

# 324-09A - Systemtests und Image Scan

## Übersicht der implementierten Systemtests

Wir haben uns für die erste Option entschieden (Erweitere die Tests mit Systemtests). Das Todo-List Projekt wurde um umfassende Systemtests erweitert, die sowohl Frontend als auch Backend abdecken. Alle 8 geforderten Testfälle wurden implementiert und durch zusätzliche User-Story-Tests ergänzt.

### Vollständige Abdeckung der geforderten Testfälle

| Nr. | Testfall                                   | Implementierung          |
|-----|--|--------------------------|
| 1   | Neues Element zur Aufgabenliste hinzufügen | Frontend Systemtest      |
| 2   | Korrekte Anzahl von Elementen anzeigen     | Frontend Systemtest      |
| 3   | "Erledigt"-Button funktioniert             | Frontend Systemtest      |
| 4   | Element aus Liste entfernen                | Frontend Systemtest      |
| 5   | Fehlermeldung bei leerem Element           | Frontend Systemtest      |
| 6   | Fehlermeldung bei Lade-Fehler              | Frontend Systemtest      |
| 7   | Aufgabenliste korrekt laden                | Backend Integration Test |
| 8   | Doppelte Einträge ablehnen                 | Backend Integration Test |

### Zusätzliche User Story Tests

- **Prioritäten-System:** Farbliche Darstellung (hoch = rot, niedrig = grün)
- **Filter-Funktionalität:** Alle/Offen/Erledigt Filter
- **Status-Management:** Checkbox für completed/uncompleted
- **API-Versionierung:** Tests für v1/v2 Endpoints

## Frontend Systemtests

**Datei:** [https://gitlab.com/me7554020/m324\\_projekt\\_todolist-gemeinsam2/-/blob/main/frontend/src/App.system.test.js?ref\\_type=heads](https://gitlab.com/me7554020/m324_projekt_todolist-gemeinsam2/-/blob/main/frontend/src/App.system.test.js?ref_type=heads)

**Framework:** React Testing Library + Jest

**Ergebnis:**  7/7 Tests grün

Die Frontend-Tests verwenden Mock-Strategien für Backend-Calls und testen realistische Benutzerinteraktionen.

## Implementierte Tests

### Test 1: Neues Element hinzufügen

```
test('sollte neues Element zur Aufgabenliste hinzufügen', async () => {  
  // Testet: Eingabe → Absenden-Button → API-Call  
  // Verifiziert: Korrekte JSON-Struktur an Backend  
});
```

**Zweck:** Grundfunktion der Todo-App – neue Aufgaben erstellen

### Test 2: Korrekte Anzahl anzeigen

```
test('sollte korrekte Anzahl von Elementen anzeigen', async () => {  
  // Testet: Task-Counter (Gesamt/Erledigt/Offen)  
  // Verifiziert: Mathematische Korrektheit der Anzeige  
});
```

**Zweck:** UI-Feedback für Benutzer über Aufgabenstatus

### Test 3: Status umschalten (Erledigt-Button)

```
test('sollte Status mit Checkbox umschalten können', async () => {  
  // Testet: Checkbox-Click → API-Call → UI-Update  
  // Verifiziert: toggle-status Endpoint wird korrekt aufgerufen  
});
```

**Zweck:** Kernfunktion für Aufgaben-Management

### Test 4: Element entfernen

```
test('sollte Element aus Liste entfernen', async () => {  
  // Testet: Löschen-Button → Delete-API → UI-Update  
  // Verifiziert: Korrekte Entfernung aus Interface  
});
```

**Zweck:** CRUD-Operation "Delete" im Frontend

### Test 5-6: Filter und Prioritäten

- **Filter-Test:** Verifiziert Alle/Offen/Erledigt Buttons
- **Prioritäten-Test:** Prüft farbliche Darstellung je nach Priorität

**Zweck:** User-Story Funktionalitäten für bessere UX

# Backend Integration Tests

**Datei:** [https://gitlab.com/me7554020/m324\\_projekt\\_todolist-gemeinsam2/-/blob/main/backend/src/test/java/com/example/demo/TodoIntegrationTest.java?ref\\_type=heads](https://gitlab.com/me7554020/m324_projekt_todolist-gemeinsam2/-/blob/main/backend/src/test/java/com/example/demo/TodoIntegrationTest.java?ref_type=heads)

**Framework:** Spring Boot Test + TestRestTemplate

**Ergebnis:**  5/5 Tests grün

Die Backend-Tests verwenden eine H2-In-Memory-Datenbank und testen echte HTTP-Calls.

## Implementierte Tests

### Test 1: Task erstellen und abrufen (Full-Stack)

```
@Test
public void sollteTaskErstellenUndAbrufen() throws Exception {
    // POST /tasks → GET / → Datenbank-Verifizierung
    // Testet: Kompletter CRUD-Zyklus mit Persistierung
}
```

### Test 2: Task löschen

```
@Test
public void sollteTaskLoeschen() throws Exception {
    // Create → DELETE → Verifizierung der Entfernung
    // Testet: Datenbankänderungen und API-Response
}
```

### Test 3: Task Status umschalten

```
@Test
public void sollteTaskStatusUmschalten() throws Exception {
    // toggle-status Endpoint → Boolean-Umschaltung in DB
    // Testet: User Story Status-Management
}
```

### Test 4: API Versionierung

```
@Test
public void sollteAPIVersionenTesten() throws Exception {
    // /api/v1/ und /api/v2/ Endpoints
    // Testet: Backward-Compatibility und neue Features
}
```

### Test 5: Doppelte Tasks ablehnen

```
@Test
public void sollteDoppelteTasksIgnorieren() throws Exception {
    // Duplikatsschutz der Geschäftslogik
    // Testet: Datenintegrität und Validierung
}
```

### 3.2 Test-Konfiguration

**Datei:** [https://gitlab.com/me7554020/m324\\_projekt\\_todolist-gemeinsam2/-/blob/main/backend/src/test/resources/application-test.properties?ref\\_type=heads](https://gitlab.com/me7554020/m324_projekt_todolist-gemeinsam2/-/blob/main/backend/src/test/resources/application-test.properties?ref_type=heads)

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.jpa.hibernate.ddl-auto=create-drop
logging.level.org.springframework.web=DEBUG
```

**Zweck:** Isolierte Testumgebung ohne Seiteneffekte

## CI/CD Integration

### GitLab Pipeline Status

me / M324\_PROJEKT\_TODOLIST-Gemeinsam2 / Pipelines / #1846222433

#### Systemtests hinzugefügt: Frontend E2E Tests und Backend Integration Tests Delete

✓ Passed Kristian Lubina created pipeline for commit ee7e729d 1 hour ago, finished 57 minutes ago

For **main**

latest branch 8 jobs 10.25 5 minutes 15 seconds, queued for 1 seconds

Pipeline Jobs 8 Tests 5

| build             | test             | docker-build            | docker-push            |
|-------------------|------------------|-------------------------|------------------------|
| ✓ build:java-api  | ✓ test:java-api  | ✓ docker:build-backend  | ✓ docker:push-backend  |
| ✓ build:react-app | ✓ test:react-app | ✓ docker:build-frontend | ✓ docker:push-frontend |

## Lokale Testausführung

### Frontend Tests

| File        | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s |
|-------------|---------|----------|---------|---------|-------------------|
| All files   | 68.68   | 56.45    | 63.15   | 68.42   |                   |
| App.js      | 77.27   | 60.34    | 66.66   | 77.38   | ...14,290-345,373 |
| index.js    | 0       | 100      | 100     | 0       | 7-17              |
| ...itals.js | 0       | 0        | 0       | 0       | 1-8               |

Test Suites: 2 passed, 2 total  
Tests: 7 passed, 7 total  
Snapshots: 0 total  
Time: 5.136 s, estimated 7 s  
Ran all test suites.

kris@kris-VirtualBox: ~/M324\_PROJEKT\_TODOLIST-Gemeinsam2/frontend\$

## Backend Tests

```
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.0
26 s -- in com.example.demo.TODOIntegrationTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO]
[INFO] Total time: 29.598 s
[INFO] Finished at: 2025-05-31T11:37:38+02:00
[INFO] -----
-----
```

## Technische Details

### Frontend Testing Stack

- **React Testing Library:** Realistische DOM-Interaktionen
- **Jest Mocks:** API-Call Simulation ohne Backend-Abhängigkeit
- **waitFor:** Asynchrone UI-Updates abwarten

### Backend Testing Stack

- **Spring Boot Test:** Vollständiger Application Context
- **TestRestTemplate:** Echte HTTP-Calls für Integration Testing
- **H2 Database:** In-Memory DB für schnelle, isolierte Tests

## Qualitätssicherung

### Testqualität:

- **Isolation:** Jeder Test läuft unabhängig
- **Determinismus:** Reproduzierbare Ergebnisse
- **Performance:** Schnelle Ausführung (<35s total)
- **Wartbarkeit:** Klare, verständliche Testnamen

### Mock-Strategien:

```
// Frontend: API-Calls mocken
fetch.mockResolvedValueOnce({
  ok: true,
  json: async () => mockData
});
```

```
// Backend: Echte Integration Tests
@SpringBootTest(webEnvironment = RANDOM_PORT)
// Vollständiger Application Context + HTTP-Calls
```

## Fazit und Bewertung

### Zielerreichung

#### **Vollständige Erfüllung der Aufgabenstellung:**

- Alle 8 geforderten Testfälle implementiert
- Zusätzliche User-Story-Tests für vollständige Abdeckung
- Frontend und Backend Integration getestet
- CI/CD Pipeline erfolgreich integriert

### Technischer Mehrwert

Die Implementierung geht über die Mindestanforderungen hinaus:

- **Umfassende Testabdeckung:** Frontend + Backend + Integration
- **Moderne Testpraktiken:** React Testing Library + Spring Boot Test
- **CI/CD Integration:** Automatisierte Qualitätssicherung
- **Wartbare Architektur:** Erweiterbar für zukünftige Entwicklungen