



Abschlussprüfung

Fullstack

Datum: 22.01.2021 – 03.02.2021

Teilnehmercodes: _____

Auto Daten List Informationen System (ADLIS)

Inhaltsverzeichnis

Vorwort	2
Aufgabe 1	3
1. Produkteinsatz	3
2. Produktübersicht	3
3. Produktfunktionen	4
5. Produktleistungen	7
6. Qualitätsanforderungen	7
7. Ergänzungen	7
8. Anforderungen an die Dokumentation (Prio 1)	8
9. Betrieb (Prio 1)	8
10. Anhang	9

Vorwort

Das moderne Auto generiert Unmengen an Daten, die für unterschiedliche Zwecke genutzt werden können.

Aufgabe 1

1. Produkteinsatz

Es ist ein Softwareprodukt namens ADLIS zu entwickeln, das das Sammeln der Daten im Auto unterstützt. Vehicle Owners, Werkstätten und Service Center können via Webschnittstelle und/oder mobiler Applikationen auf die Daten zugreifen. Die Anforderungen (Lastenheft, LF) sind im Abschnitt Produktfunktionen aufgelistet und hinsichtlich ihrer Relevanz priorisiert:

Prio 1: Anforderung muss realisiert werden.

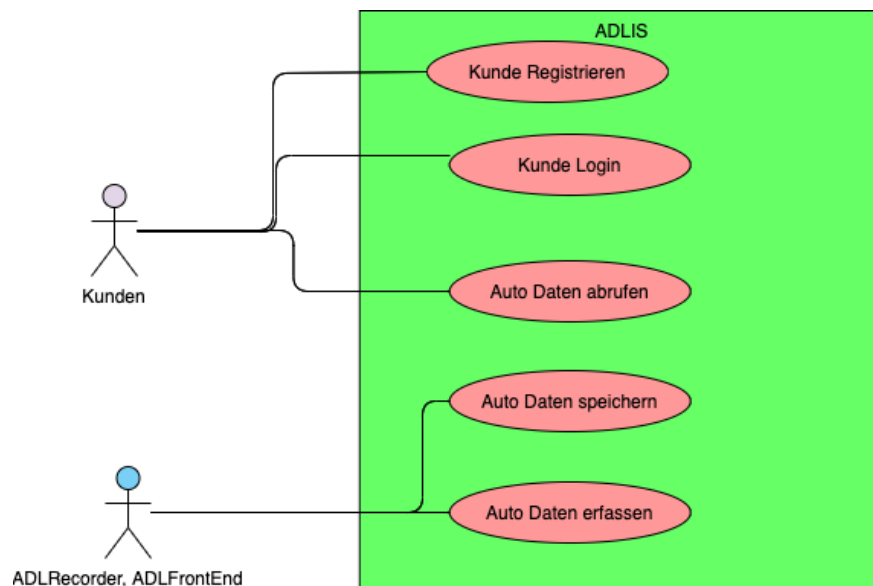
Prio 2: Anforderung soll realisiert werden, falls diese im vorgegebenen Zeitraum möglich ist.

Prio 3: Im Optimalfall kann die Anforderung realisiert werden.

Das Produkt besteht aus mehreren Komponenten. Jede davon muss containerisiert und im Kubernetes deployed werden (siehe Abschnitt Betrieb).

2. Produktübersicht

Das System soll folgende Anwendungsfälle unterstützen:



3. Produktfunktionen

/LF10/

Prio 1

Komponent:	ADLRecorder
Anwendungsfall:	Auto Daten erfassen
Akteur:	Time Trigger
Beschreibung:	Während der Fahrt sollen ständig Auto-Daten (siehe Anhang) erfasst und in die Cloud übertragen werden. Dafür soll ein "ADLRecorder" entwickelt werden, der alle zwei Minuten Daten erfasst und in die Cloud überträgt.

In der Praxis würde dies über die Installation eines mobilfunkfähigen Geräts im Fahrzeug realisiert werden, das die Datenerfassung übernimmt.

Im vorliegenden Fall soll ein "ADLRecorder" entwickelt werden, der alle zwei Minuten Daten aus einem konfigurierten JSON/YAML File liest, dessen Initial-Werte mit zufälligen Werten ersetzt, und an das "ADLBackEnd" via HTTP (siehe LF20) sendet. Diese Komponente ist mit Java SE (Standard Edition, kein weiteres Framework) oder einer geeigneten alternativen Programmiersprache zu entwickeln. Eine alternative Programmiersprache sprechen Sie bitte vorher mit ihrem Betreuer ab. Der "ADLBackEnd" Service Name muss konfigurierbar sein.

/LF20/

Prio 1

Komponent:	ADLBackEnd
Anwendungsfall:	Auto Daten speichern
Akteur:	ADLRecorder, ADLFrontEnd
Beschreibung:	Die von der ADL Recorder erfassten Auto Daten sollen in der Cloud gespeichert werden.

Entwickeln Sie einen Rest/HTTP-API Microservice, der die Möglichkeit anbietet, Daten in einer NOSQL Datenbank (Mongo DB, ADLISDB) zu speichern und zu lesen.

Beschreiben sie zuerst jedoch die Schnittstelle Ihrer API mit OpenApi/Swagger und achten Sie auf folgendes:

basePath: /adl-api/v1

post: Speichen neuer ADL Datenpaket

- operationID: ~~saveADL~~
- responses:
 - 201: Successful
 - 500: Server Error
- Sicherheit
 - ~~adl basicAuth~~

get: Lesen der ADL Datenpaket

- operationID: ~~retrieveADLByFin~~
- responses:
 - 200: Successful
 - 500: Server Error
- Sicherheit
 - adl basicAuth

Diese Komponente soll mit Spring Boot entwickelt werden.

/LF21/

Prio 2

Komponent:	ADLBackEnd
Anwendungsfall:	ADL BackEnd per Basic Authentication schützen
Akteur:	n/a
Beschreibung:	Das Rest API ist per Basic Authentication zu schützen.

Benutzen Sie folgende Namen und Passwort:

USERNAME: ~~tu_ADLBackEnd_prod~~
PASSWORD: ~~tu ChangeMe!~~

Der Administrator muss ohne Änderungen am Code Username und Passwort ändern können.

/LF30/	
Prio 1	
Komponent:	ADLFrontEnd
Anwendungsfall:	Auto-Daten anzeigen
Akteur:	Vehicle Owners, Werkstätten, Service Center
Beschreibung:	Kunden können sich über die Webseite einloggen und die Auto-Daten abrufen.

Entwickeln Sie das Web User Interface. Die HTML-Ausgabe sollte mit Thymeleaf oder einer geeigneten Alternative erzeugt werden. Als Default sind die jüngsten Daten anzuzeigen.

Variation für Login:

AccountService ist noch nicht implementiert: Zugangsdaten können zuerst fest im Quelltext hinterlegt sein. Verwenden Sie folgende Namen und Passwort:

USERNAME: WVWZZZ1KZDP045466
PASSWORD: rootPass123!

AccountService ist implementiert: Nutzen Sie den Service entsprechend (LF50).

/LF31/	
Prio 2	
Komponent:	ADLFrontEnd
Anwendungsfall:	Auto-Daten je nach Zeitspanne anzeigen
Akteur:	Vehicle Owners, Werkstätten, Service Center
Beschreibung:	Das UI soll die Möglichkeit bieten, Daten der letzten 2, 4, 10 Minuten anzuzeigen.

/LF40/	
Prio 3	
Komponent:	Account
Anwendungsfall:	Kunden registrieren
Akteur:	Vehicle Owners, Werkstätten, Service Center
Beschreibung:	Kunden müssen sich registrieren können. Eingehende Informationen sind FIN (Fahrzeug-Identifizierungsnummer) und ein Passwort. Benutzen Sie für diese Daten eine PostgreSQL-Datenbank.

/LF50/

Prio 3

Komponent:	Account
Anwendungsfall:	Login
Voraussetzung:	LF40
Akteur:	Vehicle Owners, Werkstätten, Service Center
Beschreibung:	Nutzen Sie nun den Account Service, um die Kunden einzuloggen..

4. Produktdaten

“Nicht anwendbar”

5. Produktleistungen

“Nicht anwendbar”

6. Qualitätsanforderungen

/LQ10/


Clean code: Achten Sie auf Clean Code Prinzipien wie:

- Single Responsibility
- Testbarkeit
- Geeignete Exceptions

/LQ20/

Benutzerfreundlichkeit: gut (benutzbar)

7. Ergänzungen

- Alle Aktivitäten müssen ~~geloggt werden~~. Eine Logging Library ist zu benutzen.
 - Schreiben Sie einen Readme der ~~kurz beschreibt~~, wie Ihre Software zu benutzen ist.
 - Source Code und Kubernetes Manifest sind im Repository einzuchecken.
- 

8. Anforderungen an die Dokumentation (Prio 1)

Die Projektdokumentation muss folgende Anforderungen erfüllen:

- Umfang: ~~mind. 8 Seiten / max. 10 Seiten~~
- OAUTH im Bezug mit JWT(Json Web Token) Konzept für "Authentication" und "Authorization"
- Skizze der Architektur inklusive OAUTH Komponenten

9. Betrieb (Prio 1)

Das Produkt wird in einem Kubernetes Cluster betrieben. Einen Cluster mit zugehörigen Anmeldeinformationen sowie einer "Docker Registry" werden Ihnen zu Beginn zur Verfügung gestellt. Jede Komponente muss containerisiert und im Kubernetes deployed werden.

Sowohl das Dockerfile als auch das Kubernetes Manifest und der Code der Komponenten sind im Repository einzuchecken.

Für das Deployment der Komponente ist folgendes zu beachten:

ADLRecorder

- ~~Job Name: adlrecorder~~

ADLBackEnd

- Deployment Name: ~~adlbackend~~
- Service Name: ~~adlbackendService~~

ADLFrontEnd

- Deployment Name: adlfrontend
- Service Name: adlfrontendService

Account

- Deployment Name: account
- Service Name: AccountService

Zu jeder Komponente ist ein entsprechendes Docker Image in die Docker Registry zu pushen.

10. Anhang

Folgende Daten werden im Auto erfasst:

- FIN
- GPS-Position
- Zahl der elektromotorischen Gurtstraffungen
- Betriebsstunden der Fahrzeugbeleuchtung
- Kilometerstand
- Tankfüllung
- Reifendruck
- Füllstand Kühlmittel
- Füllstand Bremsflüssigkeit
- Füllstand Wischwasser
- Gefahrene Kilometer Autobahn
- Gefahrene Kilometer Landstraße
- Gefahrene Kilometer Stadt
- Temperatur
- Zahl der Verstellvorgänge des elektrischen Fahrersitzes
- Anzahl der eingelegten Medien des CD-/DVD-Laufwerks
- Lade- und Entladezyklen der Starterbatterie

11. Ergänzungen / Anpassungen für die 3er Gruppe

Anforderungen an den Code

- /LF40/ und /LF50/ werden zu Prio 1 (statt Prio 3)
- Logging wird zu Prio 2 (statt Prio 3)
- Mehr Testfälle erforderlich (mehr)
- Zusätzlich in Prio 3: Erstellung von Integrationstests (neu)

Anforderung an die Dokumentation

- Zusätzlich in Prio 1: Betriebskonzept (neu)
 - Beschreiben Sie die Inbetriebnahme der Anwendung.
 - Beschreiben Sie die Betriebsumgebung, insbesondere deren Stärken und Schwächen.
 - Beschreiben und bewerten Sie Ihr Vorgehen beim Deployment im Kontext von Continuous Delivery.
- Umfang: 10-12 Seiten (statt 8-10)