Hinweise zur Assessment Übungsaufgabe am 10.09.2020

Die folgenden Zeilen entstammen aus einem Scrum-Entwicklertagebuch der Firma Capgemini. Alle hier auftretenden Personen oder Namen sind aus Datenschutzgründen verfälscht werden. Außerdem weist der Autor darauf hin, dass die folgende Situation rein aus fiktiven Vermutungen erstellt worden ist.

In den folgenden Zeilen ist der zeitliche Ablauf einer Feature Entwicklung eines neu eingestellten *Junior Developers* der Firma skizziert. Die Firma hat sich überlegt die Personalverwaltung auf eine Webapplikation auszulagern, wobei die Personaldaten (PersonalID, Name und Standort) zentral auf einer relationalen Datenbank gelagert werden sollen. Der *Junior Developer* hat die Aufgabe bekommen im Rahmen der Entwicklung ein Feature für die Verwaltung der Personaldaten mit *SpringBoot* zu entwickeln. Das Feature hat die folgenden Eigenschaften:

1. Personaldaten können aus der Datenbank mittels http-Protokoll abgefragt werden.
2. Personaldaten können in die Datenbank über http hinzugefügt werden.

Die entscheidende Phase der Testung und der Inbetriebnahme muss der *Junior Developer* in Zusammenarbeit mit einem *DevOps* (deine Rolle) durchführen.

Timeline:

Zeitstempel	
9:37	Der Junior übergibt dem *DevOps* seine Applikation. Geschockt von den vielen Fehler, kippt sich der *DevOps* seinen Kaffee über die Hose. Die Anzahl der Kompilierungsfehler ist nicht schön und die Lösung erscheint zeitintensiv. Folgende Aufgabe gibt es zu tun: 1. Beseitige alle Kompilierungsfehler 2. Schreibe die Datei /src/main/ressources/application.properties in eine yaml Datei um 3. Führe deine Springboot-Applikation aus 4. Teste die Datenanbindung mit Postman, in dem du deinen Namen aus der SQL-Datei löschst und mitttels einer Post-Methode wieder einfügst
12:24	*DevOps* guckt genervt auf seine Uhr – der Magen knurrt – es wird noch ein Container gebraucht. Der Junior fragt sich, was Schiffscontainer mit Softwareentwicklung zu tun haben? – Der *Devops* kann nur die Augen verdrehen: 1. Ändere deine pom.xml Datei ab, damit Maven in seinem Lebenszyklus ein Docker-Image baut. 2. Schreibe einen passenden Dockerfile

14:42 Der *Junior* ist mit seiner Arbeit zufrieden und jubelt, dass mit dem Dockerfile endlich das gesamte Setup gestartet werden kann. Der *DevoOps* betont die notwendige Automatisierung und die Notwendigkeit Arbeit auszulagern:

1. Schreibe eine Docker-Composer im YAML Format und teste die Applikation erneut.

16:17 *Devops* schaut auf die Uhr fast Feierabend nur noch die Pipeline fehlt. Wieder ein nicht notwendiger Einwand seitens des *DevOps*, was denn Kraftstoffe mit Softwareentwicklung zu tun hätten? Der *DevOps* reisst sich zusammen, dennoch pulsiert die Ader auf seiner Stirn deutlich. Sichtlich genervt schickt er den Junior Pizza holen:

1. Push deine Applikation auf dein Remote Repository.
2. Setze eine Jenkins CI-Pipeline auf und achte auf die Antwort auf der GitHub Webseite.

17:13 Der *Junior Developer* steht vor einer verschlossenen Tür mit zwei halbkalten Pizzen und denkt sich – „*DevOps* müsste man sein…“