# M294 PROJEKTDOKUMENTATION - SLOT MACHINE CASINO

**Modul:** M294 - Frontend für Applikationen realisieren  
**Student:** Dusan Stevic  
**Datum:** 18. Juli 2025  
**GitHub Repository:** https://github.com/tu-shang/M294-Slot-Machine-Casino

## INHALTSVERZEICHNIS

1. Projektidee
2. Anforderungskatalog
3. Klassendiagramm
4. Storyboard
5. Screen-Mockups
6. REST-Schnittstellen
7. Testplan
8. Test-Durchführung
9. Unit-Test Durchführung
10. Installationsanleitung
11. Hilfestellungen

## 1. PROJEKTIDEE

### Elevator Pitch

Das **Slot Machine Casino** ist eine vollständige Casino-Simulation mit Spielerverwaltung und realistischer Slot Machine. Spieler können sich registrieren, Coins verwalten und an einer 3-Reel Slot Machine mit verschiedenen Gewinnkombinationen spielen.

Die Anwendung kombiniert ein Spring Boot REST-API Backend mit einem modernen React Frontend und bietet eine authentische Casino-Erfahrung mit Animationen, Gewinnlogik und persistenter Datenhaltung.

### Kernfunktionen

* Player Management (CRUD Operations)
* Slot Machine mit realistischen Gewinnchancen
* Coin-System mit Aufladung (max. 1000 Coins)
* Casino-Design mit professionellen Animationen
* MySQL Datenpersistierung
* Responsive Design für verschiedene Bildschirmgrößen

### Technologie-Stack

* **Backend:** Spring Boot 3.5.3, Java 17, MySQL
* **Frontend:** React 19.1.0, Vite, CSS3 Animationen
* **Testing:** Vitest, React Testing Library
* **Deployment:** GitHub Repository, start-all.cmd

## 2. ANFORDERUNGSKATALOG

### User Story 1: Player Registration

**Als** neuer Benutzer  
**möchte ich** mich als Spieler registrieren können  
**damit ich** am Casino teilnehmen kann

**Akzeptanzkriterien:** Spielername muss eingegeben werden

* Start-Coins können festgelegt werden (Standard: 1000, max. 1000)
* Spieler wird in MySQL Datenbank gespeichert
* Validation: Name darf nicht leer sein
* Validation: Coins müssen zwischen 0-1000 liegen

### User Story 2: Slot Machine Spielen

**Als** registrierter Spieler  
**möchte ich** an der Slot Machine spielen können  
**damit ich** Coins gewinnen kann

**Akzeptanzkriterien:** 10 Coins Einsatz pro Spiel

* 3 zufällige Symbole werden server-seitig generiert
* Gewinn nur bei 3 gleichen Symbolen
* Coins werden automatisch aktualisiert
* Realistische Animation während des Spiels
* Verschiedene Gewinnwerte je Symbol

### User Story 3: Coin Management

**Als** Spieler  
**möchte ich** meine Coins verwalten können  
**damit ich** weiterspielen kann

**Akzeptanzkriterien:** Aktuelle Coins werden in Echtzeit angezeigt

* Coins können auf 1000 aufgeladen werden
* Spiel nur bei genügend Coins (≥10) möglich
* Coin-Änderungen werden persistent gespeichert
* Maximale Coins begrenzt auf 1000

### User Story 4: Player Verwaltung

**Als** Spieler  
**möchte ich** meinen Account verwalten können  
**damit ich** die Kontrolle über meine Daten habe

**Akzeptanzkriterien:** Player können gelöscht werden

* Bestätigungsdialog vor Löschung
* Player-Wechsel jederzeit möglich
* Übersichtliche Player-Auswahl mit Coin-Anzeige

### User Story 5: Benutzerfreundlichkeit

**Als** Benutzer  
**möchte ich** eine intuitive Casino-Erfahrung  
**damit ich** das Spiel sofort verstehe

**Akzeptanzkriterien:** Animationen für bessere User Experience

* Klare Gewinn/Verlust Anzeigen
* Responsive Design für Mobile und Desktop
* Casino-typisches Design mit Gold-Akzenten
* Einfache Navigation zwischen Screens

## Ein Bild, das Text, Screenshot, Diagramm enthält. KI-generierte Inhalte können fehlerhaft sein.3. KLASSENDIAGRAMM

### Beziehungen:

* **PlayerController** verwaltet Player-Entitäten über PlayerRepository
* **SlotController** nutzt Player-Daten für Spiellogik über PlayerRepository
* **GlobalExceptionHandler** fängt Exceptions ab und wandelt sie in Benutzerfreundliche responses um.
* **PlayerRepository** ist die Datenzugriffsschicht (JPA Interface)
* **Player** ist die zentrale Entity mit JPA Annotationen
* **Corsconfig** legt fest, welche Domains, HTTP-Methoden und Header beim Cross-Origin-Zugriff auf eine Webressource erlaubt sind.

## 4. STORYBOARD

### Hauptablauf der Anwendung

Ein Bild, das Screenshot, Diagramm, Design, Kunst enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Use Case: Slot Machine Spielen

1. **Player auswählen** → Aus Player-Liste wählen
2. **“Pull Lever” klicken** → Hebel-Animation startet
3. **Reels drehen** → 3 Sekunden realistische Animation
4. **Backend-Call** → Server generiert zufällige Symbole
5. **Ergebnis anzeigen** → Gewinn/Verlust Message mit Coin-Update

## 5. SCREEN-MOCKUPS

### Screen 1: Player Selection

╔══════════════════════════════════════════════════════════╗  
║ 🎰 CASINO ROYALE ║  
║ Wähle deinen Spieler oder ║  
║ erstelle einen neuen ║  
╠══════════════════════════════════════════════════════════╣  
║ 👤 Neuen Spieler erstellen ║  
║ ┌─────────────────────────────┐ ┌─────────┐ ║  
║ │ Spielername eingeben... │ │ 1000 │ ║  
║ └─────────────────────────────┘ └─────────┘ ║  
║ [✨ Spieler erstellen] ║  
║ ║  
║ 🎮 Bestehenden Spieler auswählen ║  
║ ┌────────────────────────────────────────────────────┐ ║  
║ │ Max 500 💰 Coins ▶️ │ ║  
║ ├────────────────────────────────────────────────────┤ ║  
║ │ Anna 750 💰 Coins ▶️ │ ║  
║ └────────────────────────────────────────────────────┘ ║  
╚══════════════════════════════════════════════════════════╝

### Screen 2: Slot Machine

╔══════════════════════════════════════════════════════════╗  
║ 🎮 Max 💰 500 Coins ║  
║ [← Spieler wechseln] ║  
╠══════════════════════════════════════════════════════════╣  
║ 🎰 SLOT MACHINE ║  
║ ║  
║ ╔═══╗ ╔═══╗ ╔═══╗ ║  
║ ║ 🍒 ║ ║ 🍋 ║ ║ 💎 ║ ║  
║ ╚═══╝ ╚═══╝ ╚═══╝ ║  
║ ║  
║ [🕹️ PULL (10 Coins)] ║  
║ ║  
║ ❌ Kein Gewinn - Versuch es nochmal! ║  
║ 🔴 ║  
║ │ ║  
║ │ ║  
║ [💳 Coins aufladen (1000)] [🗑️ Spieler löschen] ║  
╚══════════════════════════════════════════════════════════╝

## 6. REST-SCHNITTSTELLEN

### Player Management Endpoints

#### GET /player

**Beschreibung:** Alle Spieler abrufen  
**URL:** http://localhost:8080/player  
**Method:** GET  
**Response:**

[  
 {  
 "id": 1,  
 "name": "Max",  
 "coins": 500  
 },  
 {  
 "id": 2,  
 "name": "Anna",  
 "coins": 750  
 }  
]

#### POST /player

**Beschreibung:** Neuen Spieler erstellen  
**URL:** http://localhost:8080/player  
**Method:** POST  
**Request Headers:** Content-Type: application/json  
**Request Body:**

{  
 "name": "Max",  
 "coins": 1000  
}

**Response:** Created Player Object mit generierter ID **Validation:** - Name nicht leer - Coins zwischen 0-1000

#### PUT /player/recharge/{id}

**Beschreibung:** Spieler-Coins auf 1000 aufladen  
**URL:** http://localhost:8080/player/recharge/1  
**Method:** PUT  
**Response:** Updated Player Object **Fehlerbehandlung:** RuntimeException wenn Player nicht gefunden

#### DELETE /player/{id}

**Beschreibung:** Spieler löschen  
**URL:** http://localhost:8080/player/1  
**Method:** DELETE  
**Response:** 200 OK

### Slot Machine Endpoints

#### POST /slot/pull/{playerId}

**Beschreibung:** Slot Machine ziehen  
**URL:** http://localhost:8080/slot/pull/1  
**Method:** POST  
**Response bei Gewinn:**

{  
 "slots": ["🍒", "🍒", "🍒"],  
 "win": true,  
 "winAmount": 100,  
 "coins": 590  
}

**Response bei Verlust:**

{  
 "slots": ["🍒", "🍋", "💎"],  
 "win": false,  
 "winAmount": 0,  
 "coins": 490  
}

**Error Response:**

{  
 "error": "Not enough coins"  
}

### Gewinnkombinationen

* 💎 Diamant: 500 Coins (50x Einsatz)
* ⭐ Stern: 200 Coins (20x Einsatz)
* 🔔 Glocke: 150 Coins (15x Einsatz)
* 🍒 Kirsche: 100 Coins (10x Einsatz)
* 🍋 Zitrone: 80 Coins (8x Einsatz)
* 🍉 Wassermelone: 60 Coins (6x Einsatz)

## 7. TESTPLAN

### Testfall 1: Player erstellen

**Vorbedingung:** Anwendung läuft, MySQL verfügbar  
**Schritte:** 1. Namen “TestUser” eingeben 2. Coins “500” eingeben  
3. “Erstellen” klicken **Erwartetes Ergebnis:** Player erscheint in Player-Liste, DB-Eintrag erstellt

### Testfall 2: Player erstellen - Validation

**Vorbedingung:** Anwendung läuft  
**Schritte:** 1. Namen leer lassen 2. “Erstellen” klicken **Erwartetes Ergebnis:** Validation verhindert Erstellung, Error-Message

### Testfall 3: Slot Machine - Erfolgreich

**Vorbedingung:** Player mit ≥10 Coins ausgewählt  
**Schritte:** 1. “Pull Lever” klicken 2. Animation abwarten (3 Sekunden) **Erwartetes Ergebnis:** 10 Coins abgezogen, Ergebnis angezeigt, DB aktualisiert

### Testfall 4: Slot Machine - Nicht genügend Coins

**Vorbedingung:** Player mit <10 Coins ausgewählt  
**Schritte:** 1. Slot Machine Button prüfen **Erwartetes Ergebnis:** Button ist disabled, kein Pull möglich

### Testfall 5: Coins aufladen

**Vorbedingung:** Player ausgewählt  
**Schritte:** 1. “Aufladen” klicken **Erwartetes Ergebnis:** Coins auf 1000 gesetzt, UI aktualisiert

### Testfall 6: Player löschen

**Vorbedingung:** Player ausgewählt  
**Schritte:** 1. “Löschen” klicken 2. Bestätigung “Ja” **Erwartetes Ergebnis:** Player aus DB entfernt, zur Auswahl zurück

### Testfall 7: Coins-Limit Validation

**Vorbedingung:** Player-Erstellung Form  
**Schritte:** 1. Coins “1500” eingeben 2. “Erstellen” klicken **Erwartetes Ergebnis:** Validation verhindert Erstellung, Alert “Maximale Anzahl Coins ist 1000!”

## 8. TEST-DURCHFÜHRUNG

### Manuelle Tests

#### Testfall 1: Player erstellen ✅

**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Bemerkung:** Player wird korrekt erstellt und in Player-Liste angezeigt. MySQL Persistierung funktioniert.

#### Testfall 2: Player erstellen - Validation ✅

**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Bemerkung:** Required-Validation funktioniert, leere Namen werden verhindert.

#### Testfall 3: Slot Machine - Erfolgreich ✅

**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Bemerkung:** Animation läuft flüssig, Coins werden korrekt abgezogen, Backend-Integration funktioniert.

#### Testfall 4: Slot Machine - Nicht genügend Coins ✅

**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Bemerkung:** Button wird automatisch disabled bei <10 Coins.

#### Testfall 5: Coins aufladen ✅

**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Bemerkung:** 1000 Coins werden korrekt hinzugefügt, UI Update funktioniert.

#### Testfall 6: Player löschen ✅

**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Bemerkung:** Player wird aus DB entfernt, UI kehrt zur Auswahl zurück.

#### Testfall 7: Coins-Limit Validation ✅

**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Bemerkung:** Frontend und Backend validieren max. 1000 Coins korrekt.

## 9. UNIT-TEST DURCHFÜHRUNG

#### Unit-Test 1: App Rendering ✅

**Test:** App renders without crashing  
**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Code:** Testet ob die App ohne Fehler lädt und “CASINO ROYALE” anzeigt **Bemerkung:** Grundlegendes Rendering funktioniert fehlerfrei

#### Unit-Test 2: Form Validation ✅

**Test:** Player creation form validation works  
**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Code:** Testet HTML5 Validation (min=“0”, max=“1000”) **Bemerkung:** Eingabefelder haben korrekte Validation-Attribute

#### Unit-Test 3: Player Creation ✅

**Test:** Player can be created with valid data  
**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Code:** Mock-API Test für Player-Erstellung **Bemerkung:** API-Call wird mit korrekten Daten ausgeführt

#### Unit-Test 4: Player Selection ✅

**Test:** Player selection shows game screen  
**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Code:** Testet Navigation zur Slot Machine **Bemerkung:** Screen-Wechsel funktioniert korrekt

#### Unit-Test 5: Slot Button Disabled ✅

**Test:** Slot machine button disabled when insufficient coins  
**Durchgeführt am:** 17.07.2025  
**Ergebnis:** BESTANDEN  
**Code:** Testet disabled-State bei <10 Coins **Bemerkung:** Button-Logic funktioniert wie erwartet

### Test-Execution

**Test-Kommando:** npm test  
**Test-Framework:** Vitest mit React Testing Library  
**Coverage:** 5/5 Tests erfolgreich (100%)  
**Execution Time:** ~1.2 Sekunden

## 10. INSTALLATIONSANLEITUNG

### GitHub Repository

**GitHub URL:** https://github.com/tu-shang/M294-Slot-Machine-Casino  
**Backup:** Vollständiges Projekt ist in GitHub versioniert und gesichert

### Systemvoraussetzungen

* **Java:** Version 17 oder höher
* **Node.js:** Version 18 oder höher
* **MySQL Server:** läuft auf localhost:3306
* **Maven:** für Backend-Build
* **Git:** für Repository-Klonen

### Installation Schritt-für-Schritt

#### 1. Repository klonen

git clone https://github.com/tu-shang/M294-Slot-Machine-Casino.git  
cd M294-Slot-Machine-Casino

#### 2. MySQL Datenbank erstellen

CREATE DATABASE slotmachine;  
CREATE USER 'slotuser'@'localhost' IDENTIFIED BY 'slotpass';  
GRANT ALL PRIVILEGES ON slotmachine.\* TO 'slotuser'@'localhost';  
FLUSH PRIVILEGES;

#### 3. Backend-Konfiguration prüfen

**Datei:** backend/src/main/resources/application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/slotmachine  
spring.datasource.username=slotuser  
spring.datasource.password=slotpass  
spring.jpa.hibernate.ddl-auto=create-drop

**Bei Bedarf MySQL-Credentials anpassen**

#### 4. Automatischer Start (Empfohlen)

**Windows:**

start-all.cmd

Startet automatisch: - MySQL Workbench  
- Backend (Spring Boot) - Frontend (Vite Dev Server) - Browser mit <http://localhost:5173>

#### 5. Manueller Start

**Backend:**

cd backend  
mvnw spring-boot:run

**Frontend:**

cd frontend/frontend  
npm install  
npm run dev

### URLs nach erfolgreichem Start

* **Frontend:** http://localhost:5173
* **Backend API:** http://localhost:8080
* **API Test:** http://localhost:8080/player
* **Health Check:** http://localhost:8080/actuator/health

### Troubleshooting

* **MySQL Connection Error:** Credentials in application.properties prüfen
* **Port 8080 belegt:** Backend Port in application.properties ändern
* **Port 5173 belegt:** Frontend Port in vite.config.js ändern
* **Node Modules fehlen:** npm install in frontend/frontend/ ausführen

## 11. HILFESTELLUNGEN

### Verwendete Ressourcen

#### Offizielle Dokumentationen

* **Spring Boot Documentation:** https://spring.io/projects/spring-boot
  + REST Controller Setup
  + JPA Entity Relationships
  + MySQL Integration
* **React Documentation:** https://react.dev/
  + Hooks (useState, useEffect)
  + Component Architecture
  + Event Handling
* **Vite Documentation:** https://vitejs.dev/
  + Build Configuration
  + Development Server Setup

#### Web-Ressourcen

* **MDN Web Docs:** CSS Animations und Flexbox Layout
* **Stack Overflow:** Spezifische Implementierungsfragen
  + React Animation Timing
  + MySQL Connection Troubleshooting
  + Git Repository Setup
* **W3Schools:** HTML5 Form Validation

#### Tools und Software

* **GitHub Copilot:** Code-Completion und Debugging-Hilfe
* **MySQL Workbench:** Datenbank-Management
* **Postman:** REST-API Testing während Entwicklung
* **VS Code Extensions:**
  + Spring Boot Tools
  + React Snippets
  + GitLens
  + Github Copilot

### Unterstützung von Mitstudierenden

#### Peer Code Review

* **Luca Stalder:** Tipps
* **Carmine Vaccaro:** Feedback zum CSS Casino-Design

## FAZIT

Das Slot Machine Casino Projekt demonstriert eine vollständige Full-Stack Webanwendung mit modernen Technologien. Die Implementierung erfüllt alle Anforderungen des Moduls M294:

### Erfüllte Anforderungen

**Frontend Development:** React mit professionellen Komponenten  
**REST-API Integration:** Vollständige CRUD-Operationen  
**Datenvalidierung:** Client- und serverseitig  
**Responsive Design:** Mobile-friendly Casino-Interface  
**Automatisierte Tests:** 5 Unit-Tests mit 100% Success Rate  
**Dokumentation:** Vollständige technische und funktionale Dokumentation  
**Version Control:** GitHub Repository mit professioneller Historie

### Technische Highlights

* **Real-time Animation:** Flüssige Slot Machine Reels
* **State Management:** Konsistenter Application State
* **Error Handling:** Graceful Fehlerbehandlung
* **Security:** Input Validation und SQL Injection Prevention
* **Performance:** Optimierte Bundle Size und Loading Times

### Lessons Learned

* **Component Architecture:** Modulare React-Struktur vereinfacht Wartung
* **API Design:** RESTful Endpoints verbessern Frontend-Integration
* **Testing Strategy:** Unit-Tests erhöhen Code-Qualität signifikant
* **Documentation:** Umfassende Docs beschleunigen Onboarding

**Projektabschluss:** 18. Juli 2025  
**GitHub Repository:** https://github.com/tu-shang/M294-Slot-Machine-Casino  
**Gesamtaufwand:** ~40 Stunden