Projektantrag

[Prüfung – G3T7]

Inhaltsverzeichnis

1. Projektbezeichnung

Software zur Administration sowie Simulation von Produktionsstraßen für die Volkswagen AG

1.1 Kurzform der Aufgabenstellung

Zur Verwaltung und Simulation von Produktionsstraßen soll Mitarbeitern der Volkswagen AG die Möglichkeit gegeben werden, Produktionsstraßen zu erstellen und zu simulieren. Ziel einer jeden Produktionsstraße ist dabei die Fertigstellung eines beliebigen Autos der Volkswagen AG. Dabei soll einer jeweiligen Produktionsstraße eine Untermenge von Akteuren (Roboter, Stationen, Mitarbeiter) zugeordnet werden können. Durch das Starten einer Produktionsstraße produziert diese eine bestimmte Menge Autos pro Zeiteinheit.

2. Zielsetzung des Projekts/Soll-Konzept

Benutzeroberfläche
Ein lokal ausgeführter JavaFX-Client bzw. ein in Azure-gehosteter React-Client repräsentiert das Frontend einer Software zur Administration sowie Simulation von Produktionsstraßen. Auf einer Landing-Page erhält der Benutzer eine Übersicht über bereits angelegte Produktionsstraßen und via Filter kann nach passenden Produktionsstraßen gesucht werden. Es können neue Produktionsstraßen angelegt bzw. bereits existierende Produktionsstraßen editiert werden.

Datenhaltung
Die Daten zur Software sollen in einer passenden Datenbank in Azure abgelegt werden. Die Wahl der konkreten Datenbank ist in der Dokumentation entsprechend begründet. Die jeweiligen Entitäten (Produktionsstraßen, Roboter, Stationen, Mitarbeiter) sowie jeweils assoziierte Attribute gilt es zu identifizieren und – nach Möglichkeit – logisch zu verknüpfen.

Aufbau der Softwarearchitektur
Die Software soll in einer wartbaren und verständlichen Architektur aufgebaut werden, die sich, sofern möglich, am MVC-Modell und einer Client-Server-Architektur mit **Spring Boot** im Backend orientiert. Die Entscheidung für eine konkrete Architektur ist begründet und in der Dokumentation der Software beschrieben.

Die Gesamtheit aller Services (Datenbank, Frontend (ausgenommen JavaFX[1]), Backend, etc.) gilt es als geeigneten *Azure Service* zu deployen. Für die Entwicklungsphase können die zu speichernden Daten in einer *In-Memory H2*-Datenbank oder via Docker-Container mit DB-Image persistiert werden. Der Server kann lokal gestartet werden. Die jeweilige Konfiguration sollte in Git hinterlegt und in der *readme.md*-Datei berücksichtigt werden.

2.1 Was soll am Ende des Projektes erreicht werden?

Am Ende des Projektes soll eine auslieferbare Software entstehen. In jedem Fall muss die Ausführung der Software dokumentiert sein (readme.md). Sie selbst verstehen sich dabei als Full-Service-Provider. Dies bedeutet, dass Sie nicht nur die Anwendung in allen Facetten entwickeln, sondern ebenfalls das Hosting (sowie etwaige Wartungsarbeiten) übernehmen.

Der Client soll sich an fiktiven Daten orientieren. Diese sollen in der Datenbank der deployten Version hinterlegt und für zukünftige Entwicklungseinsätze (bspw. für das Aufsetzen eines neuen, lokalen Workspaces) verfügbar sein. Der Client kann sich optisch an den bisherigen, in der Volkswagen AG bestehenden, Systemen orientieren.

2.2 Welche Anforderungen an die Software müssen erfüllt sein?

Funktionale Anforderungen

- **Fachliche Anforderungen**
  - Die Anwendung dient der Administration und Simulation von Produktionsstraßen sowie den damit assoziierten Robotern, Stationen und Mitarbeitern.
  - Eine Produktionsstraße beinhaltet zugewiesene Produktionsschritte (Roboter, Stationen und Mitarbeiter) in einer festgelegten Reihenfolge.
  - Die Anwender:innen können neue Produktionsstraßen anlegen.
    - Eine Produktionsstraße hat einen Status (lauffähig, unvollständig) und einen Simulationsstatus (läuft, läuft nicht).

[1] Da sich ein JavaFX-Frontend nur schwer via Azure App Service bereitstellen lässt, fällt hierfür eine weitere funktionale Anforderung an.

- Eine Produktionsstraße besitzt einen Namen.
- Jede Produktionsstraße produziert genau ein Automodell. Sofern eine Produktionsstraße nicht läuft, kann das jeweilige Automodell geändert werden, sodass die ausgewählte Straße nun ein anderes Fahrzeug produziert.
    - Jedes Automodell verfügt über einen jeweiligen Faktor ([0.5, 1.5]), der sich auf jeden einzelnen Produktionsschritt auswirkt, sodass gewisse Automodelle schneller produziert werden können als andere.
- Einer Produktionsstraße können beliebig viele Roboter zugeordnet werden.
    - Ein Roboter besitzt einen Namen und eine Fertigungsdauer für den jeweiligen Produktionsschritt.
- Einer Produktionsstraße können beliebig viele Stationen zugeordnet werden.
    - Eine Station repräsentiert einen manuellen Arbeitsschritt bei der Produktion eines Autos. Somit hat eine Station einen Namen und eine feste Dauer für diesen Arbeitsschritt.
        - Einer Station kann eine beliebige Anzahl von Mitarbeitern zugeordnet werden. Mitarbeiter verfügen lediglich über einen Namen.
- Für die Zuordnung von Produktionsstraßen zu Robotern sowie zu Stationen und deren Mitarbeiter:innen gilt es, eine geeignete Visualisierung zu schaffen.
- **Zusatz für JavaFX-Frontend:**
  Ein Roboter besitzt zusätzlich eine Lebensdauer.
    - Wenn die Lebensdauer eines Roboters abgelaufen ist, kann dieser gewartet werden und ist wieder voll einsatzfähig.
    - Sofern die Lebensdauer eines Roboters abgelaufen ist, ist eine Produktionsstraße nicht lauffähig und wird pausiert.
- **Zusatz für 3er-Gruppen:**
  Bereits bestehende Produktionsstraßen können editiert werden. Dabei können bspw. zugewiesene Produktionsschritte aus der Produktionsstraße entfernt werden, was unmittelbar die Indexe der nachfolgenden Produktionsschritte beeinflusst und sinnig behandelt werden muss.
    - Eine Produktionsstraße kann nur dann editiert werden, wenn sie vorher gestoppt wurde.

- Eine Produktionsstraße kann gestoppt und gestartet (oder pausiert[2]) werden. Die Transitionen zwischen diesen Zuständen sind sinnig zu wählen.
    - Eine Produktionsstraße ist nur dann lauffähig, wenn sie mindestens drei Produktionsschritte besitzt und alle notwendigen Attribute definiert sind.
        - Sofern einer Station nicht mind. ein/e Mitarbeiter:in zugeordnet ist, ist die Produktionsstraße nicht lauffähig.

[2] Zusatzanforderung für JavaFX (siehe Fußnote 1)

        - Die Produktionsdauer, die eine Produktionsstraße benötigt, um ein einzelnes Auto zu produzieren, ist abhängig vom jeweiligen Faktor des Autos sowie der Gesamtdauer der einzelnen Arbeitsschritte.
            - Nach Ablauf einer bestimmten Zeiteinheit produziert eine Produktionsstraße ein Auto. Die Simulation kann innerhalb eines sinnvollen Bereichs beschleunigt bzw. entschleunigt werden.
        - Für eine Produktionsstraße kann für eine abgeschlossene (oder eine pausierte[3]) Simulation eingesehen werden, wie viele Autos von welchem Typ bisher produziert wurden.
            - Alte Simulations- bzw. Produktionsergebnisse für eine Produktionsstraße müssen nicht persistiert werden und können verworfen werden. Es soll nur die jeweils letzte Simulation einer Produktionsstraße persistiert werden.
            - Sofern eine Produktionsstraße nicht lauffähig ist, lässt sich diese nicht starten.
    - Es können beliebig viele verschiedene Produktionsstraßen simuliert werden. Diese agieren völlig unabhängig voneinander.
    - Eingabefelder (wie bspw. Datumseingaben) sind mit einer sinnvollen Validierung zu versehen. Diese gilt es je nach Use-Case zu identifizieren.
- **Technische Anforderungen**
    - Die Simulation einer Produktionsstraße wird serverseitig ausgeführt. Somit laufen Simulationen auch dann, wenn kein/e Anwender:in das Frontend geöffnet hat.
    - Der Source Code wird in dem Versionskontrollsystem Git[4] gepflegt.
    - Der Source Code ist klar strukturiert.
    - Der Source Code ist gut lesbar, konsistent und entspricht den vorgegebenen Code Conventions.
    - Die wichtigsten Funktionalitäten sind durch sinnvolle Testfälle abgedeckt.
    - Verwendung einer Icon-Bibliothek, wie bspw. Font Awesome.

2.3 Welche Einschränkungen müssen berücksichtigt werden?

Die Software soll in verschiedenen Umgebungen ausführbar sein. Relevante Konfigurationen wie bspw. Datenbankparameter müssen also über eine Konfiguration (bspw. Springs *application.properties*) veränderbar sein.

2.4 Weitere Ergänzungen

Dieser Projektantrag wurde durch einen Kunden erstellt. Dadurch könnten Anforderungen unklar oder missverständlich formuliert sein. Sollten daher in der Prüfung Fragen aufgrund von unklaren oder missverständlich formulierten Anforderungen aufkommen, müssen diese mit dem Kunden (s. 4. Namen der Projektverantwortlichen & Ansprechpartner:innen) unverzüglich geklärt werden.

3. Projektstrukturplan

Das Projekt soll nach agilen Prinzipien entwickelt werden. Passende Anforderungen sind zu Beginn in Form von **User Stories zu formulieren und zu dokumentieren**. Bei etwaigen Fragen

[3] Zusatzanforderung für JavaFX (siehe Fußnote 1)
[4] Es muss das zuvor eingerichtete Repository unter www.devstack.vwgroup.com/bitbucket verwendet werden.

fungiert Ihr Ansprechpartner als Kunde. Da das Projekt in einem kurzen Zeitraum abläuft, wird jedoch auf mehrere Entwicklungsiterationen verzichtet.

Das gesamte Projekt wird in Bitbucket gepflegt und Arbeitsergebnisse werden mindestens täglich, am besten mehrmals, in Form von Commits, bereitgestellt. Diese Commits sollen sinnvolle, kleinteilige Zwischenergebnisse zusammenfassen und gut kommentiert sein, um Nachvollziehbarkeit zu gewährleisten.

3.1 Aufgabenauflistung

Analyse

- Erstellung von User Stories für gesammelte Anforderungen (mind. drei Anforderungen sollten vollständig erfasst sein; der Rest kann stichpunktartig vorliegen)
- Aufarbeitung des Konzepts eines Datenmodells anhand der Anforderungen
- Entwicklung eines Konzepts zur Umsetzung der Konfigurationsregeln von Produktionsstraßen

Entwurf

- Grobentwurf der Oberflächen z.B. mit Mockups
- Grobentwurf der Softwarearchitektur
- Datenmodell mit einer Modellierungsmethode skizzieren (bspw. ER-Modell)

Implementierung

- Implementierung der Software
- Absichern der Funktionalität durch Tests
- Abschließende Funktionstests durch den/die Entwickler:in

Dokumentation

- Dokumentation der Analyseergebnisse und der Designs inkl. der Designentscheidungen
- Dokumentation der Tests und der Entscheidungen für die entsprechenden Testfälle
- Die Dokumentation sollte angemessen formuliert sein. Achten Sie auf die Verwendung korrekter Fachtermini, Ausdrucksweise, etc.
- Retrospektive
    - Vergleich Ist- und Soll-Zustand
    - Einschätzung der eigenen Zielerreichung

Präsentation

- Präsentation des Endergebnisses und der wichtigsten Aspekte der Dokumentation (Gruppenpräsentation[5])
- Vorstellen von Funktionen inkl. deren Codestellen, die vom Prüfteam ausgewählt werden (je Fachgespräch)

3.2 Anforderungen an weitere Ergebnisse des Projekts

Die Dokumentation sollte 8-10 Seiten (ohne Deckblatt und Verzeichnisse) umfassen und die Ergebnisse der Software beschreiben. Der Arbeitsauftrag ist genauer beschrieben, inklusive formulierter Anforderungen. Technische sowie fachliche Entscheidungen und Aspekte sollen nachvollziehbar beleuchtet werden. Auch die Architektur der Software soll mit geeigneten Mitteln, bspw. UML, beschrieben werden.

[5] Die Gruppenpräsentation soll nicht ausschließlich aus einem Medium (z.B. PowerPoint), sondern aus mehreren Elementen für ein Kundengespräch (z.B. ein Review) bestehen.

Auf der obersten Ebene des Projekts liegt eine ***readme.md*-Datei**, welche kurz beschreibt, wie das Projekt aufgebaut und zu starten ist.

3.3 Abgabe

Das Projekt muss inkl. der Dokumentation (lediglich als .pdf-Datei) am Freitag, den 03.02.2023 bis spätestens um 17 Uhr via Git abgegeben sowie in Azure vollständig deployt werden. Spätere Abgaben werden nicht berücksichtigt und mit 0% bewertet.

4. Namen der Projektverantwortlichen & Ansprechpartner:innen

- Maximilian Czerlitzki
- Sarah Pagliardini
- Dominik van der Hoeven