Übungsangabe
184.153 Distributed Systems Engineering
2016S, VU, 2.0h, 3.0EC

1 Einleitung

Der österreichische Ministerrat hat beschlossen, die Wartelisten für planbare Operationen transparent zu machen. Beachten Sie dazu die folgenden Medienberichte:

PLANBARE OPERATIONEN

Nationalrat: Ja zu transparenten OP-Wartelisten
08. Juli 2011 19:11

Bevorzugungen bei planbaren Operationen sollen der Vergangenheit angehören

Wien - Bevorzugungen für Privatpatienten bei Hüft- oder Augenoperationen sollen der Vergangenheit angehören. Der Nationalrat hat Freitagabend einstimmig Gesundheitsminister Alois Stöger dazu ermächtigt, ein transparentes Wartelisten-System im Gesundheitswesen zu etablieren.

Das Wartelistenregime für planbare Eingriffe soll in anonymisierter Form vorliegen und für Operationen etwa in der Augenheilkunde, der Orthopädie, orthopädischen Chirurgie sowie Neurochirurgie gelten. Die für den Eingriff vorgemerkte Person ist auf ihr Verlangen über die gegebene Wartezeit zu informieren.

Quelle: http://derstandard.at/1308680805456/Planbare-Operationen-Nationalrat-Ja-zu-transparenten-OP-Wartelisten

CHIRURGISCHE EINGRIFFE

OP-Wartelisten bleiben in Länderhand
18. Mai 2011 18:23

Wie die vom Ministerrat beschlossene Datenbank für planbare Operationen in der Praxis aussehen soll, obliegt den Ländern. Auch private Krankenhäuser müssen ab Frühjahr 2012 ihre Wartelisten transparent machen

Wien - Dass es künftig Datenbanken geben soll, mittels derer Wartelisten für Operationen transparent gemacht werden, darauf hat man sich im Ministerrat bereits geeinigt. Wie diese genau aussehen - und wie wirksam sie daher sind - liegt aber im Einflussbereich der Länder. Denn der Gesetzesentwurf von Gesundheitsminister Alois Stöger ist nur eine Rahmenvorgabe, die Ausführung obliegt den Landespolitikern. Grundsätzlich sollen Spitäler ab dem Frühjahr 2012 Online-Wartelisten führen

müssen, um zu verhindern, dass Privatpatienten bei planbaren Operationen Kassenpatienten bei der Terminvergabe "überholen" können.

Quelle: http://derstandard.at/1304552267364/Chirurgische-Eingriffe-OP-Wartelisten-bleiben-in-Laenderhand

1.1 Aufgabe

Im Rahmen dieser Übung werden Sie eine moderne Softwarelösung für die Online-Wartelistenverwaltung planbarer Operationen umsetzen.

Die Module des Systems sollen als „Microservices“ umgesetzt werden, die in leichtgewichtigen Deployment- und Runtime-Umgebungen – sogenannten „Containern“ - betrieben werden. Dadurch ergibt sich – im Gegensatz zu einem monolithisch aufgebauten System - ein lose gekoppeltes, verteiltes System, wodurch grundlegende Probleme verteilter Systeme im Design und in der Umsetzung gelöst werden müssen.

Das User Interface soll als Web Applikation ausgelegt werden.

1.2 Funktionale Anforderungen

Die funktionalen Anforderungen stellen sich in einer Art Use-Case Beschreibung wie folgt dar (Akteure werden unterstrichen dargestellt):

1) Die interessierte Öffentlichkeit kann jederzeit Einsicht in die OP-Wartelisten der einzelnen Krankenhäuser nehmen
    a. Dabei wird eine Liste von angebotenen OP Slots angezeigt und eine etwaige Reservierung durch einen Patienten
        i. Aufgrund des Datenschutzes wird der Name des Patienten, dem dieser Slot zugeteilt wurde, nicht angezeigt
        ii. Die Liste ist nach Datum sortiert - der zeitlich nächste OP-Slot wird ganz oben angezeigt
        iii. Die Liste kann nach Datum, Uhrzeit von, Uhrzeit bis, OP Typ, Krankenhaus, Arzt/Ärztin und Status eingeschränkt werden
        iv. *Optional*: Neue OP Termine werden dynamisch der Liste hinzugefügt und begonnene OP Termine aus der Liste entfernt (ohne Refresh der Seite)
2) Patienten können
    a. Ihre persönliche Reservierungsliste von OP Slots anzeigen lassen
        i. Die Liste ist nach Datum sortiert - der zeitlich nächste OP-Slot wird ganz oben angezeigt
        ii. Die Liste kann nach Datum, Uhrzeit von, Uhrzeit bis, OP Typ, Krankenhaus und Arzt/Ärztin eingeschränkt werden
    b. Ihre persönlichen Notifications anzeigen lassen
        i. Die Stakeholder bekommen eine Notification, wenn eine durchgeführte Reservierung eines OP Slots für einen

bestimmten OP Typ für diesen Patienten erfolgreich war bzw. eine bestehende Reservierung erfolgreich wieder gelöscht wurde
        ii. *Optional*: Eingehende Notifications werden in Echtzeit angezeigt
3) Ärzte/Ärztinnen (Operateure) können
    a. Ihre persönliche OP-Slot Liste als Operateur anzeigen lassen
        i. Die Liste ist nach Datum sortiert - der zeitlich nächste OP-Slot wird ganz oben angezeigt
        ii. Die Liste kann nach Datum, Uhrzeit von, Uhrzeit bis, OP Typ, Krankenhaus und PatientIn eingeschränkt werden
    b. Eine Terminreservierung für einen Patienten für einen passenden OP Slot eines Krankenhauses vornehmen. Der Arzt gibt nur den gewünschten OP Zeitraum, die OP Slot Länge und den Patienten an - aufgrund der von den Krankenhäusern angelegten und verfügbaren OP Slots, deren Typ (Orthopädie etc.), der Geo-Daten und des Zeitraums nimmt ein zu erstellendes Microservice eine Zuteilung vor.
    c. Eine so vorgenommene Terminreservierung wieder stornieren
    d. Ihre persönlichen Notifications anzeigen lassen
        i. Der Arzt bekommt eine Notification, wenn eine von ihm durchgeführte Reservierung eines OP Slots für einen bestimmten Patienten erfolgreich war bzw. die Löschung einer bestehenden Reservierung erfolgreich war
        ii. *Optional*: Eingehende Notifications werden in Echtzeit angezeigt
4) Krankenhäuser können
    a. Ihre OP-Slot Liste anzeigen lassen. Dabei kann es Slots geben, die bereits reserviert sind und noch freie angebotene Slots. Der Unterschied ist entsprechend zu visualisieren.
        i. Die Liste ist nach Datum sortiert - der zeitlich nächste OP-Slot wird ganz oben angezeigt
        ii. Die Liste kann nach Datum, Uhrzeit von, Uhrzeit bis, OP Typ, Arzt/Ärztin und PatientIn eingeschränkt werden
    b. OP-Slots für geplante OPs eintragen. Bei der Eintragung wird die Länge des OP Slots festgelegt
    c. OP-Slots für geplante OPs wieder löschen (sofern sie noch nicht reserviert sind)
    d. Ihre persönlichen Notifications anzeigen lassen
        i. Durch die vollständige asynchrone Arbeitsweise des Systems bekommt ein Krankenhaus eine Notification, wenn eine für dieses Krankenhaus durchgeführte Reservierung eines OP Slots für einen bestimmten Patienten erfolgreich war bzw. die Löschung einer bestehenden Reservierung erfolgreich war
        ii. *Optional*: Eingehende Notifications werden in Echtzeit angezeigt

OP Slot Listen GUI Mockup:

Anzeige für die Öffentlichkeit:

Datum	von	bis	Typ	KH	Arzt/Ärztin	Status
12.03.2016	12:00	13:30	Augen	SMZ	Dr. Aufbesser	reserviert
13.03.2016	08:00	10:00	Ortho	LK-K	-	frei
15.03.2016	06:00	08:00	Kardio	LK-B	Dr. Meier	reserviert

…

Anzeige für PatientInnen – z.B. persönliche Terminliste für Patientin Adelheid Amann:

Datum	von	bis	Typ	KH	Arzt/Ärztin
12.03.2016	12:00	13:30	Augen	SMZ	Dr. Aufbesser
15.03.2016	06:00	08:00	Kardio	LK-B	Dr. Meier

…

Anzeige für Ärzte/Ärztinnen - Beispiel für Dr. Aufbesser:

Datum	von	bis	Typ	KH	PatientIn	
12.03.2016	12:00	13:30	Augen	SMZ	Adelheid Amann	[x]

…

[Reservierung für eine PatientIn vornehmen]

[x] … Schaltfläche, um eine bestehende Reservierung zu löschen
[Reservierung für einen Patienten/eine Patientin vornehmen] … Schaltfläche, um eine neue Reservierung für einen Patienten/eine Patientin vorzunehmen

Anzeige für Krankenhäuser – Beispiel für SMZ West:

Datum	von	bis	Typ	Arzt/Ärztin	PatientIn	
12.03.2016	12:00	13:30	Augen	Dr. Aufbesser	Adelheid Amann	
15.04.2016	06:00	08:00	Kardio	Dr. Eder	Alfons Berger	
15.05.2016	10:00	11:00	Kardio	-	-	[Slot löschen]

…

[Neuen Slot anlegen]

[Slot löschen] … Schaltfläche, um einen bestehenden, noch freien Slot zu löschen
[Neuen Slot anlegen] … Schaltfläche, um einen neuen Slot für dieses Krankenhaus anzulegen

Anmerkungen zu den funktionalen Anforderungen (FAQ):

1) Ist eine Ärztin einem Krankenhaus zugeordnet und kann sie nur für ihr Krankenhaus Reservierungen vornehmen?
   Antwort: Es gibt keine fixe Zuordnung Ärztin - Krankenhaus. Ärztinnen operieren Patienten im zugeteilten Krankenhaus.

2) Kann die Patientin eine Reservierungsanforderung schicken und - falls die Suche erfolgreich ist - wird automatisch eine Reservierung getätigt?

Antwort: Laut Angabe können Patientinnen nichts dergleichen. Sie müssen die Ärztin konsultieren und diese stellt Reservierungsanforderungen für OP Slots.

3) Ist eine Patientin einer Ärztin zugeordnet oder kann sich die Patientin bei der Reservierung quasi eine Ärztin aussuchen?
   Antwort: Die Patientin darf sich die Ärztin aussuchen (wenn sie zusatzversichert ist) oder aber sich zu einer Ärztin überweisen lassen - es ist davon auszugehen, dass die Patientin je nach Beschwerden die entsprechende Ärztin aufsucht.

4) Darf die Ärztin über das User Interface Termine direkt in die Datenbank eintragen?
   Antwort: Das Krankenhaus legt Termine (OP-Slots) an und die Ärztin reserviert einen entsprechenden Termin für eine Patientin.

Anforderungen zur Sicherheit:

Nachdem in der Übung nur ein Prototyp umgesetzt wird, soll die Rolle (Public, Patient, Arzt, Krankenhaus) über ein Dropdown im oberen Bildschirmbereich ausgewählt werden können. Je nach Auswahl werden die für die Rolle relevanten Informationen angezeigt. Es gibt also kein klassisches Login mit Usernamen und Passwort.

Optional: Anzeige aller Rollen parallel auf einem Schirm, damit Änderungen über alle Rollen hinweg gleichzeitig angesehen werden können.

Anforderungen zu den Services und Schnittstellen:

1) Die Benutzer bedienen das System über ein Web GUI in einem Browser.
2) Für jedes Microservice ist eine geeignete Schnittstelle zu entwerfen.
3) Für jedes Microservice ist eine geeignete Datenhaltung zu entwerfen.
4) Microservices werden nicht direkt, sondern über ein API Gateway angesprochen. Auch das API Gateway soll in einem eigenen Docker Container laufen.

Anforderungen zum Deployment:

Jedes Microservice läuft in einem eigenständigen **Docker** Container und kommuniziert – je nach Implementierung – synchron bzw. asynchron über die entworfenen Interfaces mit anderen Microservices.

Anforderungen zur technischen Umsetzung:

Mindestens zwei Microservices sind in Java zu entwickeln.

Verpflichtend zu verwenden sind:

1) Java 7+ für zumindest zwei Microservices
2) Hystrix [https://github.com/Netflix/Hystrix]
3) MongoDB [http://www.mongodb.org] mit nativ unterstützter Geo-Suche für den OPscanner
4) Docker [https://www.docker.com]
5) REST Schnittstelle für das API Gateway (siehe unten).

Die folgenden Frameworks können, müssen aber nicht verwendet werden:

1) Spring Boot [http://projects.spring.io/spring-boot]
    a. Spring Boot eignet sich besonders für die Implementierung von Microservices.
2) DropWizard [http://dropwizard.io]
    a. DropWizard eignet sich besonders für die Implementierung von Microservices.
3) Spring Data MongoDB [http://projects.spring.io/spring-data-mongodb/]
    a. Spring Data MongoDB bietet eine Java Repository Abstraktion zum Zugriff auf MongoDB Funktionen.

Tutorials:

1) Docker Tutorial
    a. https://www.docker.com/tryit/
2) Hystrix Tutorial
    a. https://github.com/Netflix/Hystrix/wiki/How-To-Use
3) Spring Boot Tutorial
    a. https://spring.io/guides/gs/spring-boot/
4) MongoDB Geospatial Data
    a. http://blog.mongolab.com/2014/08/a-primer-on-geospatial-data-and-mongodb/

Links zum Thema Microservices:

1) http://martinfowler.com/articles/microservices.html
2) http://blog.ralfw.de/2014/08/warnung-vor-dem-microservice-versuch.html
3) http://microservices.io/

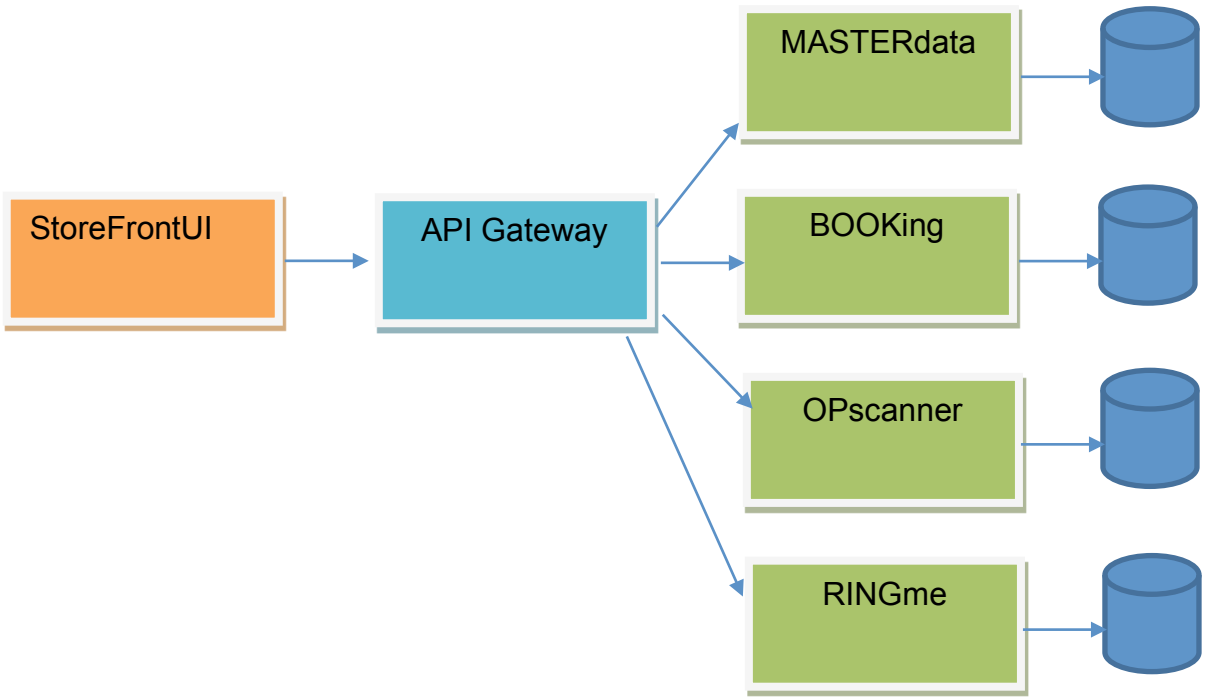
1.3 Grundlegende Lösungs-Architektur

Die grundlegende Lösungs-Architektur besteht aus einem *Store Front User Interface*, einem *API Gateway* und einer Menge von Microservices.

1) **Store Front User Interface als Web GUI**
    a. Das StoreFrontUI implementiert ein Web GUI für Anwender unter der Verwendung des API Gateways.

2) **API Gateway**
    a. Aufgabe des API Gateways ist die zur Verfügung Stellung eines geeigneten APIs für verschiedene Clients. Das API Gateway fasst die APIs der Microservices dabei geeignet zusammen.
    b. Um im Falle der Nichtverfügbarkeit von Microservices im Backend die Reaktivität des Systems sicherzustellen, ist das Framework „Hystrix“ zu verwenden.
    c. Das API Gateway soll über eine REST Schnittstelle verfügen.
3) **MASTERdata**
    a. Das Service MASTERdata verwaltet die Daten aller Spitäler, Ärzte, Patienten.
4) **BOOKing**
    a. Das Service BOOKing verwaltet jegliche Reservierungsinformationen.
5) **OPscanner**
    a. Das Service OPscanner nimmt Reservierungsanfragen entgegen und versucht, den zeitlich und geografisch (zum Patienten) nächsten OP Slot im eingegebenen Umkreis [in km] innerhalb eines gewünschten Termin-Zeitraums zu reservieren
        i. Dazu müssen Patienten und Krankenhäuser mit Geokoordinaten ausgestattet werden, um eine Geo-Suche durchführen zu können
        ii. Die Ärzte operieren die Patienten dann in diesem – aufgrund der Geo-Suche ermittelten - Krankenhaus
        iii. Hinweis: MongoDB unterstützt Geo-Suchen nativ!
        iv. Im positiven Fall erfolgt eine Reservierung und die entsprechenden Notification-Requests werden an das Microservice „RINGme“ weitergeleitet
        v. Im negativen Fall wird keine Reservierung durchgeführt und die entsprechenden Notifications-Requests werden an den „RINGme“ weitergeleitet
        vi. Allgemein gilt: Krankenhäuser legen die Länge von OP Slots fest, der Arzt initiiert eine Terminreservierung, der OPscanner nimmt eine Terminreservierung vor (sofern möglich).
6) **RINGme**
    a. Das Service RINGme nimmt Notification-Requests entgegen und speichert sie für die weitere Verarbeitung - z.B. für die Anzeige im User Interface.

Architekturskizze:



- Jedes Microservice verfügt seine eigene Datenhaltung! D.h., dass kein Microservice auf die „Datenbank" eines anderen Microservices zugreifen darf. Weil Docker Container im Allgemeinen *stateless* sind, was einer persistenten Datenspeicherung nicht unbedingt entgegen kommt, reicht es aus, ein oder mehrere Datenbanksysteme direkt auf Betriebssystem-Ebene zu installieren (d.h. eine MongoDB Installation würde ausreichen, wenn alle Microservices MongoDB als Datenbanksystem benutzen!). Die Microservices in den Docker Containern greifen dann auf abgegrenzte Bereiche in diesen Datenbanksystemen zu – wie gesagt darf es keine Querzugriffe geben…!
- Die Microservices können natürlich untereinander kommunizieren.

Anforderungen an die Darstellung der Microservice-Eigenschaften in verschiedenen Systemzuständen beim Abgabegespräch:

- Während der Präsentation müssen Microservices beliebig gestartet und gestoppt werden (können).
- Das System muss sich dabei immer in einem reaktiven Zustand befinden. Hänger/Timeouts beim Reload sind tabu.
- Falls ein Microservice nicht gestartet, nicht erreichbar oder ausgefallen ist muss das User Interface diesen Ausfall sinnvoll kompensieren (d.h. zum Beispiel keine unschönen Anzeigefehler).
- Das User Interface muss **die Verfügbarkeit aller Microservices in einem eigenen Bereich pro Microservice anzeigen:**
    - Name des Microservices: Status (verfügbar / nicht verfügbar)

1.4 Verifikation

Testdaten:

Folgende Daten sind (zumindest) für die Tests automatisiert zu erzeugen (Namensgleichheiten mit existierenden Personen sind unerwünscht und rein zufällig):

Patienten:

1. Fr. Adelheid Amann
2. Hr. Alfons Berger
3. Fr. Beatrix Binder
4. Hr. Franz Fuchs
5. Fr. Gloria Greiner
6. Hr. Manfred Mann

Ärzte:

1. Dr. Albert Aufbesser
2. Dr. Emily Eder
3. Dr. Adam Augapfel
4. Dr. Maria Meier

Krankenhäuser:

1. SMZ West
2. LKH Mistelbach
3. LKH Tulln
4. KH Rudolfstiftung

OP Typen:

1. Augen
2. Ortho
3. HNO
4. Neuro
5. Kardio

OP Slot Längen:

1. 60 Minuten
2. 120 Minuten
3. 180 Minuten
4. 240 Minuten

Optional: Erstellung von umfangreicheren Testdaten

Tests:

Entwerfen und dokumentieren Sie zwei Testszenarien, in dem gewisse Backend-Services Requests blockieren und zeigen Sie die Funktionsweise von Hystrix.

Schreiben Sie zumindest einen Lasttest – z.B. unter Verwendung der „rest-assured" Library (http://code.google.com/p/rest-assured/) gegen das API Gateway.

Optional: Automatisierte GUI Tests mit z.B. Selenium können helfen, die Testabdeckung zu verbessern.

Abgabe:

Im Abgabegespräch werden Sie den Aufbau und die Funktion Ihrer Lösung präsentieren. Details dazu werden in Kapitel 3 dargelegt.

2 Deliverables

2.1 Projektplan

Erstellen Sie ein Dokument mit folgendem Inhalt:

Deliverable 1: Dokument: Projektplan

- Enthält eine kurze Projektbeschreibung mit den Projektzielen und Nicht-Zielen (Projektzieleplan)
- Enthält die Rollenvergabe im Projektteam (Projektorganigramm)
    - Wer ist der Projektleiter, wer die Projektteammitglieder?
- Enthält die Arbeitspaketspezifikation oder Work Breakdown Structure (WBS)
    - Welchen Aufwand planen Sie für die Tätigkeiten ein (Aufwandsschätzung)?
        - Wie hoch ist der Aufwand für jedes Projektteammitglied (in Personenstunden)?
        - Wie hoch ist der Gesamtaufwand (in Personenstunden)?
- Enthält die Ressourcenplanung und Milestoneplanung
    - Wer im Team macht was und bis wann?
    - Festlegung der Fertigstellungstermine für die Arbeitspakete und Deliverables
- Welche Methoden, Technologien und Werkzeuge werden Sie verwenden und wozu (technische Planung)?
- *Optional*: Projektbalkenplan oder Sprintplanung

Beachten Sie die zeitgerechte Abgabe des Projektplans (siehe „Übungsablauf und Organisation" unten)!

2.2 Architektur- und Designdokument

Erstellen Sie ein Dokument mit dem folgenden Inhalt:

Deliverable 2: Dokument: Architektur- und Designdokument

- Beschreibung des Software Architektur
  - Welche Komponenten / Microservices existieren?
  - Welche Schnittstellen existieren?
  - Was trägt Hystrix zur Qualität der Lösung bei?
- Stellen Sie den „Logical View“ und den „Deployment View“ wie im „4+1 Architectural Model“ von Philippe Kruchten dar (siehe dazu auch http://en.wikipedia.org/wiki/4%2B1_Architectural_View_Model)

2.3 Wartelistenimplementierung für planbare OPs

Erstellen Sie das in Übungsangabe beschriebene verteilte System.

Deliverable 3: Codeabgabe: Wartelistenimplementierung für planbare OPs

2.4 Wartungshandbuch

Erstellen Sie ein Dokument mit dem folgenden Inhalt:

Deliverable 4: Dokument: Wartungshandbuch

- Beschreibung der Entwicklungsumgebung
- Beschreibung der eingesetzten Frameworks
- Beschreibung des Build Prozesses
- Beschreibung der Unit, GUI- und Lasttests

2.5 Projektabschlussdokumentation

Jedes Teammitglied erstellt einen Projektabschlussbericht. Der Name des Teammitglieds muss aus dem Titel des Dokuments klar erkennbar sein!

Deliverable 5: Lessons Learned und Conclusio

- Beurteilen Sie den Ablauf des Projektes
  - Was hat gut funktioniert, was weniger gut? Begründen Sie ihre Beurteilung.
  - Haben Sie den ursprünglich geschätzten Aufwand überschritten? Um wieviel und warum?
  - Was würden Sie anders bzw. besser machen, wenn Sie mit ihrer jetzigen Erfahrung das Projekt nochmals machen müssten.
- Beurteilen Sie die finale Software
  - Was ist gut geglückt, welche Teile würden Sie beim nächsten Mal anders/besser lösen.
- Verfassen Sie ihr persönliches Fazit zur Lehrveranstaltung.

3 Übungsablauf und Organisation

- Über die Gruppenanmeldung im TISS System werden Übungsgruppen zu je drei (3) Personen gebildet (http://tiss.tuwien.ac.at).
- Laden Sie den Projektplan (Deliverable Dokument 1.1) **bis spätestens 15.04.2016, 24:00** im TUWEL hoch.
    - **Alle** Dokumente müssen in einem der folgenden Formate abgegeben werden: .rtf, .doc, .pdf oder .html! Im speziellen ist für den Projektplan kein Microsoft Project® oder anderes Projektmanagementtool-spezifisches Format erlaubt! Achten Sie darauf, dass keine Links auf externe Objekte (Bilder, Diagramme) in den Dokumenten vorhanden sind. Achten Sie darauf, dass die Dokumente nicht exorbitant groß sind (z.B. durch Einbinden großer Bilder).
    - **Alle** abgegebenen Dokumente müssen als Dateinamen bzw. Dokumenttitel die Nummer des Deliverables und den Dokumenttitel (wie in der Aufgabenstellung spezifiziert) tragen! Der Titel muss auf der ersten Seite sowie über Fußzeilen im ganzen Dokument ersichtlich sein.
        - Der Dateiname für die Projektplanabgabe als pdf ist also **„DSE16Gruppe<nn>Projektplan.pdf“**
        - <nn> entspricht ihrer Gruppennummer formatiert auf zwei Stellen
        - Der Dateiname der Projektplan Datei in der Gesamtabgabe ist also „**1_Projektplan.pdf**“
    - Falls Sie z.B. HTML als Format wählen und das Dokument mehrere HTML Dateien umfasst, packen Sie die Dateien in ein ZIP Archiv. Achten Sie darauf, dass das Dokument direkt aus dem ZIP Archiv durchgebrowst werden kann, d.h. das ZIP Archiv nicht auf die Festplatte entpackt werden muss.
        - Der Dateiname für die Projektplanabgabe als zip Archiv ist also **„DSE16Gruppe<nn>Projektplan.zip“**
- Buchen Sie pro Gruppe im TUWEL einen Termin für Ihr Abgabegespräch **bis spätestens 29.04.2016, 24:00**.
    - Die Abgabegespräche finden vom **20.06.2016** bis **23.06.2016** im Institutsgebäude Argentinierstr. 8 statt.
- Laden Sie die Gesamtabgabe aller Deliverables im TUWEL hoch **bis spätestens 15.06.2016, 24:00**.
    - Alle Files der Abgabe müssen in einer sinnvollen Verzeichnisstruktur abgelegt werden, die der Nummerierung der Deliverables entspricht
        - Legen Sie der Abgabe unbedingt auch wieder Deliverable 1 (Projektplan) bei.
    - Der Dateiname der Abgabe muss **DSE16Gruppe<nn>Abgabe.zip** lauten
- Ablauf des Abgabegesprächs
    - Zur Präsentation innerhalb des Abgabegesprächs ist die Anwesenheit **aller** Gruppenmitglieder erforderlich.
    - Sie präsentieren Ihre Lösung mittels
        - Kurzüberblick

        - Live Demo (Melden Sie rechtzeitig (bis spätestens 31.03.2016, wenn Sie nicht über die erforderliche Infrastruktur für die Demo verfügen!)
        - Code Walkthrough
        - Build Durchlauf mit erfolgreicher Ausführung aller Unit Tests
    - Jedes Gruppenmitglied muss die von ihm erarbeiteten Teile präsentieren
    - Jedes Gruppenmitglied muss das Gesamtprojekt - d.h. auch die Teile, die die anderen Gruppenmitglieder präsentieren – kennen, vollständig verstehen und auch Fragen dazu beantworten können.
    - Wir behalten uns vor, Gruppenmitgliedern weniger Punkte zu vergeben, wenn der Eindruck entsteht, dass diese nicht adäquat am Gesamtumfang der Übung mitgearbeitet haben.

4 Übungsinhalte

- Projektarbeit/Projektplanung
- Architektur- und Designentwurf eines verteilten Systems im Kontext von Microservices und leichtgewichtigen Runtime-Umgebungen
- Implementierung
- Test
- Deployment
- Erstellung eines Wartungshandbuchs
- Gruppenkommunikation- und Zusammenarbeit, Präsentationstechnik

5 Wichtige Anmerkungen und Tipps

- **Studieren Sie die Angabe genau!** Es werden immer wieder Deliverables vergessen bzw. Themen verfehlt. Für eine gute Note müssen alle Aufgaben erfüllt und alle Deliverables bereitgestellt werden. Wir behalten uns vor, formal nicht entsprechende Deliverables nicht zu akzeptieren.
- Verwenden sie **keine öffentlich zugänglichen** Groupware Accounts wie zum Beispiel **GitHub** oder andere öffentliche Source Code Repositories! Andere Gruppen könnten Zugriff auf Ihren Source Code bzw. Ihre Dokumente erlangen! Verwenden Sie z.B. Bitbucket [https://bitbucket.org/] Accounts oder ähnliche Angebote – hier können sie kostenlose private Arbeitsbereiche einrichten.
- Teilen Sie Rollen bzw. Zuständigkeiten innerhalb der Gruppe ein (für z.B. Projektleitung, Koordination, Technik, Tools, Dokumentation). Planen Sie Projektmeetings auch während der Laufzeit, z.B. zu bestimmten Milestones. Heben Sie nicht alles für den Schluss auf.
- Bei der Präsentation muss man über das Gesamtprojekt informiert sein – nicht nur über seinen eigenen Teil!
- Wenn Sie bei der Lösung aufgrund von unvollständigen Angaben bzw. Informationen *Annahmen* treffen, dokumentieren Sie diese unbedingt! Annahmen dürfen in keinem Fall die Lösung trivialisieren.

- Plagiate: Bitte kopieren Sie keine Lösungen anderer Gruppen oder Lösungen, die im Internet angeboten werden. Wenn wir Plagiate entdecken, vergeben wir 0 Punkte und Sie können die Übung nicht positiv absolvieren. Natürlich ist es erlaubt, Problemstellungen und mögliche Lösungen mit Ihren Kollegen zu diskutieren, aber die abgegebene Lösung muss zu 100% von Ihnen erstellt worden und einzigartig („unique“) sein. Auch wenn Gruppen gemeinsam arbeiten, dürfen die Lösungen zueinander nicht zu ähnlich sein.
- Deadline Verschiebungen: Deadline Verschiebungen werden nur in speziellen Situationen zugestanden, nachdem eine individuelle Vereinbarung mit der Kursleitung getroffen wurde. Wenn Sie Ihr Beispiel nicht rechtzeitig beenden können, geben sie Ihr Beispiel unvollendet ab – das ist immer noch besser, als gar nichts abzugeben. Achten Sie darauf, dass bei der Abgabe immer etwas schiefgehen kann und planen Sie das entsprechend ein. Geben Sie ihre Lösung rechtzeitig ab und beachten Sie, dass kurz vor der Deadline die Server überlastet sein können. Wir können keine Email Abgaben oder andere Uploads akzeptieren. Wenn Sie Probleme mit der Abgabe haben, senden sie einen Hash (z.B. MD5) Ihrer Abgabe an die Kursleitung rechtzeitig vor der Abgabe-Deadline. Mittels dieses Hashes können wir überprüfen, ob die Abgabe nach der Deadline modifiziert worden ist.

6 Anhang

Dokumenteninhalt:

Auflistung der Deliverables, Unterverzeichnisstruktur der Abgabe und Bewertungs-Informationen:

Deliverable 1: Dokument: Projektplan

5 Punkte – Abzüge bei Unvollständigkeit bzw. fehlender Plausibilität

- **1 Punkt** - Enthält eine kurze Projektbeschreibung mit den Projektzielen und Nicht-Zielen (Projektzieleplan)
- **1 Punkt** - Enthält die Rollenvergabe im Projektteam (Projektorganigramm)
  - Wer ist der Projektleiter, wer die Projektteammitglieder?
- **1 Punkt** - Enthält die Arbeitspaketspezifikation oder Work Breakdown Structure (WBS)
  - Welchen Aufwand planen Sie für die Tätigkeiten ein (Aufwandsschätzung)?
  - Wie hoch ist der Aufwand für jedes Projektteammitglied?
  - Wie hoch ist der Gesamtaufwand?
- **1 Punkt** - Enthält die Ressourcenplanung und Milestoneplanung
  - Wer im Team macht was und bis wann?
  - Festlegung der Fertigstellungstermine für die Arbeitspakete und Deliverables
- **1 Punkt** - Welche Methoden und Tools werden Sie verwenden und wozu
  - Technische Planung detailliert ausgeführt.

Deliverable 2: Dokument: Architektur- und Designdokument

5 Punkte – Abzüge bei Unvollständigkeit bzw. nicht nachvollziehbaren Designentscheidungen, die im Gegensatz zu dem in der Vorlesung vermittelten state-of-the-art stehen.

Deliverable 3: Codeabgabe: Wartelistenimplementierung für planbare OPs

45 Punkte – Abzüge bei Unvollständigkeit: fehlendes Javadoc, fehlende Inline Kommentare, fehlende Funktionalität, fehlende Kommunikation, fehlende Unabhängigkeit der Komponenten, nicht funktionierendes User Interface, unerlaubte Trivialisierung, fehlende Tests, fehlende Verwendung vorgeschriebener Frameworks.

Deliverable 4: Dokument: Wartungshandbuch

3 Punkte – Abzüge bei Unvollständigkeit gegenüber der Aufgabenstellung.

Deliverable 5: Dokument: Lessons Learned und Conclusio

2 Punkte – Abzüge bei Unvollständigkeit gegenüber der Aufgabenstellung.

Höchstpunktezahl: **60 Punkte**
Mindestpunktezahl**: 30 Punkte**

Für die Umsetzung von optionalen Anforderungen (gekennzeichnet mit dem Wort „Optional:“) werden keine Punkte vergeben. Es können damit auch keine Punkte für Optionen gegen nicht erreichte Punkte nicht-optionaler Teile „eingetauscht“ werden.

Kalender:

15.04.2016, 24:00: Deadline Abgabe Projektplan via TUWEL
29.04.2016, 24:00: Deadline für Buchung Abgabegesprächstermin
15.06.2016, 24:00: Deadline Gesamtabgabe aller Deliverables via TUWEL
20.06.2016 bis **23.06.2016:** Abgabegespräche

Dokumentenlenkung:

V1.0 Version vom 25.02.2016

Copyright:

Kontaktadresse für die Übung:

dse--AT--infosys.tuwien.ac.at