# Test Makroarchitektur, Teil 2

==========

(Schwerpunkt der Überprüfung sind Aufgaben 6-10 des Übungszettels, also eigenständiges Entwerfen/Codieren)

## Aufgabenstellung “Buchhandlung”

Es soll der Use Case einer Buchbestellung umgesetzt werden:

1. es wird eine Buchbestellung angenommen
2. danach wird der Buchdruck passend zur Bestellung durchgeführt
3. sobald der Buchdruck abgeschlossen ist, kann das Buch "abgeholt" werden.

Bitte in der folgenden Detailbeschreibung die Namen in Anführungszeichen auch genau so im Programm benennen!

Details:

- Über eine REST-API kann beim “buchhandlung-service” die Buchbestellung initiiert werden. Dazu wird ein entsprechendes “Bestellung”-Objekt angelegt und in der Datenbank gespeichert.
- Das “buchhandlung-service” übermittelt in weiterer Folge ein “BuchBestelltEvent” (dieses Event hat genau diesen Namen) an den Message Broker (idealerweise RabbitMQ Message Broker, wie bereits in Aufgabe 6 “Cargo” verwendet -> gern dort nachsehen; zur Erinnerung: auf Moodle findet ihr auch das Video zur Inbetriebnahme von Cargo mit RabbitMQ). Nach erfolgreichem Versand des Event wird am zugehörigen “Bestellung”-Objekt ein Status “BESTELLT” gesetzt.
- Das “druckerei-service” reagiert auf das “BuchBestelltEvent” (erstellt und gesendet vom “buchhandlung-service”) und führt den Buchdruck durch.
- Sobald der Buchdruck im “druckerei-service” durchgeführt wurde, übermittelt das “druckerei-service” das “BuchGedrucktEvent” über den Message Broker. Über den Message Broker wird dieser Event an entsprechende Konsumenten (in diesem Fall lediglich das “buchhandlung-service”) weiter geleitet.
- Bei Erhalt des “BuchGedrucktEvent” setzt der “buchhandlung-service” den Status am zugehörigen “Bestellung”-Objekt auf “ABHOLBEREIT”

Zusätzliche Rahmenbedingungen:

- zuerst ist die Architektur zu entwerfen; dazu sind geeignete Diagramme bzw. Handskizzen zu erstellen

- Frameworks/Laufzeitumgebung: Spring Boot Anwendung(en) auf Basis Maven
- von Beginn an mittels Git versionieren (auch gleich mit Remote GitHub/GitLab)
- Interaktion mit der Anwendung mittels REST-API (keine GUI gefordert)
- Group-ID (Maven): “at.itkollegimst.<nachname>.pos1makro.test2”
- Artefact-ID (Maven): “buchhandlung” (= Name der Anwendung)
- Java Package (Basis): Group-ID + “.” + Artefact-ID
  (unterhalb der Basis dann die Paketnamen entsprechend eurer Architekturentscheidungen sinnvoll wählen!)

Abgaben:

- Die Erarbeitung des Beispiels erstreckt sich über die Einheiten am 28.4. und am 5.5.
- es ist in jeder Einheit eine Abgabe zu machen, spätestens Ende der Einheit am 5.5. muss das gesamte Projekt abgegeben sein! (also wichtig laufend Commits in Git machen und Clone-Link in Moodle abgeben)
- bereits vor Ende Einheit 28.4.: wesentliche Architektur- / Design-Überlegungen bereits VOR Beginn des Codierens erstellen und abgeben in Form eines PDF (geeignete Diagramme / Skizzen verwenden)
  -> Updates im Zuge der Entwicklung können dann in der zweiten Abgabe eingearbeitet werden
  -> im Projekt ein Unterverzeichnis “unterlagen” anlegen und PDF dort abspeichern
  - wesentlich: Begründungen für eure Entscheidungen anführen!
- Zwischenstand zum Ende der Einheit am 28.4. und fertiges Projekt vor Ende der Einheit am 5.5.:
  das Programm (bzw. die Programme, je nach eurer Entscheidung bezüglich Architekturstil)
- Code ist von Beginn an auf GitHub (oder GitLab) zu versionieren
- Abgaben erfolgen in Form eines GitHub / GitLab Clone-Link zu diesem Repository auf Moodle (zugehörige Abgabe: Überprüfung Makroarchitektur, Teil 2)
- Postman Testcases zum Aufruf der REST-Schnittstelle: zumindest für “Buch bestellen” und “Bestellstatus abfragen”
  -> Postman Collections gleich mit beim Projekt in Git versionieren und damit abgeben
- Screenshot einer erfolgreichen (Buch gedruck / abholbereit) Statusabfrage in Postman
  -> im Projekt ein Unterverzeichnis “unterlagen” anlegen und Screenshot dort abspeichern und versionieren (muss zu Git hinzugefügt werden!) -> damit automatisch abgegeben...