### GSO3 Killer App

**Inleiding**

Het doel van het vak GSO3 om ‘real-time’ interactieve applicatie te ontwerpen en te realiseren. In deze opdracht ga je je eigen GSO3 applicatie maken, dus je eigen GSO3 ‘Killer App’. De functionaliteit van de applicatie mag je helemaal zelf bepalen. Mogelijke voorbeelden zijn: een gedistribueerd spel, gedistribueerd internetbankieren en een gedistribueerd whiteboard. De volgende voorwaarden zijn van toepassing:

1. De applicatie moet ‘real-time’ interactief zijn. Met ‘real-time’ wordt bedoeld dat een actie van een gebruiker leidt tot een verandering op het scherm van een andere gebruiker zonder noemenswaardige vertraging.
2. De applicatie bestaat uit tenminste 3 verschillende componenten.
3. Voor tenminste één van de communicatiemogelijkheden tussen componenten wordt Remote Method Invocation (RMI) toegepast volgens het client-pull mechanisme.
4. Voor tenminste één van de communicatiemogelijkheden tussen componenten wordt Remote Method Invocation (RMI) toegepast volgens het server-push mechanisme.
5. De applicatie bevat een registratie en login-mechanisme. Er wordt gebruik gemaakt van sessies om ongeoorloofd gebruik tegen te gaan.
6. Er wordt gebruik gemaakt van een database voor persistentie van gegevens.
7. De applicatie wordt ontwikkeld volgens het waterval model.

Indien je naast RMI nog een andere communicatietechniek wilt toepassen, overleg dan met de docent.

**Tip:** hoewel het een individuele opdracht is is het toegestaan om elkaars werk te bekijken en van elkaar te leren. De tip is om elkaars documenten en software te reviewen. Het is toegestaan om (delen van) de aangeleverde voorbeelden (RMI Student Administration, FontysPublisher, FontysPublisherWhiteboard, BankierenNoObserverFX) te gebruiken in jouw killer-app, uiteraard met bronvermelding in de code.

**Op te leveren producten:**

1. User Requirements Specification (URS)
2. Software Architectuur Document (SAD)
3. Systeemtestplan inclusief testgevallen
4. Gedistribueerde applicatie inclusief unittests
5. Systeemtestrapport

**Globale planning (zie Canvas voor inlevermomenten)**

**Les 9: User Requirements Specification (URS)**

1. Maak een beschrijving van je killer app en neem deze op in je URS.
2. Stel een URS op aan de hand van het template document URS-template.docx. Voor een voorbeeld, zie URS-internetbankieren.docx. Zie ook de slides van Les 1.
3. Lever het URS in via Canvas en bespreek het document met de docent.

**Les 10: Componenten en sessies**

1. Stel een klassediagram op van het domein van jouw applicatie in visual paradigm.
2. Stel een componentendiagram op in visual paradigm.
3. Bedenk hoe je sessies kunt realiseren voor jouw applicatie: hoe zorg je ervoor dat een persoon niet ongeoorloofd een actie uit kan voeren waartoe hij niet gerechtigd is?

**Les 11: Software Architectuur Document (SAD)**

Maak een architectuur document op basis van het template document ArchitectuurDocumentTemplate.docx.

1. Werk het hoofdstuk ‘Domeinmodel’ uit.
2. Werk het hoofdstuk ‘Opdeling in componenten’ uit. Geef aan welke objecten (van welke klasse) er actief zijn binnen welke component. Zie de video StudentAdminComponentDiagram <https://www.youtube.com/watch?v=FBESl0gF6qo>
3. Werk het hoofdstuk ‘Communicatie tussen componenten’ uit. Stel een of meerdere (communicatie) klassediagrammen op in Visual Paradigm zodat de RMI-techniek tot uitdrukking komt in het model. In het geval van RMI volgens het server-push principe moet je gebruik maken van de klasse RemotePublisher en de interfaces IRemotePublisherForListener en IRemotePropertyListener (zie FontysPublisherHandleiding.docx). Geef in het klassediagram aan hoe je gebruik maakt van de klasse UnicastRemoteObject en de interfaces Remote en Serializable.
4. Zorg ervoor dat in het communicatie-klassediagram duidelijk wordt door welke klassen de interfaces van het componentdiagram gerealiseerd worden en aan welke interfaces de verschillende klassen behoefte hebben.
5. Communicatie tussen componenten komt neer op (remote) aanroep van methoden. De methoden waar een component behoefte aan heeft worden in de vorm van een 1 of meer interfaces gepresenteerd. Definieer de methoden (inclusief parameters en return waarden) voor alle interfaces in de klassediagrammen.
6. Vraag feedback aan medestudenten en de docent tijdens deze of de volgende les.

**Les 12: vervolg Software Architectuur Document**

Breid je architectuur document uit als volgt:

1. Maak een gedetailleerd ontwerp per component (klassediagram, sequence diagrammen).
2. Werk het hoofdstuk ‘Persistentie per component’ uit.
3. Werk het hoofdstuk ‘Detailed design per component’ uit.
4. Werk het hoofdstuk ‘Realisatie niet-functionele eisen’ uit.
5. Maak een deployment diagram in visual paradigm en werk het hoofdstuk ‘Deployment’ uit.
6. Werk het hoofdstuk ‘Specificatie van interfaces’ uit. Defineer voor elke methode de naam, parameters (inclusief type) en returnwaarden, de preconditie, een beschrijving van de methode en mogelijke excepties. Uiteindelijk moet elke interface worden gespecificeerd met behulp van javadoc.
7. Schrijf de inleiding.
8. Laat je SAD reviewen door een medestudent.
9. Lever het SAD in via Canvas en bespreek het document met de docent.

**Les 13 - 16**

Realiseer je GSO3 Killer App:

1. Gebruik GitLab voor versiebeheer.
2. Zorg dat je voor alle functionaliteit javadoc en unittests op orde hebt.
3. Gebruik JaCoCo om unittest coverage aan te tonen (zie S3\_JaCoCoPluginNetBeans.docx voor beschrijving installatie en gebruik JaCoCo).
4. Gebruik SonarQube om een indruk te krijgen van de kwaliteit van je code.
5. Stel een systeemtestplan met testgevallen op aan de hand van de templates Template TESTPLAN.docx en Template TESTGEVALLEN.docx.
6. Pas het architectuur document aan indien nodig.
7. Voer de systeemtest uit en stel een testrapport op.
8. Voeg alle op te leveren producten samen in een .zip of .rar bestand en lever dit in in Canvas.