INF202 : Autohaus Service Projekt

Test Cases

Türkisch-Deutsche Universität
Dipl.- Ing.Ömer KARACAN

Java Benders

Melih Eyüboğlu (190503020) : e190503020@stud.tau.edu.tr
Berkan Türkoğlu (190501030) : e190501030@stud.tau.edu.tr

Gliederung

- Systemüberblick
- Systemtestfälle
  - Systemtest
  - System/Komponent-Integrationtest
- Rückverfolgbarkeit

Systemüberblick

Das zu testende System ist eine Spring Boot-Backend-Applikation. Es handelt sich um eine Applikation, die den lokalen Zugriff auf das System ermöglicht und wir bestimmte Datenbankoperationen durchführen können.

Es gibt 3 verschiedene Entitäten in der Applikation. Dies sind Auto, Kunde und Mitarbeiter. In der Applikation funktionieren CRUD-Dienste wie GET, POST, PUT und DELETE für jede dieser Entitäten.

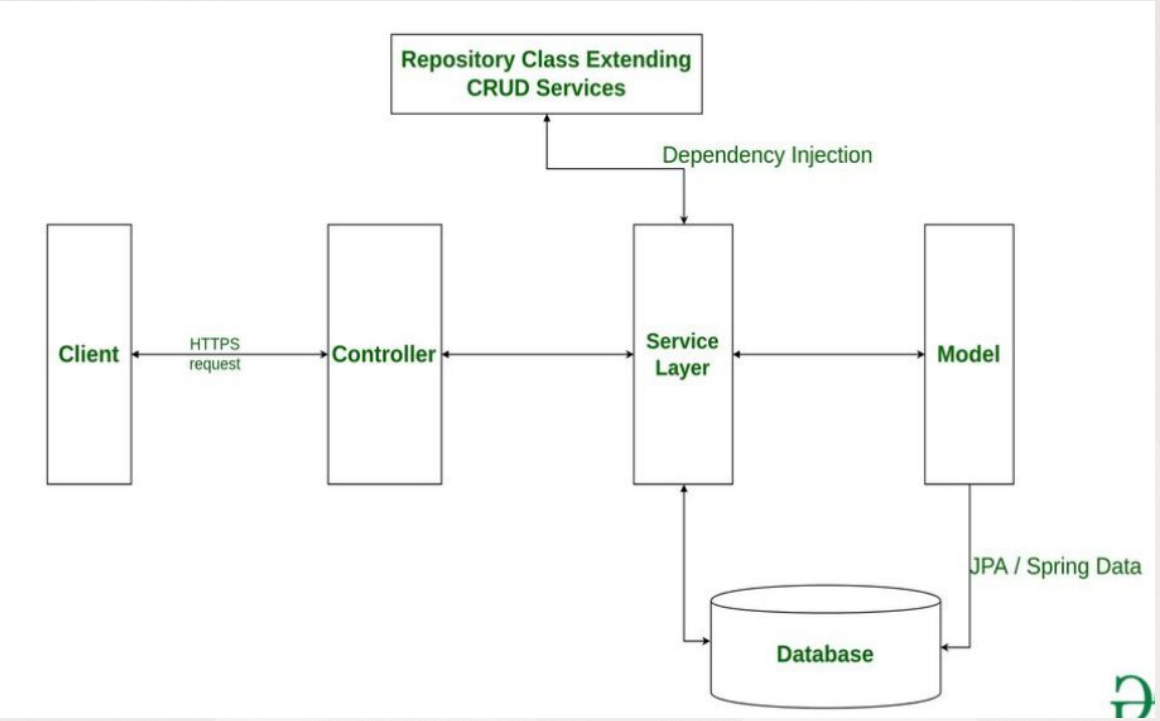


Abb. 1: Systemarchitektur für Spring Boot Application

Systemüberblick

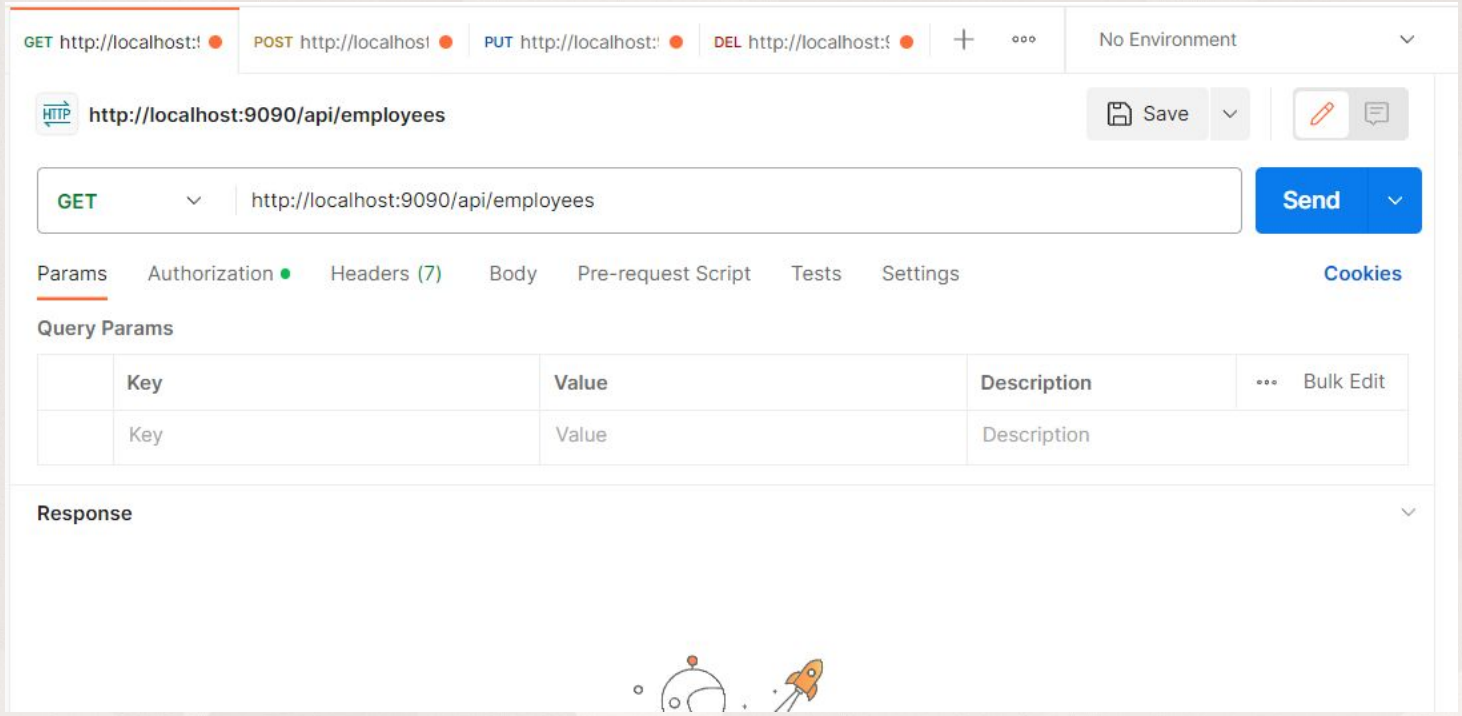
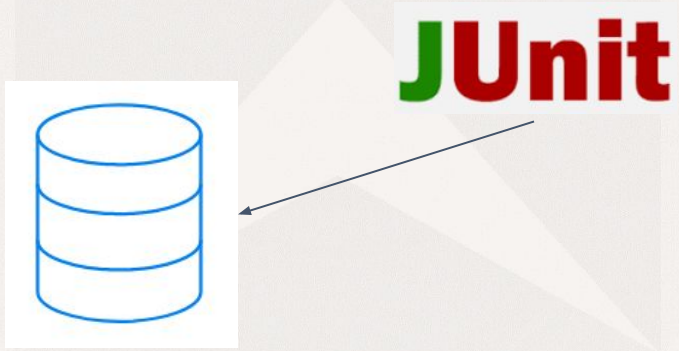


Abb. 2: Ansicht der Postman-Homepage, Implementierung des GET-Dienstes

Systemtestfälle werden auf der externen Anwendung Postman getestet. CRUD-Dienste (GET, POST, PUT, DELETE) sind in den Testfällen des Systems enthalten. Diese werden auf Postman implementiert und die erwarteten Antworten werden mit den erhaltenen Antworten verglichen.

Die Tests im Zusammenhang mit der Datenbankintegration werden mit dem JUNIT Test Framework durchgeführt. Diese Datenbanktests werden durch Anwendung des „Request-Response-Protokolls“ durchgeführt.



Systemtestfälle ➔ Systemtest

Systemtestfälle werden gemäß den zuvor in der Pflichtenheft- und Achitekturdesign-Dokumentation festgelegten Systemanforderungen entworfen. In den Systemanforderungen gibt es niemanden ohne Systemtestfall.

Es gibt 3 Entitäten in dieser App. Für jede dieser Einheiten werden Systemtests durchgeführt. In der Systemtests-Anzeige wird nur die Entität Auto angezeigt. Allerdings werden auch die beiden anderen Entitäten während der Testphase evaluiert.

Systemtests werden für CRUD-Dienste in einer Backend-Anwendung durchgeführt. Es wird überprüft, ob die Dienste korrekt funktionieren oder nicht. Diese Dienstleistungen werden für alle zuvor genannten Unternehmen erbracht. Mit anderen Worten: Für jede Entität werden die Dienste GET, POST, PUT und DELETE getestet.

➔ Die nächsten beiden Folienseiten zeigen die CRUD-Systemtestfälle für das Auto.

Systemtestfälle → Systemtest

System Test Case ID Name	S_TC_1 Test liefert alle Autos mit GET-getAuto()
Preconditions	Anfrage: GET getAllAuto() Antwort: Liste aller Autos • Data: "Auto" ◦ int AutoId //Unique {generiert} ◦ String Marke ◦ String Model ◦ int Jahr ◦ String Zustand App port number: 9090 Das System ist aktiv
Test Steps	1) Führen Sie das System auf der IDE aus und verwenden Sie den Postmande GET-Dienst 2) Überprüfen Sie die generierte Antwort in der Postman-Antwort.
Post-Condition	keiner
Test Data	keiner

Expected Result	{
	Marke: "BMW"
	Model: "320"
	Jahr: "2020"
	Zustand: "Verkauft"
	}
	• Es liegt wie oben gezeigt im JSON-Format vor und alle Autos sollten als solche aufgeführt sein.

Actual Result	{
	Marke: "BMW"
	Model: "320"
	Jahr: "2020"
	Zustand: "Verkauft"
	}
	• Liste aller Werkzeuge wie oben gezeigt

Verdict (Pass/Fail)	PASS
Verified UC & Req. IDs	/BFK-4/ , /ZR-6/ , /ZR-7/ , /BFK-3/

System Test Case ID Name	S_TC_2 Test liefert suchende Auto mit GET-getAutoById(int AutoId)
Preconditions	Anfrage: GET getAllAutoById(int AutoId) Antwort: Suchende Auto • Data: "Auto" ◦ int AutoId //Unique {generiert} ◦ String Marke ◦ String Model ◦ int Jahr ◦ String Zustand App port number: 9090 Das System ist aktiv
Test Steps	1) Führen Sie das System auf der IDE aus und verwenden Sie den Postmande GET-Dienst 2) Überprüfen Sie die generierte Antwort in der Postman-Antwort.
Post-Condition	keiner
Test Data	keiner

Expected Result	{
	Marke: "BMW"
	Model: "320"
	Jahr: "2020"
	Zustand: "Verkauft"
	}
	• Es liegt wie oben gezeigt im JSON-Format vor und suchende Auto sollten als solche aufgeführt sein.

Actual Result	{
	Marke: "BMW"
	Model: "320"
	Jahr: "2020"
	Zustand: "Verkauft"
	}
	• Suchende Auto soll so zuruckgegeben

Verdict (Pass/Fail)	PASS
Verified UC & Req. IDs	/BFK-4/ , /ZR-6/ , /ZR-7/ , /BFK-3/

Systemtestfälle → Systemtest

System Test Case ID Name	S_TC_3 Test speichert eingegebene Fahrzeuge POST/PUT-save(Auto auto)
Preconditions	Anfrage: POST/PUT save(Auto auto) Antwort: Informationen zum zugelassenen Fahrzeug • Data: "Auto" ○ int AutoId //Unique {generiert} ○ String Marke ○ String Model ○ int Jahr ○ String Zustand App port number: 9090 Das System ist aktiv
Test Steps	1) Führen Sie das System auf der IDE aus und verwenden Sie den Postmande POST/PUT-Dienst 2) Überprüfen Sie die generierte Antwort in der Postman-Antwort.
Post-Condition	keiner
Test Data	keiner

Expected Result	{
	Marke: "BMW"
	Model: "320"
	Jahr: "2020"
	Zustand: "Verkauft"
	}
	• Fahrzeuginformationen mit Aktualisierungs- oder Erstellungsvorgängen sollten wie oben zurückgegeben werden..

Actual Result	{
	Marke: "BMW"
	Model: "320"
	Jahr: "2020"
	Zustand: "Verkauft"
	}
	• Fahrzeug mit Aktualisierungs- oder Erstellungsvorgängen

Verdict (Pass/Fail)	PASS
Verified UC & Req. IDs	/ZR-4/ , /ZR-5/ , /ZR-7/

System Test Case ID Name	S_TC_4 Der Test liefert das gelöschte Auto DELETE-deleteById(int Id)
Preconditions	Anfrage: DELETE deleteById(int Id) Antwort: Informationen zum geloschte Fahrzeug • Data: "Auto" ○ int AutoId //Unique {generiert} ○ String Marke ○ String Model ○ int Jahr ○ String Zustand App port number: 9090 Das System ist aktiv
Test Steps	1) Führen Sie das System auf der IDE aus und verwenden Sie den Postmande POST/PUT-Dienst 2) Überprüfen Sie die generierte Antwort in der Postman-Antwort.
Post-Condition	keiner
Test Data	keiner

Expected Result	{
	Marke: "BMW"
	Model: "320"
	Jahr: "2020"
	Zustand: "Verkauft"
	}
	• Fahrzeuginformationen mit Aktualisierungs- oder Erstellungsvorgängen sollten wie oben zurückgegeben werden..

Actual Result	{
	Marke: "BMW"
	Model: "320"
	Jahr: "2020"
	Zustand: "Verkauft"
	}
	• Fahrzeuginformationen gelöscht

Verdict (Pass/Fail)	PASS
Verified UC & Req. IDs	/ZR-6/

Systemtestfälle → System/Komponent-Integrationtest

Integrationstestfälle sind ein wesentlicher Bestandteil des Softwareentwicklungsprozesses, um sicherzustellen, dass verschiedene Komponenten einer Anwendung ordnungsgemäß miteinander interagieren. Diese Testfälle zielen darauf ab, die reibungslose Integration von Modulen, Schnittstellen oder Systemen zu überprüfen und potenzielle Fehler oder Inkompatibilitäten aufzudecken.

Integrationstestfällen werden durch JUnit ausgeführt und für jede Szenario werden die Testen erfolgreich erledigt. Einige davon sind in der nächsten Folienseiten gezeigt.

Systemtestfälle → System/Komponent-Integrationtest

System Test Case ID Name	CL_TC_1 Test liefert gesuchte Autos

Preconditions	• Datenbank Tabelle für Autos
	  ◦ int AutoId = 1
	  ◦ String Marke = "BMW"
	  ◦ String Model = "320"
	  ◦ int Jahr = 2020
	  ◦ String Zustand = "Verkauft"
	• Service call is mocked

Test Steps	1. Sicherstellen, dass das Service mocked ist. 2. JUnit Test Case ruft die Methode getAuto(//gewünschte Attributen) mit AutoId. 3. Antwortdaten werden analysiert und die erwarteten Daten werden aufgerufen.
Post-Condition	keiner
Test Data	getAuto(String BMW, String 320)
Expected Result	Auto.AutoId = 1 Auto.Marke = "BMW" Auto.Model = "320" Auto.Jahr = 2020 Auto.Zustand = "Verkauft"
Actual Result	Auto.AutoId = 1 Auto.Marke = "BMW" Auto.Model = "320" Auto.Jahr = 2020 Auto.Zustand = "Verkauft"
Verdict (Pass/Fail)	PASS
Verified UC & Req. IDs	/ZR-7/ , /BFK-3/

System Test Case ID Name	CL_TC_2 Test löscht Autos mit ihren Ids

Preconditions	• Datenbank Tabelle für Autos
	  ◦ int AutoId = 1
	  ◦ String Marke = "BMW"
	  ◦ String Model = "320"
	  ◦ int Jahr = 2020
	  ◦ String Zustand = "Verkauft"
	• Service call is mocked

Test Steps	1. Sicherstellen, dass das Service mocked ist. 2. JUnit Test Case ruft die Methode löscheAuto(int AutoId) mit der AutoId. 3. Antwortdaten werden analysiert und die erwarteten Daten werden aufgerufen.
Post-Condition	keiner
Test Data	löscheAuto(int AutoId)
Expected Result	Gewünschtes Ergebnis ist, dass das Auto mit der Id = 1 gelöscht ist.
Actual Result	Das Auto mit der Id = 1 ist gelöscht.
Verdict (Pass/Fail)	PASS
Verified UC & Req. IDs	/ZR-6/ , /BFK-3/

Systemtestfälle → System/Komponent-Integrationtest

System Test Case ID Name	CL_TC_3 Test speichert eingegebene Autos POST/PUT save(Auto auto)
Preconditions	• Datenbank Tabelle für Autos ist frei • Service call is mocked
Test Steps	1. Sicherstellen, dass das Service call mocked ist. 2. JUnit Test Case ruft die Methode saveAuto(Auto Auto). 3. Antwortdaten werden analysiert und die erwarteten Daten werden aufgerufen.
Post-Condition	keiner
Test Data	"BMW", "320" , 2020, "Verkauft", mit diesen Attributen wird einen Objekt erstellt.

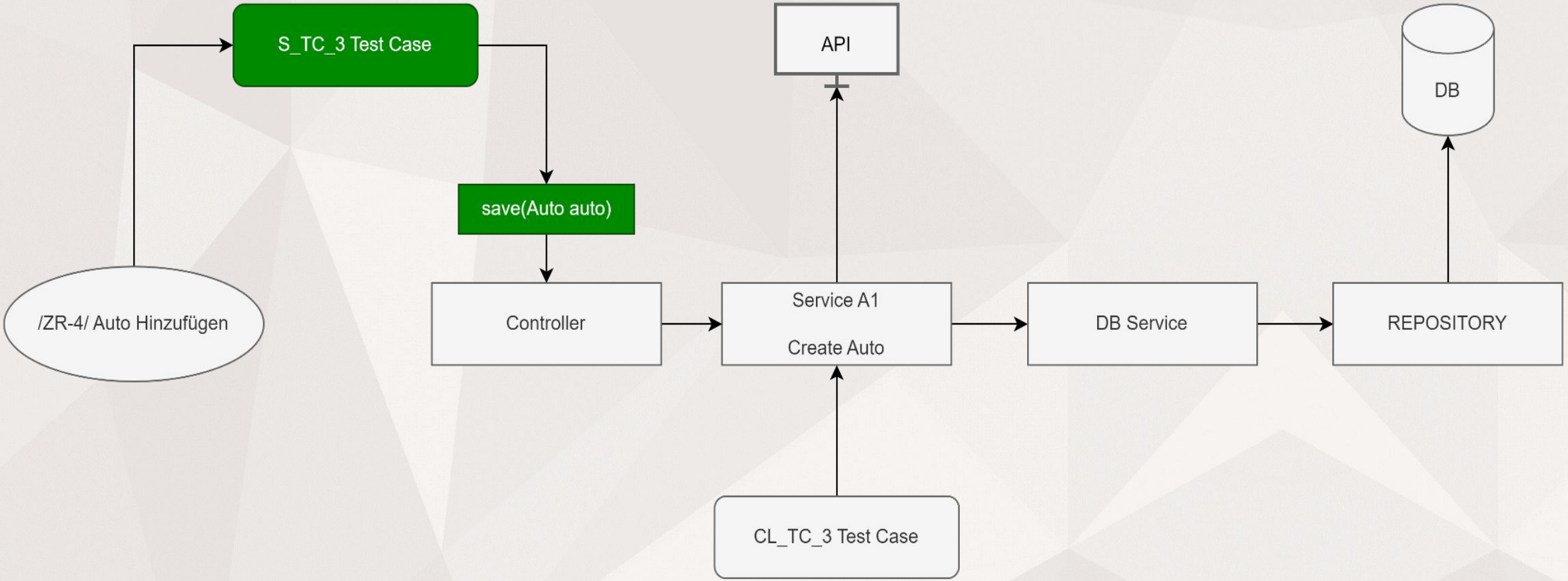
Expected Result	{
	Marke = "BMW"
	Model = "320"
	Jahr = 2020
	Zustand = "Verkauft"
	}

Actual Result	{
	Marke = "BMW"
	Model = "320"
	Jahr = 2020
	Zustand = "Verkauft"
	}

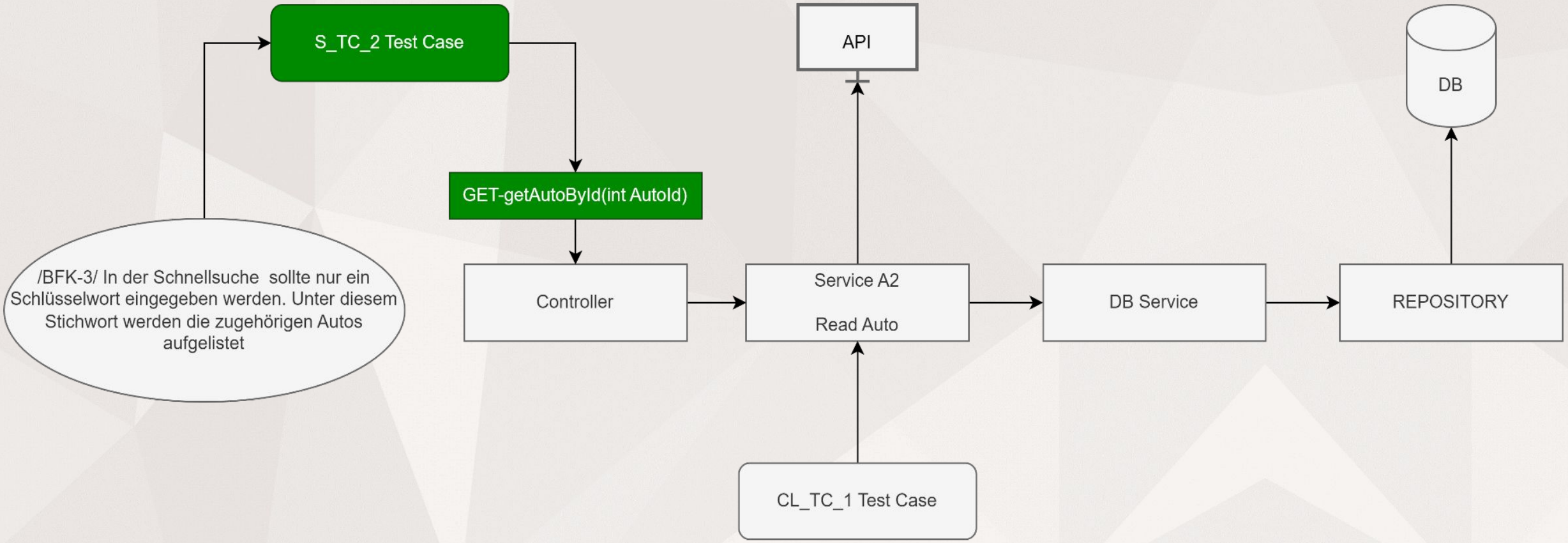
Verdict (Pass/Fail)	PASS
Verified UC & Req. IDs	/ZR-5/ , /ZR-4/ ,

Rückverfolgbarkeit

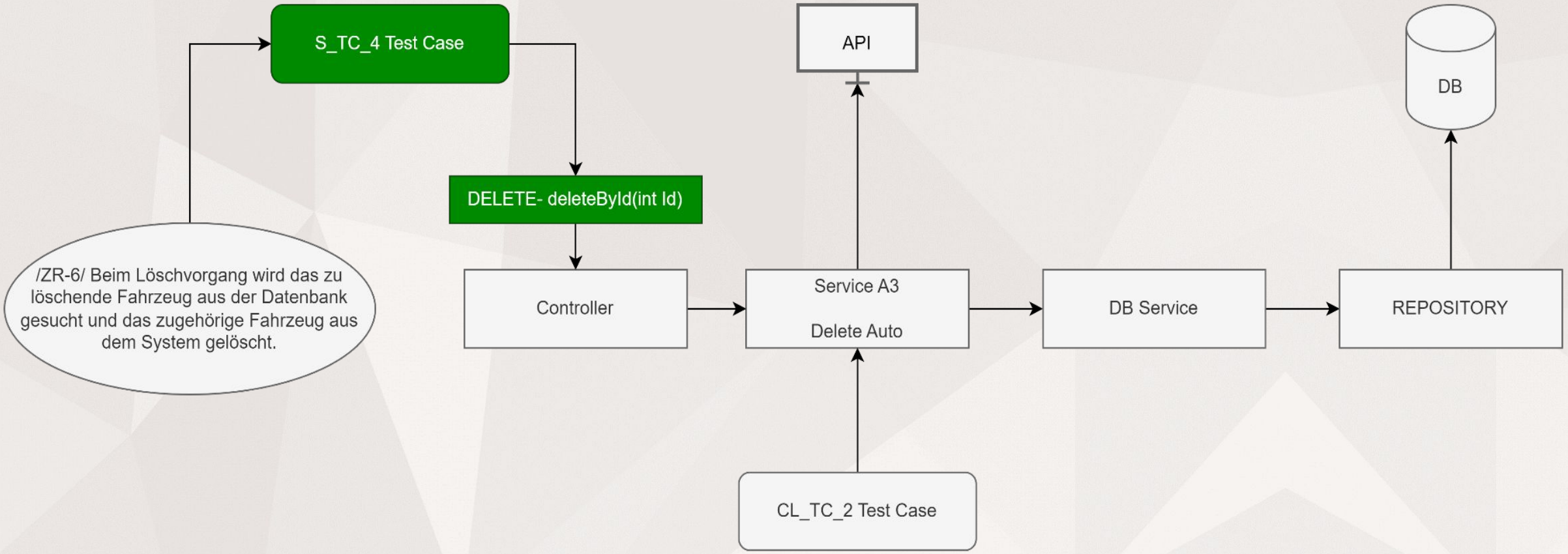
In diesem Kapitel wird die Rückverfolgbarkeit zwischen Anforderungen im Pflichtenheft und Schnittstellen in Architekturdokument zu Systemtestfälle gezeigt.



Rückverfolgbarkeit



Rückverfolgbarkeit



Danke für ihre Aufmerksamkeit