitese:feleves_feladat** django framework segítségével, készítsen unit, funkcionális és GUI teszteket (egyenként kettőt). Telepítsen Jenkins integrációs szervert és futtassa vele a teszteket.