ZÜRCHER LEHRBETRIEBSVERBAND ICT

Praktische Prüfung 295

Backend für Applikationen realisieren

Teilnehmer/in: Bitte vor Prüfungsbeginn ausfüllen.

Vorname	Nachname
Kursnummer	Datum

Kursleiter/in: Bitte nach Korrektur ausfüllen.

Erreichte Punkte	Note
Korrigierende/r	Datum Korrektur

Erstellt am	20.01.2022	Autor	Diego Steiner
Dokumentenstatus	Entwurf	Version	1.1
Verteiler	MP Teilnehmer/in		
Änderungen			

Hinweise

Arbeitsform, Beschreibung	Element a: Schriftliche Prüfung, als Einzelarbeit zu lösen, Element b: Projekt, als Einzelarbeit zu lösen
Zu überprüfende Handlungsziele	HZ1: Richtet die lokale Entwicklungs- und Laufzeitumgebung so ein, dass ein vorgegebenes Projekt entwickelt werden kann. HZ2: Implementiert und dokumentiert mittels vorgegebener Technologie eine effiziente und strukturierte Back-End-Schnittstelle zur Verwaltung (Create, Read, Up-date, Delete) einer existierenden Datenquelle. Nutzt dabei aktuelle Schnittstellen-Standards und hält sich an relevante Vorgaben. HZ3: Überprüft Zwischenergebnisse mit den Anforderungen (funktional, nichtfunktional, Sicherheit) und nimmt laufend Korrekturen vor. HZ4: Hält vorgegebene Coderichtlinien ein und überprüft laufend deren Einhaltung. HZ5: Legt Änderungen und Erweiterungen der Implementierung übersichtlich und zuverlässig in einem Softwareverwaltungssystem ab. HZ6: Implementiert im Back-End einen aktuellen Authentifizierungsmechanismus und schützt mindestens einen Bereich des Back-Ends vor anonymen Zugriffen.
Prüfungsdauer	Element a: 15 Minuten, Element b: Projekt über 1 Tag
Zeitpunkt der Prüfung	Element a: nach dem Vermitteln der geprüften Inhalte (Blöcke 1-4) Element b: ab Tag 5
Punktetotal	Element a: 15, Element b: 90 Bei jeder Aufgabe sind die möglichen Punkte angegeben.
Gewichtung	Element a: 35%, Element b: 65%
Bewertungskriterien	Bewertungsraster mit Punkten und lineare Umrechnung in Noten nach der Formel: ((erreichte Punkte Element a / maximale Punkte Element a * Gewichtung a) + (erreichte Punkte Element b / maximale Punkte Element b * Gewichtung b) *5 / 1 + 1 Der Kompetenznachweis wird auf halbe und ganze Noten gerundet.
Erforderliche Hilfsmittel	Keine
Erlaubte Unterlagen	Es dürfen sowohl persönliche Unterlagen als auch die offiziellen Schulungsunterlagen verwendet werden.
Rohdaten	Element a: keine Element b: OpenAPI Spezifikation
Bemerkungen	Elemente a: Jegliche Kommunikation (Mobiltelefon, Internet, usw.) ist untersagt. Eigene elektronische Hilfsmittel sind erlaubt (Diskette, CD, Memory-Stick, Notebook, Taschenrechner, usw.). Betrugsversuche werden mit der Note 1 gewertet.

Praktische Prüfung

Auftrag

Entwickeln Sie mit den in diesem Modul behandelten Technologien ein funktionstüchtiges Backend, welches die nachfolgenden Anforderungen erfüllt.

Hauptanforderungen

Gewichtung: x3

Das Backend stellt folgende HTTP Endpunkte im JSON-Format bereit:

1. GET /tasks Endpunkt, welcher eine Liste aller Tasks zurück gibt
2. POST /tasks Endpunkt, welcher einen neuen Task erstellt und diesen zurück gibt
3. GET /tasks/{id} Endpunkt, welcher einen einzelnen Task zurück gibt
4. PATCH /tasks/{id} Endpunkt, welcher einen bestehenden Task verändert und diesen zurück gibt
5. DELETE /tasks/{id} Endpunkt, welcher einen bestehenden Task aus der Liste löscht

Anforderungen an die Authentifizierung

Gewichtung: x2

Das Backend stellt zusätzlich folgende HTTP Endpunkte bereit:

6. POST /login Endpunkt, welcher die Credentials entgegennimmt, überprüft und ein Token oder Cookie zurück gibt
7. GET /verify Endpunkt, welcher ein Token oder Cookie auf Gültigkeit überprüft und das Ergebnis zurück gibt
8. DELETE /logout Endpunkt, welcher das mitgegeben Token oder Cookie als ungültig markiert
9. Das mitgegebene Token oder Cookie wird beim Aufruf aller bestehenden Endpunkte überprüft

Zusätzliche Anforderungen

Gewichtung: x1

10. Fehlerhafte Eingaben werden durch das Backend abgefangen (Validation)
11. Fehler und Ausnahmen werden im Code behandelt, abgefangen und geloggt (Errorhandling)
12. Das Projekt wurde sauber aufgesetzt und ist in einer sinnvollen Ordnerstruktur organisiert
13. Der Quellcode wird mit git versioniert
14. Coderichtlinien werden eingehalten und automatisch überprüft
15. Die Funktionsweise der Applikation wird mit Tests geprüft
16. Das Projekt wurde sauber dokumentiert

Bewertung

Hauptanforderungen

Anforderungen	Punkte			
Gewichtung: 3x	0 --	1 -	2 +	3 ++

1. GET /tasks Endpunkt, welcher eine Liste aller Tasks zurück gibt

› Endpunkt antwortet mit dem Statuscode 2XX
› Endpunkt antwortet mit korrektem JSON und passendem Content-Type Header
› Endpunkt antwortet mit den aktuellen Daten

2. POST /tasks Endpunkt, welcher einen neuen Task erstellt und diesen zurück gibt

› Endpunkt antwortet mit dem Statuscode 2XX
› Endpunkt antwortet mit korrektem JSON und passendem Content-Type Header
› Endpunkt aktualisiert die Daten und antwortet mit dem neu erstellen Objekt inkl. ID

3. GET /task/{id} Endpunkt, welcher einen einzelnen Task zurück gibt

› Endpunkt antwortet mit dem Statuscode 2XX oder 4XX, falls es die ID nicht gibt
› Endpunkt antwortet mit korrektem JSON und passendem Content-Type Header
› Endpunkt antwortet mit den aktuellen Daten

4. PUT /task/{id} Endpunkt, welcher den bestehenden Task verändert und diesen zurück gibt

› Endpunkt antwortet mit dem Statuscode 2XX oder 4XX, falls es die ID nicht gibt
› Endpunkt antwortet mit korrektem JSON und passendem Content-Type Header
› Endpunkt aktualisiert die Daten und antwortet mit dem aktualisierten Objekt

5. DELETE /task/{id} Endpunkt, welcher den bestehenden Task löscht

› Endpunkt antwortet mit dem Statuscode 2XX oder 4XX, falls es die ID nicht gibt
› Endpunkt antwortet mit korrektem JSON und passendem Content-Type Header
› Endpunkt aktualisiert die Daten und antwortet mit dem gelöschten Objekt

Anzahl pro Gütestufe

Erreichte Punkte (max. 45 Punkte)

Anforderungen an die Authentifizierung

Beurteilungskriterien	Punkte			
Gewichtung: 2x	0 --	1 -	2 +	3 ++

6. POST /login Endpunkt, welcher die Credentials entgegennimmt, überprüft und ein Token oder Cookie zurück gibt

› Endpunkt antwortet mit dem Statuscode 2XX oder 4XX, falls ungültig
› Endpunkt antwortet mit JSON, setzt einen passenden Header oder übergibt ein Token
› Endpunkt akzeptiert alle Kombinationen mit E-Mail und dem Passwort «m295» als gültige Credentials

7. GET /verify Endpunkt, welcher ein Token oder Cookie auf Gültigkeit überprüft und das Ergebnis zurück gibt

› Endpunkt antwortet mit dem Statuscode 2XX oder 2XX, falls ungültig
› Endpunkt antwortet mit JSON und passendem Content-Type Header
› Endpunkt reflektiert den aktuellen Stand des Token oder Cookies

8. DELETE /logout Endpunkt, welcher das mitgegeben Token oder Cookie als ungültig markiert

› Endpunkt antwortet mit dem Statuscode 2XX
› Endpunkt antwortet ohne Content
› Endpunkt markiert das Token oder Cookie als ungültig

9. Das mitgegebene Token oder Cookie wird beim Aufruf aller bestehenden Endpunkte überprüft

› Bisherige Endpunkte antworten mit dem Statuscode 4XX falls nicht authentifiziert
› Bisherige Endpunkte reflektieren den aktuellen Stand des Token oder Cookies
› Wird ein Statuscode 403 zurück gegeben, wird eine entsprechende Meldung als JSON mitgegeben

Anzahl pro Gütestufe

Erreichte Punkte (max. 24 Punkte)

Zusätzliche Anforderungen

Beurteilungskriterien	Punkte			
Gewichtung: 1x	0 --	1 -	2 +	3 ++

10. Fehlerhafte Eingaben werden durch das Backend abgefangen (Validation)

› Anfragen an nicht existente Endpunkte werden mit dem HTTP Statuscode 404 beantwortet
› Tasks mit leerem Titel werden von keinem Endpunkt akzeptiert und mit dem HTTP Statuscode 4XX beantwortet
› Zusätzlich wird eine Fehlermeldung als JSON mitgegeben

11. Fehler und Ausnahmen werden im Code behandelt, abgefangen und geloggt (Errorhandling)

› Fehler in einem Endpunkt werden abgefangen und führen nicht zu einem Absturz, sondern werden mit dem HTTP Statuscode 5XX beantwortet.
› Falsches Aufrufen von Endpunkten führt zu keinen unbehandelten Fehlern
› Alle Anfragen werden mit sinnvollem Detailgrad geloggt.

12. Das Projekt ist sauber aufgesetzt und ist in einer sinnvollen Ordnerstruktur organisiert.

› Die Ordnerstruktur ist zweckmässig unterteilt mit separaten Ordnern für Code, Konfiguration und Dokumentation
› Die Dateien sind entsprechend benannt und eingeordnet
› Der Code ist sauber aufgeteilt, z.B. mit Controllern

13. Der Quellcode des Projektes wird mit git versioniert

› Git ist im Quellcodeordner korrekt aufgesetzt
› Pro Halbtag des Kurses gibt es mindestens einen Commit
› Commit-Messages sind aussagekräftig und sinnvoll

14. Coderichtlinien werden eingehalten und automatisch überprüft

› Der Code genügt den von der Kursleitung festgelegten Kriterien für sauberen Code (2 Punkte)
› Es wurde geeignetes Werkzeug zur automatisierten Überprüfung von Coderichtlinien eingerichtet

15. Die Funktionsweise der Applikation wird mit Tests überprüft

› Testfälle für die Hauptanforderungen wurden definiert (automatisiert und/oder manuell)
› Testfälle für die Anforderungen an die Authentifizierung wurden definiert
› Die Testfälle laufen fehlerfrei durch oder wurden sauber protokolliert

16. Das Projekt wurde sauber dokumentiert

› Die Applikation verfügt über eine README-Datei mit Namen des Projektes und Autor
› Die README-Datei beinhaltet die Dokumentation der Entwicklungsumgebung (Setup, Runtime, …)
› Die Dokumentation der Endpunkte wurde im Projekt mitgeliefert.

Anzahl pro Gütestufe

Erreichte Punkte (max. 21 Punkte)