Internetauftritt Studentenkasino

Projektdokumentation

Studienjahrgang: 2024
Kurs: A

Fakultät: Wirtschaft
Studiengang: Wirtschaftinformatik
Modul: Fallstudien / Herr Schulik
Semester: Sommersemester 2025

Projektteam:

Corvin Annen – Projektleitung
Elijah Mossmann – Backend
Timo Manz – Frontend
Michael Zerner – Hosting & Deployment

DHBW Villingen-Schwenningen
Abgabedatum: 27.11.2025

DHBW Duale Hochschule Baden-Württemberg Villingen-Schwenningen

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
DMZ	Demilitarisierte Zone
DNS	Domain Name System
ERM	Entity-Relationship-Modell
ORM	Object-Relational Mapping
PaaS	Platform as a Service
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
JSON	JavaScript Object Notation
REST	Representational State Transfer
SOP	Same-Origin Policy
SQLite	Structured Query Language lite
SSL	Secure Sockets Layer
UI	User Interface
URL	Uniform Resource Locator
VPS	Virtual Private Server

Abbildungsverzeichnis

1 Einleitung

1.1 Problemstellung

Die vorliegende Arbeit behandelt die Konzeption und Umsetzung einer Webanwendung, die sowohl eine gute Performence wie auch eine komfortable skalierbarkeit enthalten soll. Moderne Webprojekte stehen vor der Herausforderung, gleichzeitig nutzerfreundlich, zuverlässig und bei wachsender Nutzung skalierbar zu bleiben. Besonders bei Anwendungen, die spielerische Elemente oder Wettbewerbsmechanismen enthalten, steigt die technische Komplexität deutlich, da Echtzeitinteraktionen und schnelle Datenverarbeitung erforderlich sind. Ein weiteres Problem ist die sichere Verwaltung von Benutzerdaten und Spielwährungen, da diese Systeme potenzielle Angriffsziele darstellen. Gleichzeitig muss die Oberfläche intuitiv gestaltet werden, um auch Studierende ohne technische Vorkenntnisse eine Speilerfahrung zu bieten. Die Plattform soll darüber hinaus flexibel erweiterbar sein, sodass neue Spiele, Module oder Funktionen ohne große strukturelle Umbauten integriert werden können. Zusätzlich wird im Rahmen der Arbeit untersucht, welche Marketingmethoden angewannt werden müssen um die gewünschte Zielgruppe zu erreichen. Insgesamt besteht die zentrale Herausforderung darin, eine robuste, skalierbare und benutzerorientierte Webanwendung zu entwickeln, die sowohl technisch als auch organisatorisch langfristig betreibbar ist.

1.2 Zielsetzung

Ziel der Arbeit ist die Entwicklung eines sicheren, skalierbaren und modular aufgebauten Studentencasinos, das durch spielerische Elemente den Wettbewerb zwischen den Nutzern fördert. Dazu wird eine fiktive Spielwährung eingeführt, der „Coin“, der als zentrale Ressource für alle Spiele und Mechaniken dient. Ein übergreifendes Punktesystem soll alle Spieler anhand ihrer Coin-Anzahl in einer Rangliste absteigend sortieren und so einen dauerhaften Anreiz zur Teilnahme schaffen. Durch verschiedene Minigames erhalten die Nutzer die Möglichkeit, ihre Coins zu vermehren, wodurch Dynamik und Variation entstehen. Die Plattform wird so entworfen, dass neue Spiele ohne größere strukturelle Anpassungen integriert werden können, was langfristige Erweiterbarkeit sicherstellt. Ein weiterer

Schwerpunkt liegt auf der Sicherheit des Systems, insbesondere der korrekten und manipulationssicheren Verwaltung der Spielwährung. Zusätzlich berücksichtigt die Zielsetzung Aspekte wie Nutzerfreundlichkeit, klare Interface-Gestaltung und die Schaffung eines motivierenden Spielerlebnisses. Insgesamt soll eine Plattform entstehen, die stabil, attraktiv und technisch sauber umgesetzt ist.

1.3 Aufbau

Die Arbeit gliedert sich in mehrere Hauptkapitel, die den gesamten Projektverlauf von der Planung bis zur Reflexion strukturiert darstellen. Nach der Einleitung folgt das Kapitel Projektmanagement, in dem die gewählte Methodik, die Organisation des Teams sowie Kommunikations- und Planungsprozesse beschrieben werden. Anschließend widmet sich das Kapitel Konzeption den ermittelten Anforderungen, der technologischen Auswahl und der Ausarbeitung der Gesamtarchitektur des Systems. Die darauf folgenden Kapitel Backend und Frontend beschreiben im Detail die technische Umsetzung, die verwendeten Frameworks und die Art, wie Logik, Datenhaltung und Benutzeroberfläche miteinander interagieren. Im Kapitel Hosting wird erläutert, wie die Anwendung bereitgestellt, versioniert und deployt wird, um eine stabile und skalierbare Umgebung sicherzustellen. Danach folgt das Kapitel Projektergebnis/Reflexion, das die wesentlichen Resultate zusammenfasst und den Entwicklungsprozess hinsichtlich Effizienz, Herausforderungen und gewonnener Erfahrungen bewertet. Die Struktur der Arbeit ermöglicht eine klare Nachvollziehbarkeit der Entscheidungen sowie der technischen und organisatorischen Umsetzung des Projekts. Dadurch wird sowohl die fachliche Tiefe als auch die praktische Relevanz der Arbeit sichtbar gemacht.

2 Projektmanagement

In dem Kapitel Projektmanagement werden die organisatorischen und methodischen Grundlagen des Projekts beschrieben. Es umfasst die gewählte Projektmethodik, die Kommunikations- und Austauschprozesse innerhalb des Teams sowie die Planung des Projekts. Ziel dieses Kapitels ist es, einen klaren Überblick über die Strukturierung und Steuerung des Projekts zu geben, um die Effizienz und den Erfolg der Umsetzung sicherzustellen.

2.1 Projektmethodik

Für die Umsetzung des Projekts wurde eine Iterative, agile Projektmethode mit regelmäßigen Meetings gewählt. Diese Methode ermöglicht es, flexibel auf Änderungen zu reagieren und kontinuierlich Verbesserungen vorzunehmen. Durch kurze Entwicklungszyklen konnten Anforderungen schnell umgesetzt und getestet werden, was die Qualität des Endprodukts erhöhte. Die agile Methodik förderte zudem die Zusammenarbeit im Team, da regelmäßige Abstimmungen und Feedbackrunden eingeplant wurden. Insgesamt trug diese Vorgehensweise dazu bei, das Projekt effizient zu steuern und den Fokus auf die wichtigsten Ziele zu legen.

2.2 Austausch

Die Kommunikation und der Austausch im Projektteam erfolgten überwiegend über eine gemeinsame WhatsApp-Gruppe sowie durch persönliche Gespräche im Anschluss an Vorlesungen. Für die kontinuierliche Abstimmung kamen OneNote und verschiedene Funktionen von GitHub[1] zum Einsatz. GitHub ermöglichte es, Kommentare direkt im Code zu hinterlassen und Aufgaben eindeutig zuzuweisen. Zur Planung und Nachverfolgung einzelner Arbeitspakete wurde der „Issues“-Bereich von GitHub genutzt, der eine strukturierte Verwaltung von Milestones, Tasks und Fehlerberichten erlaubte. Auf dieser Grundlage konnten Fortschritte transparent dokumentiert und Prioritäten festgelegt werden. Ergänzend fanden wöchentliche Meetings über Discord statt, um komplexere Fragestellungen zu besprechen, Entscheidungen zu treffen und den direkten Austausch

[1] In Github wurde dazu ein öffentliches Repo erstellt: https://github.com/coan988/student-gambling-site.git

zu stärken. Zentrale Projektdokumente und relevante Informationen wurden in OneNote abgelegt, sodass alle Teammitglieder jederzeit Zugriff auf den aktuellen Wissensstand hatten. Durch die kombinierte Nutzung dieser Werkzeuge wurde eine transparente und effiziente Kommunikation ermöglicht, was wesentlich zur erfolgreichen Zusammenarbeit im Projekt beitrug.

2.3 Planung

Die Planung bildete den organisatorischen Rahmen des Projekts und strukturierte die Zusammenarbeit über den gesamten vierwöchigen Bearbeitungszeitraum. Zu Beginn wurden Zuständigkeiten definiert sowie Anforderungen und Ziele des geplanten Internetauftritts festgelegt. Darauf aufbauend entstand eine Projektstruktur, aus der sich Aufgaben, Meilensteine und zeitliche Prioritäten ableiten ließen. Die anschließende Feinplanung diente als Grundlage für die Ausarbeitung der Rollen, die Erstellung eines realistischen Zeitplans sowie die Festlegung zentraler Meilensteine. Während der Umsetzung wurde diese Planung kontinuierlich überprüft und bei Bedarf angepasst, um den engen Zeitrahmen effizient zu nutzen.

2.3.1 Rollenverteilung

Die Rollenverteilung im Projektteam wurde klar definiert, um eine effiziente Zusammenarbeit und Verantwortungsübernahme zu gewährleisten. Corvin Annen übernahm die Projektleitung und koordinierte sämtliche organisatorischen Abläufe. Dazu gehörten Terminabsprachen, die Abstimmung zwischen den Entwicklungsbereichen sowie die Überwachung des Projektfortschritts. Elijah Mossmann war für die Backend-Entwicklung verantwortlich. Sein Aufgabenbereich umfasste die Implementierung der API-Strukturen, die Anbindung der Datenbank sowie die Sicherstellung der funktionalen Logik. Timo Manz leitete die Frontend-Entwicklung. Er konzipierte die Benutzeroberfläche, setzte UI-Komponenten um und sorgte dafür, dass Funktionalität und Gestaltung konsistent ineinandergreifen. Michael Zerner betreute Hosting und Deployment. Er kümmerte sich um die Serverumgebung, richtete das Hosting ein und verantwortete die Bereitstellung der Anwendung einschließlich Tests der Live-Version. Diese Rollenverteilung gewährleistete, dass Arbeitsbereiche klar abgegrenzt waren und jedes Mitglied zielgerichtet beitragen konnte.

2.3.2 Zeitplan

Für das vierwöchige Projekt wurde ein komprimierter, aber realistischer Zeitplan ausgearbeitet. Die erste Woche war der Konzeption gewidmet: Klärung der Anforderungen, Auswahl geeigneter Technologien und Festlegung der Systemarchitektur. In den beiden folgenden Wochen erfolgte die parallele Entwicklung von Backend und Frontend, begleitet vom Aufbau der Datenbank und der Implementierung grundlegender UI-Elemente. In der letzten Woche standen Hosting, Integrationstests und Optimierungen im Vordergrund. Regelmäßige Abstimmungen erlaubten es, Fortschritte zu kontrollieren und den Zeitplan situativ anzupassen, ohne die Zieltermine zu gefährden.

2.3.3 Meilensteine

Die Meilensteine des Projekts wurden strategisch festgelegt, um den Fortschritt zu überwachen und sicherzustellen, dass die wichtigsten Ziele erreicht werden. Zu den zentralen Meilensteinen gehörten die Fertigstellung der Konzeptionsphase, die Implementierung des Backends und Frontends, die Entwicklung der Datenbankstruktur sowie die Bereitstellung der Anwendung im Hosting-Umfeld. Jeder Meilenstein markierte einen bedeutenden Fortschritt im Projektverlauf und diente als Grundlage für die nächsten Schritte. Durch die klare Definition dieser Meilensteine konnte das Team den Überblick behalten und sicherstellen, dass alle Aufgaben termingerecht abgeschlossen wurden.

3 Konzeption

Das Kapitel Konzeption beschreibt die grundlegenden Überlegungen und Entscheidungen, die der technischen Umsetzung des Projekts zugrunde liegen. Es umfasst die Analyse der Anforderungen, die Auswahl der technologischen Komponenten sowie die Ausarbeitung der Gesamtarchitektur des Systems. Ziel dieses Kapitels ist es, eine klare und nachvollziehbare Grundlage für die anschließende Entwicklung zu schaffen.

3.1 Anforderungen

Die Anforderungen an den Internetauftritt des Studentencasinos wurden sorgfältig analysiert, um eine benutzerfreundliche, sichere und skalierbare Plattform zu schaffen. Funktionale Anforderungen umfassen die Implementierung eines Punktesystems mit einer fiktiven Spielwährung („Coins“), die Integration verschiedener Minigames zur Coin-Generierung sowie die Anzeige einer Rangliste basierend auf der Coin-Anzahl der Nutzer. Darüber hinaus soll die Plattform modular aufgebaut sein, um zukünftige Erweiterungen und neue Spiele problemlos integrieren zu können. Nicht-funktionale Anforderungen beinhalten Aspekte wie Sicherheit bei der Verwaltung der Spielwährung, Performance-Optimierung für eine reibungslose Nutzererfahrung sowie eine intuitive Benutzeroberfläche. Zusätzlich wurde Wert auf eine klare und ansprechende Gestaltung gelegt, um die Zielgruppe der Studierenden effektiv anzusprechen. Insgesamt sollen diese Anforderungen sicherstellen, dass die Plattform sowohl technisch robust als auch attraktiv für die Nutzer ist.

3.2 Technologische Entscheidungen

Die technologischen Entscheidungen für die Umsetzung des Studentencasinos wurden auf Basis der Anforderungen und Zielsetzungen des Projekts getroffen. Für das Backend wurde das Django-Framework in Python gewählt, da es eine robuste Struktur für die Entwicklung von Webanwendungen bietet und eine einfache Integration mit Datenbanken ermöglicht. Django’s integriertes Authentifizierungssystem erleichtert zudem die sichere Verwaltung von Benutzerdaten.[2] Für das

[2] Dhadil u. a., „Django Unleashed: A Deep Dive into the Features and Advantages of the Django Framework“.

Frontend fiel die Wahl auf React, da es eine komponentenbasierte Architektur bietet, die eine flexible und wiederverwendbare Gestaltung der Benutzeroberfläche ermöglicht.[3] Vite wurde als Bundler eingesetzt, um schnelle Entwicklungszyklen und effizientes Build-Management zu gewährleisten.[4] Als Datenbank wurde SQLite verwendet, da es leichtgewichtig ist und sich gut für die Anforderungen des Projekts eignet, mit der Möglichkeit, bei Bedarf auf leistungsfähigere Systeme wie MySQL oder PostgreSQL zu skalieren.[5] Diese technologischen Entscheidungen tragen dazu bei, eine stabile, skalierbare und benutzerfreundliche Plattform zu schaffen.

3.3 Gesamtarchitektur

Die Gesamtarchitektur des Studentencasinos ist in verschiedene Schichten unterteilt, die jeweils spezifische Funktionen und Verantwortlichkeiten übernehmen. Das Backend bildet die Grundlage der Anwendung und ist für die Datenverarbeitung, Nutzerverwaltung und API-Bereitstellung zuständig. Es kommuniziert mit der SQLite-Datenbank, in der alle relevanten Informationen wie Nutzerprofile, Coin-Bestände und Spielstände gespeichert werden. Das Frontend ist für die Präsentation der Benutzeroberfläche verantwortlich und interagiert über eine REST-API mit dem Backend, um Daten abzurufen und zu senden. Diese Trennung von Frontend und Backend ermöglicht eine klare Strukturierung der Anwendung und erleichtert zukünftige Erweiterungen. Die Architektur ist darauf ausgelegt, Skalierbarkeit, Sicherheit und Benutzerfreundlichkeit zu gewährleisten, indem sie bewährte Technologien und Designprinzipien nutzt.

Die folgende Abbildung 1 zeigt das Klassendiagramm, das die Hauptkomponenten und deren Beziehungen innerhalb der Gesamtarchitektur des Studentencasinos darstellt.

[3] React, „React DOM Components“.
[4] Vite, „Vite: Next Generation Frontend Tooling“.
[5] SQLite, „About SQLite“.

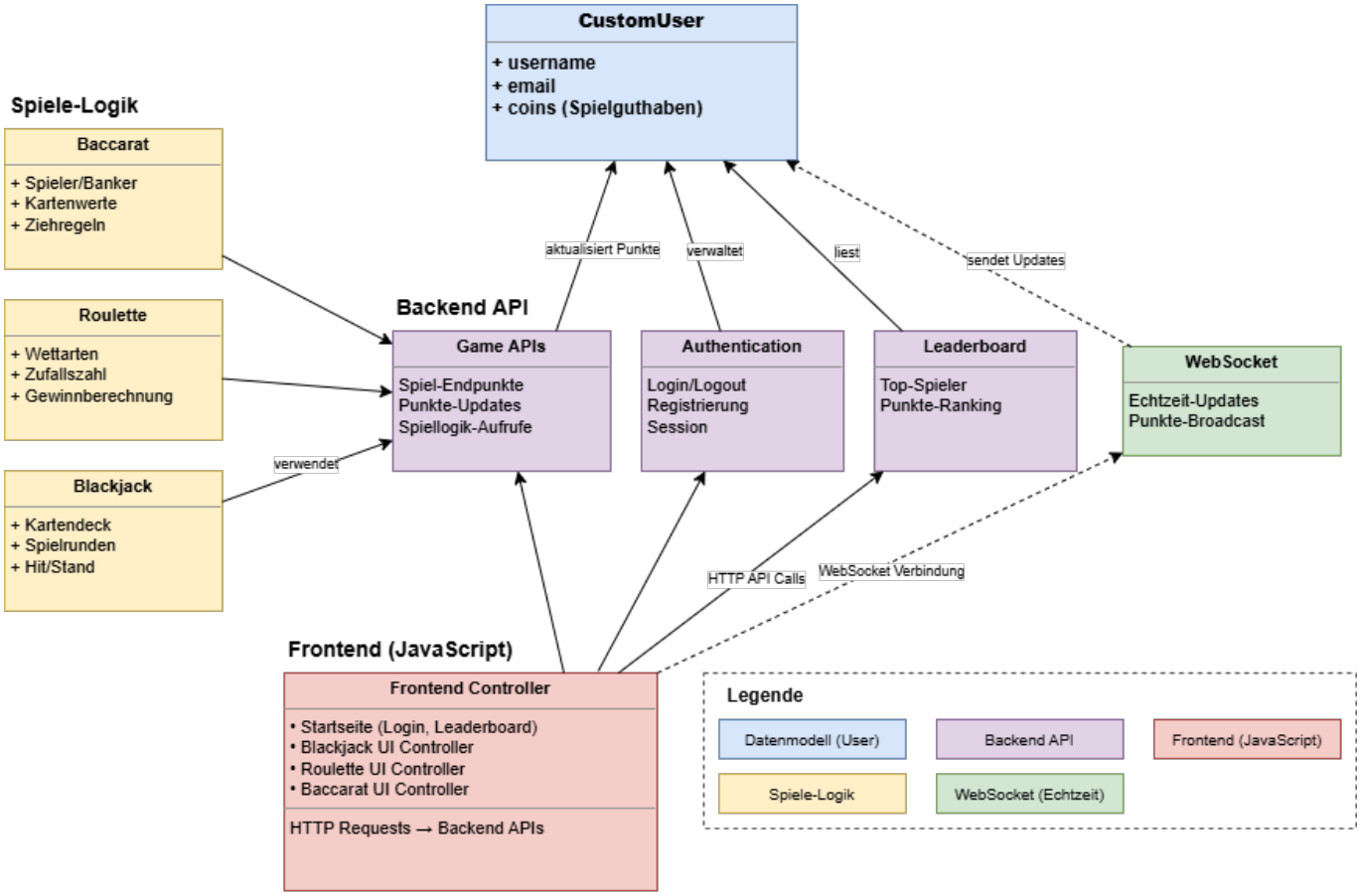

Abbildung 1: Gesamtarchitektur Internetauftritt Studentenkasino

Das Klassendiagramm illustriert die Struktur der Anwendung, indem es die wichtigsten Klassen und deren Interaktionen aufzeigt. Im Backend sind Klassen für die Nutzerverwaltung, Spielmechaniken und Datenbankinteraktionen definiert, während das Frontend Klassen für UI-Komponenten und API-Kommunikation umfasst. Diese Darstellung verdeutlicht die modulare Aufbauweise der Anwendung und die klare Trennung der Verantwortlichkeiten zwischen den verschiedenen Schichten.

4 Backend

Das Backend bildet die Grundlage für sämtliche Funktionalitäten einer Webanwendung. Es umfasst Datenverarbeitung, Kommunikation mit der Datenbank, Nutzerverwaltung, Sitzungsmanagement, die Bereitstellung von APIs für das Frontend u.v.m. Bestandteile eines Backends sind zum Beispiel Server, Programmiersprache, Framework, Package Manager, Datenbank.[6] In diesem Kapitel wird das Backend der Web-Anwendung dargestellt, wobei Implementierungsdetails, Datenbankmodellierung, API-Design und sicherheitsrelevante Funktionen erläutert werden.

4.1 Vorraussetzung

Um mit der eigentlichen Implementierung des Backends beginnen zu können, mussten noch relevante Entscheidungen bezüglich der Auswahl von oben genannten Bestandteilen getroffen werden. Als Backend-Programmiersprache fiel die Entscheidung auf Python, da sie uns bereits vertraut ist. Das genutzte Framework ist Django, da es „so ziemlich das Framework für Python ist “.[7] Django vereinfacht die Arbeit im Backend, da Funktionen wie Routing, Datenbankanbindung, Authentifizierung sowie ein Admin-Interface bereits fertig zur Verfügung stehen. Dadurch kann sich einiges an Arbeit erspart werden und direkt mit der Implementierung anwendungsspezifischer Funktionen begonnen werden. Der genutzte Package-Manager ist pip.

Zu Beginn musste Django über pip in dem Verzeichnis „backend“ von unserem Git-Repository installiert werden. Anschließend wurden in diesem Verzeichnis das Django-Projekt „casino_backend_project“ und die Django-App „casino“ erstellt. Das Django-Projekt ist die Gesamtanwendung. Hier werden globale Einstellungen, URL-Routing, Sessions oder auch die Datenbank der Anwendung bestimmt bzw. verwaltet. Die Django App dient der Implementierung einzelner Module der Anwendung, die bestimmte Funktionen beinhalten. So könnten beispielweise für das Leaderboard, die Benutzerverwaltung oder auch die Spiele selbst jeweils

[6] SuperSimpleDev, „Backend web development - a complete overview“.

[7] Programmieren-lernen, „Django Python Tutorial für Anfänger | Frontend + Backend erstellen in 60 Minuten“.

eigene Apps innerhalb des Django-Projekts erstellt werden.[8] In unserem Projekt wurde nur eine App für mehrere Funktionen erstellt. Bei größeren Anwendungen sollte dies Vermieden werden, um eine übersichtliche Projektstruktur zu gewährleisten.

4.2 API-Design und Architektur

Das Django-Rest-Framework, ebenfalls über pip im Verzeichnis backend installiert, dient als Ergänzung zu Django um eine REST-API bereitzustellen. Diese API bildet die Schnittstelle zwischen dem Frontend und dem Backend und ermöglicht eine klare Trennung zwischen Präsentationsebene und der Applikationslogik. Durch die API kann eine Anwendung auf eine Ressource (z.B. Nutzer, Session, Spielstand) einer anderen Anwendung zugreifen, indem ein HTTP-Request wie zum Beispiel POST oder GET an einen End Point der API (eine URL) gesendet wird. In der Datei urls.py der Django-App werden unsere End Points definiert. In der Datei urls.py im Django-Projekt werden die End Points in das Projekt eingebunden und es wird das Präfix /api/casino/ an sie vorangesetzt. Daraus ergibt sich nun ihr vollständiger Name. Diesen End Points liegen Funktionen zugrunde, die über diese vollständige URL aufrufbar sind. Die REST-API unseres Backendes hat die folgenden End Points mit den zugrundeliegenden Funktionen:

- /api/casino/session/: Session Prüfung und Informationen über den eingeloggten Nutzer bereitstellen
- /api/casino/register/: Registrierung neuer Nutzer inkl. Session-Erzeugung
- /api/casino/login/: Nutzer-Login inkl. Session-Erzeugung
- /api/casino/logout/: Beenden der Sitzung
- /api/casino/csrf/: Erstellt CSRF-Token bei Bedarf
- /api/casino/“Spielmechanik“/: Endpunkte zur Kommunikation mit den jeweiligen Spielmechaniken

Der auf den Request folgende Response wird dank des Django-Rest-Frameworks im JSON-Format ausgegeben und kann im Frontend verarbeitet werden.

[8] Janssen, „App“.

4.3 Datenbank und User-Modell

Die genutzte Datenbank ist SQLite, eine relationale Datenbank die standardmäßig mit Django mitinstalliert wird. SQLite erfüllt zentrale Anforderungen, die innerhalb der Entwicklung an die Datenbank aufkamen: Speicherung der Nutzer, Verwaltung des Punktestands, Speicherung von Sessions sowie die persistente Ablage relevanter Daten. SQLite ist eine kleine, aber für das Projekt vollkommen ausreichende Datenbank. Sie ist performant, sehr stabil und kann problemlos auch große Datenmengen verarbeiten. Um zu skalieren und eine höhere Anzahl paralleler Nutzer problemlos tragen zu können, sollte aber auf eine größere Datenbank wie PostgreSQL oder MySQL umgestiegen werden.

Im Verlaufe des Projekts ergab sich, dass in Abbildung 2 aus der Realität abstrahierte ER-Modell:

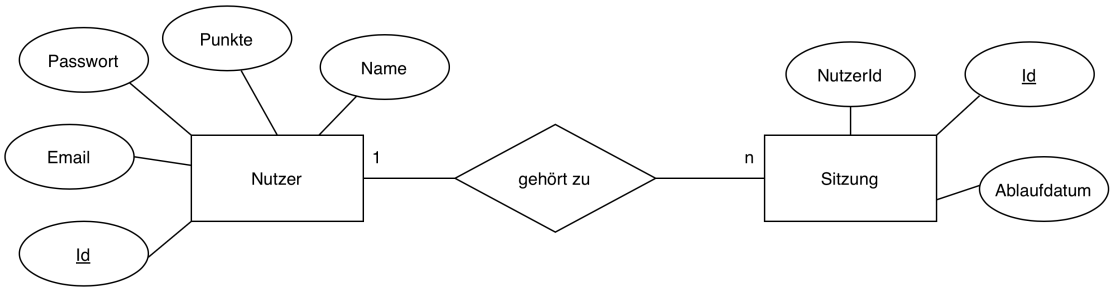

Abbildung 2: ER-Modell der Datenbank

Hierfür stellt Django ein vollständiges Authentifizierungsmodell mit standardisiertem User-Modell bereit, dass Informationen wie Benutzername, Passwort (gehasht), E-Mail oder auch berechtigungsbezogene Daten speichert. Dieses Standardmodell umfasst jedoch keine anwendungsspezifischen Attribute wie den Punktestand eines Nutzers. Um dieses Attribut zu definieren, wurde ein eigenes User-Modell auf Basis von AbstractBaseUser und PermissionsMixin implementiert. Dadurch konnte das Feld points ergänzt werden, während zentrale Django-Funktionen wie Authentifizierung, Passwort-Hashing und Session-Integration vollständig erhalten bleiben.

Das für den User benötigte Datenmodell wurde mithilfe des Object-Relational Mappings von Django in der Datei models.py im Verzeichnis casino definiert. Das Object-Relational Mapping, eine „Programmiertechnik, die den Zugriff auf relationale Datenbanken durch die Abbildung von Datenbanktabellen auf Objekte

einer Programmiersprache vereinfacht“,[9] erlaubt es, Datenbanktabellen in Form von Python-Klassen zu definieren und anschließend automatisch in die zugrunde liegende relationale Datenbank zu übertragen.[10] Durch das Session-System von Django wurde die Sitzungstabelle „django_session“ von Django automatisch erzeugt. Sie musste nicht in models.py definiert werden.

4.4 Sessions, Authentifizierung und Sicherheit

Für die Authentifizierung und Sitzungsverwaltung wird das in Django integrierte Session- und Cookie-System genutzt. Nach einer erfolgreichen Registrierung oder Anmeldung erzeugt Django automatisch ein Session-Cookie, das eine eindeutige Session-ID enthält. Dieses Cookie wird im Browser gespeichert und bei jeder API-Anfrage automatisch an das Backend übermittelt.[11] Da alle eigentlichen Session-Daten serverseitig in der Datenbank liegen, kann Django anhand der Session-ID zuverlässig den zugehörigen Nutzer identifizieren und den Anmeldestatus durchgängig aufrechterhalten. Bei einer gültigen Übereinstimmung lädt Django die spezifischen Session-Daten und authentifiziert den Nutzer.

Da das Frontend ausschließlich über Fetch-API kommuniziert, werden die Session-Cookies erst durch die Konfiguration von <credentials: „include“>
in der Datei „settings.py“ gezielt an jede Anfrage angehängt. Für potenzielles hosten über getrennte Frontend- und Backend-Server und damit einhergehend unterschiedlichen Domains, wurden die Cookie-Einstellungen (ebenfalls in settings.py) entsprechend angepasst: <SESSION_COOKIE_SAMESITE = ‚None‘> erlaubt die domainübergreifende Übertragung, während <SESSION_COOKIE_SECURE = True> sicherstellt, dass Cookies ausschließlich über HTTPS gesendet werden.

Die Sicherheit der Anwendung wird zusätzlich durch eine strenge CORS-Konfiguration in settings.py unterstützt. Diese CORS-Konfiguration stellt sicher, dass nur explizit freigegebene Frontend-Domains die Responses des Servers lesen dürfen. Ohne entsprechende CORS-Freigaben würde der Browser das Lesen der Responses durch nicht freigegebene Domains, aufgrund der Same-Origin-Policy, standardmäßig blockieren. Da das Projekt serverseitig verwaltete Session-Coo-

[9] Deinhard, „Was ist Object-Relational Mapping (ORM)“.
[10] Deinhard, „Was ist Object-Relational Mapping (ORM)“.
[11] IONOS_Redaktion, „Cookies“.

kies für die Authentifizierung nutzt, ist zusätzlich der Einsatz eines CSRF-Tokens erforderlich. Der CSRF-Mechanismus verhindert, dass externe Webseiten im Kontext einer bestehenden Session ungewollte Aktionen ausführen können. Durch entsprechend durchgeführte CSRF-Konfigurationen in der settings.py, erstellt der Server ein solches CSRF-Token und schickt es als Cookie an den Browser. Bei einem Request wird das Token nun aus dem Cookie gelesen und in den Header des Requests gegeben. Der Request wird mitsamt des Cookies an den Server geschickt. Serverseitig finden „Vergleichsverfahren„ zwischen dem Token aus dem Cookie und dem Header statt. Bei einer Übereinstimmung gilt der Request als vertrauenswürdig und wird verarbeitet.

4.5 Spiellogiken

Die Spiellogiken der Spiele Roulette, Blackjack und Baccarat wurden Im Verzeichnis „game_logic“ in der Django-App realisiert, während bestimmte Interaktionen (z.B. das Setzen der Punkte) weiterhin im Frontend stattfinden. Das Backend ist dabei für die korrekte Verarbeitung aller für die Spiele relevanten Abläufe verantwortlich. Durch die zentrale Umsetzung der Spiellogiken im Backend bleibt die Anwendung konsistent und geschützter vor clientseitigen Manipulationen, da die Berechnungen nicht im Browser des Nutzers stattfinden, sondern vom Server durchgeführt werden.

5 Frontend

Das Frontend bezeichnet den sichtbaren Teil einer Webanwendung, mit dem der Benutzer direkt interagiert. Es umfasst alles, was im Browser dargestellt wird von der Struktur der Inhalte über das Design bis hin zu interaktiven Elementen wie Buttons und Formularen.

5.1 Strategische Vorüberlegung:

Bevor die eigentliche Umsetzung der „Student Gambling Site“ begann, mussten wir uns zuerst gedanken machen, wie der Code bestmöglich strukturiert werden kann: Dabei stand besonders die User Experience im Vordergrund. Wie können wir eine bestmögliche Performance für den Nutzer gewährleisten, ohne dabei die Wartbarkeit des Codes zu gefährden? Die Herausforderung lag darin, dass nicht jeder Bereich der Webanwendung die gleichen technischen Anforderungen stellt.

Anstatt eine starre Einheitsstruktur für das gesamte Projekt zu erzwingen, haben wir uns daher dafür entschieden, flexibel an die Problematik heran zu gehen. Wir entschieden dementsprechend technisch zwischen zwei Anwendungsfällen zu entscheiden: Für einfache Unterseiten setzen wir auf einen internen Ansatz, bei dem CSS- und JavaScript-Code direkt in das Dokument eingebettet werden, um unnötige Server-Anfragen zu vermeiden und Inhalte verzögerungsfrei darzustellen.[12]

Für komplexe Hauptanwendungen hingegen, sind Übersichtlichkeit und entscheidend. Hier haben wir uns dazu entschlossen, Styles und Logik konsequent in externe Dateien auszulagern. Daher ist die startseite unserer Webanwendungin externen CSS und JavaScript Dateien zu designt insbesondere um es übersichtlich zu behalten und Änderungen schneller und weniger fehleranfällig durchführen zu können. Während wir bei den einzelnen Spielen einen Inbound Ansatz gewählt haben, da diese weniger komplex sind.

[12] MDN, „Kritischer Rendering Pfad“.

5.2 Technologien und Werkzeuge: Das Fundament des Frontends

Jede moderne Webseite, die wir täglich sehen, basiert auf dem Zusammenspiel dreier fundamentaler Sprachen: HTML, CSS und JavaScript. Sie bilden zusammen das Frontend, also den Teil der Anwendung, den die Benutzer in ihrem Browser sehen und mit dem sie interagieren können.

5.2.1 HTML

Die Grundlage jeder Website, von der einfachsten bis zur komplexesten Anwendung, bildet die HyperText Markup Language (HTML). Man kann sich HTML als das Grundgerüst oder den Bauplan einer Webanwendung vorstellen: Es definiert und strukturiert die Website, aber hat keinen Einfluss auf Design und Interaktionen. HTML ist ausschließlich dafür verantwortlich, die Struktur und den Inhalt einer Webseite festzulegen.

Der Code besteht aus einer Reihe von sogenannten Tags oder Markierungen, die dem Webbrowser mitteilen, wie der enthaltene Text zu interpretieren und darzustellen ist. Durch die korrekte Anordnung dieser Tags wird die logische Gliederung des gesamten Dokuments erstellt. HTML bestimmt somit, welche Inhalte als Überschrift, als Liste, als verlinkbarer Textabschnitt oder als Teil eines Formulars zu behandeln sind.

Ein besonders wichtiger Aspekt von modernem HTML ist die Semantik. Semantische Tags wie <header>, <nav>, <main>, <section> und <footer> geben dem Inhalt nicht nur eine optische Platzierung, sondern vor allem eine klare Bedeutung.[13]

5.2.2 CSS

Wenn HTML die Inhalte und ihre Struktur festlegt, tritt CSS (Cascading Style Sheets) in Aktion, um der Webseite ihr visuelles Erscheinungsbild zu verleihen. Es trennt die Gestaltung vollständig vom Inhalt, was eine enorme Flexibilität und Effizienz in der Webentwicklung ermöglicht. CSS ist verantwortlich für alle Design-Entscheidungen.

Dabei geht es beispielsweise um die Farbgebung und der Typografie es bestimmt die Hintergrundfarben, die Textfarben und welche Schriftarten verwendet werden,

[13] Müller, „Einstieg in HTML und CSS“.

bis hin zur präzisen Positionierung aller Elemente. Es regelt die Abstände zwischen Elementen sowie ihre Größe und wie sie zueinander auf der Seite angeordnet werden, beispielsweise nebeneinander in Spalten oder übereinander. Darüber hinaus werden alle dekorativen Elemente und visuellen Effekte über CSS gesteuert.[14]

5.2.3 JavaScript

JavaScript ist die Programmiersprache, welche dem Frontend Leben einhaucht und es interaktiv macht. Es ist das funktionale Element, das dafür sorgt, dass eine Webseite nicht nur statisch ist, sondern auf Ereignisse reagieren kann.

Die Hauptaufgabe von JavaScript liegt in der Ereignisverarbeitung und der Manipulation des Dokuments. Ein Ereignis ist jede Aktion, die ein Benutzer ausführt, wie das Klicken auf einen Button, das Ausfüllen eines Formulars oder das Bewegen der Maus über ein Element. JavaScript „hört“ auf diese Ereignisse und führt daraufhin spezifische Anweisungen aus. Klickt der Benutzer beispielsweise auf einen Registrierungs-Button, sorgt JavaScript dafür, dass die eingegebenen Daten gesammelt, validiert und weiterverarbeitet werden[15]

Durch die DOM-Manipulation kann JavaScript auf die HTML-Struktur zugreifen und diese bei Bedarf ändern, etwa um ein Popup-Fenster einzublenden, einen Ladebalken anzuzeigen oder eine Fehlermeldung direkt neben einem Eingabefeld zu platzieren.

Ein weiterer entscheidender Bereich ist die asynchrone Kommunikation mit dem Backend . Anstatt die gesamte Seite neu laden zu müssen, wenn sich ein Benutzer anmeldet oder einen Spielstand aktualisiert, sendet JavaScript die benötigten Informationen im Hintergrund an den Server. Dieses moderne Verfahren wird oft über die Fetch-API realisiert und ist der Schlüssel zu einer flüssigen, App-ähnlichen Benutzererfahrung. Der große Vorteil: Die Benutzeroberfläche (UI) bleibt für den Benutzer sichtbar und bedienbar, während die Datenübertragung stattfindet. Sobald der Server eine Antwort schickt – etwa die Bestätigung einer erfolgreichen Anmeldung – fängt JavaScript diese Antwort ab und aktualisiert dann gezielt nur

[14] Vilas, „HTML & CSS“.
[15] Ackermann, *JavaScript*.

die relevanten Teile der HTML-Seite, wie das Einblenden des Benutzernamens oder das Ausblenden des Login-Formulars.

5.3 Struktureller Aufbau der Website

Nach der Festlegung der grundlegenden technischen Strategie war der nächste wesentliche Schritt im Entwicklungsprozess die Definition der Webseiten-Struktur. Eine erfolgreiche Webanwendung zeichnet sich nicht allein durch einen fehlerfreien Programmcode aus, sondern maßgeblich durch eine intuitive Benutzerführung. Das primäre Ziel war es, den Nutzer so durch die Anwendung zu leiten, dass er zu keinem Zeitpunkt darüber nachdenken muss, wo er sich befindet oder wie er zu seinem Ziel gelangt. Um dies zu erreichen, wurde die „Student Gambling Site“ in klare logische Zonen unterteilt, die sich wie Schichten um den Kern der Anwendung – die eigentlichen Spiele – legen. Dabei lässt sich die Architektur der Webseite grundlegend in zwei Hauptbereiche differenzieren: den öffentlichen Zugangsbereich und den geschützten Nutzerbereich.

Der erste Kontaktpunkt für jeden Besucher ist die Startseite, die als öffentliche Startseite der Website funktioniert. Hier wurde bewusst das Prinzip des Minimalismus verfolgt. Da der Nutzer an dieser Stelle noch nicht angemeldet oder registriert ist, dient dieser Bereich ausschließlich der Information und der Zugangskontrolle. Die visuelle Struktur ist hier simpel gehalten, um den Fokus auf die wichtigste Aktion, die Anmeldung zu legen. Das zentrale Element ist das Authentifizierungs-Feld, das je nach Status entweder ein Login-Formular oder die Möglichkeit zur Registrierung anzeigt.

Sobald der Nutzer angemeldet ist und den geschützten Bereich betritt, ändert sich die Struktur grundlegend hin zu einem interaktiven Dashboard. Um trotz der verschiedenen Unterseiten eine hohe Konsistenz zu gewährleisten, wurde ein globales Layout-Raster entwickelt, das auf allen Seiten des internen Bereichs identisch bleibt. Dieses feste Rahmenwerk besteht aus einer permanent sichtbaren Kopfzeile und einer abschließenden Fußzeile. Die Kopfzeile ist dabei eine Art Kompass für den Nutzer. Unabhängig davon, ob er sich gerade in der Übersicht befindet oder tief in ein Spiel vertieft ist, bleibt dieses Element fixiert am oberen Bildschirmrand. Es enthält die essenziellen Navigationspunkte, um jederzeit zur Übersicht zurückkehren zu können, sowie persönliche Informationen

wie das Benutzerprofil und vor allem den aktuellen Punktestand. Gerade in einer Anwendung, die auf dem Sammeln und Setzen von Punkten basiert, ist diese ständige Sichtbarkeit des eigenen Guthabens ein entscheidender Faktor für das Nutzungserlebnis, da der Nutzer so direktes Feedback auf seine Aktionen erhält.

Der Bereich zwischen Kopf- und Fußzeile fungiert als dynamische Bühne der Webseite. Während der Rahmen statisch bleibt und Sicherheit vermittelt, wird in diesem zentralen Inhaltsbereich der eigentliche Content ausgetauscht. Das Herzstück der Navigation bildet dabei das Dashboard, welches oft auch als Lobby bezeichnet wird. Anstatt einer simplen textbasierten Liste wurde sich hier für ein modernes Kachel-Layout entschieden. Jedes verfügbare Spiel wird durch eine visuelle Karte repräsentiert. Diese Struktur gewährleistet eine hohe Skalierbarkeit für die Zukunft des Projekts: Sollten weitere Spiele hinzugefügt werden, fügen sich diese nahtlos als neue Kacheln in das bestehende Raster ein, ohne dass das grundsätzliche Design der Seite überarbeitet werden muss.

Wählt der Nutzer schließlich ein Spiel aus, verlässt er die Übersichtsebene und betritt die Detailansicht. Die Anordnung der Elemente folgt hier streng der Logik der Interaktion und der Leserichtung. Informationsrelevante Daten, wie etwa die Karten des Dealers oder der aktuelle Spieleinsatz, werden im oberen Sichtfeld platziert, während sich die Steuerelemente für die Interaktion im unteren Bereich befinden.

5.4 Herausforderungen: Sicherheit und flüssige Bedienung

Ein kritischer Aspekt bei der Entwicklung der „Student Gambling Site“ war die sichere Handhabung von Benutzerdaten und der reibungslose Ablauf. Da es sich um eine Anwendung mit echten Benutzerkonten und Punkteständen handelt, reicht ein hübsches Design allein nicht aus. Im Hintergrund muss eine Kommunikation stattfinden, die sowohl sicher als auch für den Nutzer unsichtbar ist.

5.4.1 Absicherung der Datenübertragung

Die Kommunikation zwischen dem Browser des Nutzers und unserem Server muss geschützt sein. Eine zentrale Herausforderung war es, sicherzustellen, dass Aktionen wie das Einloggen oder das Registrieren wirklich vom berechtigten Nutzer stammen und nicht von außen manipuliert werden können. Dafür haben wir einen automatischen Sicherheitsmechanismus eingebaut. Das Skript liest im

Hintergrund einen speziellen digitalen „Schlüssel“ (ein sogenanntes Token) aus, der dem Nutzer beim Besuch der Seite zugewiesen wird. Dieser Schlüssel wird bei jeder wichtigen Aktion automatisch mitgesendet. Der Server prüft dann, ob der Schlüssel passt. So stellen wir sicher, dass alle Anfragen korrekt zugeordnet werden, ohne dass der Nutzer sich um technische Details kümmern muss.

5.4.2 Intelligente Benutzeroberfläche ohne Wartezeiten

Ein modernes Nutzungserlebnis zeichnet sich dadurch aus, dass die Seite nicht bei jedem Klick komplett neu geladen werden muss, was oft zu störendem Flackern führt. Wir haben dies durch eine Hintergrund-Prüfung gelöst. Statt dem Nutzer beim Aufrufen der Seite pauschal das Anmeldeformular zu zeigen, prüft das Programm zunächst unbemerkt im Hintergrund, ob der Nutzer vielleicht schon eingeloggt ist. Ist das der Fall, passt sich die Oberfläche sofort an: Die Anmelde-Maske verschwindet und wird nahtlos durch eine persönliche Begrüßung und den aktuellen Punktestand ersetzt. Das bestätigt unseren hybriden Ansatz: Die Seite fühlt sich schnell und flüssig an, fast wie eine installierte App, da alle Aktualisierungen sofort sichtbar werden, ohne den Lesefluss zu unterbrechen.

6 Evaluierung der Hosting-Strategie und Umsetzung der Systemarchitektur

Der Übergang von einer lokalen Entwicklungsumgebung hin zu einem produktiven System ist ein entscheidender Schritt in der Softwareentwicklung. Dieses Kapitel beleuchtet den Deployment-Prozess der Casino-Applikation. Dabei stehen nicht nur die technischen Schritte im Vordergrund, sondern auch die Auswahl der passenden Infrastruktur, der Umgang mit Sicherheitsanforderungen sowie die Anpassung der Software an die Gegebenheiten der Hosting-Plattform.

6.1 Anforderungsanalyse und Ziel der Veröffentlichung

Bereits zu Beginn des Projekts wird festgelegt, dass die Anwendung nicht auf die lokale Umgebung („localhost“) beschränkt bleiben soll. Das Ziel ist es, die Webanwendung öffentlich über das Internet zugänglich zu machen. Dies erfüllt zwei wesentliche Funktionen: Zum einen lässt sich so eine realistische Nutzungssituation simulieren, zum anderen kann überprüft werden, wie sich das System verhält, wenn mehrere Nutzer von extern darauf zugreifen. Erst dieser Schritt belegt, dass das entwickelte Django-Backend und das Frontend stabil zusammenarbeiten.

6.1.1 Analyse der Option Self-Hosting

Zunächst wird geprüft, ob ein eigener Server betrieben werden kann, beispielsweise durch die Nutzung eines alten PCs als Linux-Server. Dieser Ansatz wäre aus Lernperspektive interessant, um Erfahrungen mit Server-Hardware und Netzwerken zu sammeln. Nach genauerer Betrachtung wird diese Option jedoch aus folgenden Gründen verworfen:

- Sicherheit: Ein öffentlicher Server im Heimnetzwerk erfordert das Öffnen von Ports am Router. Ohne professionelle Sicherheitsvorkehrungen (wie eine separate DMZ) macht dies das private Netzwerk anfällig für Angriffe von außen.
- Erreichbarkeit: Private Internetanschlüsse haben oft wechselnde IP-Adressen. Um die Seite dauerhaft erreichbar zu machen, wäre ein Dynamic-DNS-Dienst nötig, was den Aufwand erhöht.
- Ausfallsicherheit: Herkömmliche PC-Hardware bietet keine Ausfallsicherheit. Ein Stromausfall oder Hardwaredefekt würde sofort dazu führen, dass die Seite

nicht mehr erreichbar ist. Zudem würde die Wartung des Betriebssystems wertvolle Zeit kosten, die für die Programmierung fehlt.

6.1.2 Entscheidung für PythonAnywhere

Aufgrund der Nachteile des Eigenbetriebs fällt die Wahl auf einen professionellen „Platform-as-a-Service“ (PaaS)-Anbieter. Nach einem Vergleich verschiedener Optionen entscheidet sich das Projektteam für PythonAnywhere. Diese Wahl bietet entscheidende Vorteile für das vorliegende Projekt:

- Spezialisierung: Im Gegensatz zu leeren Servern (VPS), die komplett selbst eingerichtet werden müssen, bietet PythonAnywhere eine Umgebung, die bereits perfekt auf Python und Django abgestimmt ist. Das erleichtert die Einrichtung erheblich.
- Fokus auf den Code: Da sich der Anbieter um Hardware und Betriebssystem kümmert, kann der Fokus voll auf der Entwicklung der Applikation liegen.
- Kosten: Für ein Schulprojekt ist der kostenlose „Beginner“-Plan ideal. Er hat zwar Einschränkungen (z. B. keine eigene Domain), reicht aber völlig aus, um die Funktionsfähigkeit der App zu demonstrieren.

6.2 Technische Umsetzung und Konfiguration

Nach der Wahl des Anbieters folgt die technische Einrichtung. Dies umfasst den Upload des Codes, die Installation von Bibliotheken und die Konfiguration des Servers.

6.2.1 Versionskontrolle mit Git

Um den Code sicher auf den Server zu übertragen, wird Git verwendet. Da GitHub keine Passwörter mehr für die Befehlszeile unterstützt, wird ein „Personal Access Token“ eingerichtet. Dies erlaubt eine sichere Verbindung zwischen dem Server und dem Code-Repository. Der Ablauf ist dabei klar definiert: Änderungen werden lokal entwickelt und in den Hauptzweig („main branch“) geladen. Auf dem Server wird der aktuelle Stand dann heruntergezogen. Dabei muss besonders auf die Konfigurationsdatei (settings.py) geachtet werden, da sich die Einstellungen für die lokale Umgebung und den Server unterscheiden. Sensible Daten wie Sicherheitsschlüssel werden dabei bewusst nicht über Git geteilt, um die Sicherheit zu gewährleisten.

6.2.2 Datenbank Initialisierung und Funktionstests

Bevor die Anwendung öffentlich zugänglich gemacht wird, müssen administrative Zugänge eingerichtet und die Lauffähigkeit bestätigt werden. Hierfür werden folgende Befehle in der Bash Konsole des Servers ausgeführt:

- Administrator Zugang anlegen: Der Befehl python3 manage.py createsuperuser wird genutzt, um einen administrativen Benutzer mit vollen Rechten in der SQLite Datenbank anzulegen. Dieser Schritt ist notwendig, um später Zugriff auf das Django Admin Panel (/admin) zu erhalten, worüber Nutzerdaten verwaltet und Logs eingesehen werden können.
- Manueller Funktionstest: Mit python3 manage.py runserver wird der integrierte Entwicklungsserver temporär in der Konsole gestartet. Obwohl die produktive Auslieferung später über das WSGI Interface erfolgt, dient dieser Schritt als wichtiger Test. Er verifiziert, dass die Installation fehlerfrei durchlief, alle Datenbankmigrationen angewandt wurden und die Anwendung prinzipiell startet, ohne abzustürzen.

6.2.3 Konfiguration der Statischen Dateien (Static Files Mapping)

Ein wichtiger Schritt für die Performance und die korrekte Architektur der Anwendung ist das sogenannte Static Files Mapping. Da der Python Anwendungsserver (WSGI) primär für die Verarbeitung von Logik zuständig ist, soll er nicht mit der Auslieferung statischer Dateien (Bilder, CSS, JavaScript) belastet werden. Dafür werden spezifische URL Pfade direkt auf Ordner im Dateisystem gemappt:

- URL /static/: Verweist auf /backend/staticfiles. Hierhin wurden zuvor mit hilfe des Bash Befehls „python3 manage.py collectstatics“ alle Design Dateien des Django Admin Interfaces gesammelt. Dies garantiert, dass das Backend Design korrekt geladen wird.
- URL / (Root): Verweist auf /frontend/. Dies ist die technische Umsetzung der Same Origin Strategie, welche in Kapitel 6.5 noch näher erläutert wird. Anfragen an die Hauptseite werden nicht von Django verarbeitet, sondern der Webserver liefert direkt die starting_page.html und die Assets des Frontends aus.

Durch diese Konfiguration übernimmt der performante Nginx Webserver des Hosters die Auslieferung der Dateien, was die Python Prozesse für API Anfragen freihält.

6.3 Herausforderungen durch CORS und CSRF

Eine der größten Hürden beim Deployment sind die Sicherheitsmechanismen moderner Browser, insbesondere „Cross-Origin Resource Sharing“ (CORS). Anfangs treten Probleme beim Login auf, da Frontend und Backend technisch gesehen auf unterschiedlichen Adressen laufen. Der Browser blockiert daher die Kommunikation. Um dies zu lösen, wird das Paket django-cors-headers installiert. Es wird so konfiguriert, dass Anfragen von vertrauenswürdigen Quellen (sowohl lokal als auch von der Produktions-Domain) erlaubt sind. Zusätzlich müssen die Einstellungen für „Cross-Site Request Forgery“ (CSRF) angepasst werden, damit das Backend auch schreibende Zugriffe (wie das Speichern von Daten) akzeptiert. Auch die Einstellungen für Cookies müssen gelockert werden, damit der Login über verschiedene Domains hinweg funktioniert.

6.4 Wechsel zur Same-Origin-Strategie

Die finale Systemarchitektur ist das Ergebnis eines iterativen Prozesses, der durch sicherheitstechnische Hürden bei der Kommunikation zwischen Frontend und Backend geprägt ist.

6.4.1 Problematik des hybriden Betriebs

Zunächst wird der Plan verfolgt, lediglich das Backend auf PythonAnywhere zu veröffentlichen, während das Frontend weiterhin in der lokalen Entwicklungsumgebung (localhost:3000) betrieben wird. Dieser hybride Ansatz soll es ermöglichen, Änderungen an der Benutzeroberfläche schnell zu testen, während bereits auf die produktive Datenbank zugegriffen wird. Dieser Aufbau erweist sich jedoch als technisch kaum realisierbar. Trotz umfangreicher Anpassungen der CORS-Header (django-cors-headers) blockieren moderne Browser die Kommunikation. Das Hauptproblem liegt in den strengen Sicherheitsrichtlinien für Cookies: Da das Backend über eine verschlüsselte HTTPS-Verbindung antwortet, das lokale Frontend jedoch über unverschlüsseltes HTTP läuft, werden die für den Login essenziellen Session-Cookies vom Browser verworfen.

6.4.2 Scheitern der lokalen HTTPS-Simulation

Um diese Diskrepanz zu beheben, wird versucht, die lokale Umgebung ebenfalls auf HTTPS umzustellen. Hierfür werden mittels OpenSSL eigene Sicherheitszertifikate erstellt und in den lokalen Entwicklungsserver eingebunden. Auch

dieser Lösungsansatz führt nicht zum Erfolg. Da die selbst erstellten Zertifikate von Browsern nicht als vertrauenswürdig eingestuft werden, entstehen weiterhin Warnmeldungen und Blockaden bei den API-Anfragen. Der administrative Aufwand, um dem Browser diese Zertifikate „aufzuzwingen“, steht in keinem Verhältnis zum Nutzen.

6.4.3 Umsetzung der Same-Origin-Strategie

Aufgrund dieser persistierenden Komplikationen erfolgt eine strategische Änderung: Das Frontend wird ebenfalls auf die PythonAnywhere-Plattform migriert. Der kompilierte Frontend-Code wird auf den Server geladen und so konfiguriert, dass er unter derselben Domain wie das Backend ausgeliefert wird. Nach Anpassung der Pfade (Mapping der index.html auf die Root-URL) ist das System sofort voll funktionsfähig. Die technische Überlegenheit dieser Lösung basiert auf dem Same-Origin-Prinzip. In der vorherigen Konstellation stufte der Browser Frontend und Backend als unterschiedliche Ursprünge ein (Cross-Origin), da sie sich in Domain und Port unterschieden. Dies löste restriktive Sicherheitsprüfungen aus. Durch das Hosting auf demselben Server teilen sich Frontend und Backend nun:

- Das Protokoll (HTTPS)
- Die Domain (michi22.pythonanywhere.com)
- Den Port (443)

Da diese drei Parameter identisch sind, betrachtet der Browser die Kommunikation als vertrauenswürdig („Same-Origin“). Komplizierte Ausnahmeregeln für Cookies oder Preflight-Requests entfallen vollständig, was die Stabilität der Anwendung dauerhaft gewährleistet.

6.5 Umgang mit technischen Einschränkungen (WebSockets)

Ein zentrales Feature der Applikation ist die Aktualisierung des Punktestandes ohne manuellen Reload der Seite, um den Spielfluss nicht zu unterbrechen. Das ursprüngliche Architekturkonzept sah hierfür die Nutzung von WebSockets (via Django Channels und Redis) vor. Diese Technologie ermöglicht eine bidirektionale, persistente Verbindung, über die der Server Änderungen in Echtzeit an den Client „pushen“ kann.

6.5.1 Technische Restriktionen der Hosting-Umgebung

Während der Inbetriebnahme auf PythonAnywhere stellte sich heraus, dass der gewählte kostenlose Hosting-Plan keine Unterstützung für den Redis-Server bietet. Da Redis als Message-Broker für Django Channels essenziell ist, führten die WebSocket-Handshakes zu Server-Fehlern (HTTP 500). Eine asynchrone Server-Push-Kommunikation war unter diesen infrastrukturellen Gegebenheiten somit nicht realisierbar.

6.5.2 Strategiewechsel: Client-Side Short-Polling

Um die Konsistenz der Punktestände dennoch zu gewährleisten, wurde die Architektur auf ein Client-Pull-Verfahren umgestellt. Konkret wurde ein Short-Polling-Mechanismus mittels JavaScript implementiert. Anstatt auf ein Signal des Servers zu warten, fragt der Browser nun aktiv in regelmäßigen Intervallen den aktuellen Status ab. Die technische Umsetzung erfolgt spezifisch in den Template-Dateien starting_page.html und blackjack.html. Dort wird die JavaScript-Funktion setInterval genutzt, um alle 3000 Millisekunden (3 Sekunden) eine asynchrone Anfrage an den API-Endpunkt zu senden. Der Code-Ablauf gestaltet sich wie folgt:

- Trigger: Die Funktion updatePointsPeriodically initiiert den Timer.
- Request: Es erfolgt ein fetch-Aufruf an https://michi22.pythonanywhere.com/api/casino/session/.
- Authentifizierung: Da Frontend und Backend auf derselben Domain liegen („Same-Origin“), wird der Session-Cookie automatisch mitgesendet und vom Server zur Identifikation des Nutzers verwendet.
- Update: Die JSON-Antwort des Servers enthält den aktuellen Kontostand, welcher anschließend per DOM-Manipulation in die HTML-Anzeige injiziert wird.

6.6 Zusammenfassung der Architektur

Die finale Version auf PythonAnywhere ist eine stabile und wartbare Lösung. Statische Dateien und das Frontend werden direkt ausgeliefert, während die Geschäftslogik über die API läuft. Als Datenbank kommt das dateibasierte SQLite zum Einsatz, was für diese Projektgröße ideal ist. Der Prozess zeigt, dass Deployment mehr ist als nur das Kopieren von Dateien. Es erfordert das Verständnis

für Server-Architekturen und die Fähigkeit, theoretische Konzepte flexibel an die realen technischen Möglichkeiten anzupassen.

7 Fazit und Ausblick

Abschließend lässt sich sagen, dass die Entwicklung der Webanwendung für das Studentencasino einen erfolgreichen Verlauf genommen hat, insbesondere in Bezug auf die Skalierbarkeit und Benutzerfreundlichkeit. Das Projekt hat es ermöglicht, die Herausforderungen moderner Webentwicklung zu meistern, insbesondere in der Implementierung von Echtzeit-Interaktionen und der sicheren Verwaltung von Benutzerdaten. Ein wesentlicher Aspekt war dabei die Einführung der Spielwährung „Coin“, die als zentrale Ressource die Interaktivität und den Wettbewerb zwischen den Nutzern fördert. Auch die modulare Struktur des Systems stellt sicher, dass zukünftige Erweiterungen, wie neue Spiele und Funktionen, problemlos integriert werden können.

Die Sicherheit des Systems, insbesondere in Bezug auf die Speicherung und Verarbeitung der Spielwährung, wurde durch fundierte technische Entscheidungen gewährleistet. Ein weiteres Highlight des Projekts war die benutzerfreundliche Gestaltung der Oberfläche, die sicherstellt, dass auch nicht-technische Nutzer ohne Schwierigkeiten die Anwendung verwenden können.

Im Hinblick auf das Marketing wurde ein Instagram-Account erstellt,[16] um das Projekt bekannt zu machen und die Zielgruppe anzusprechen. Dennoch gibt es noch offene Aufgaben im GitHub-Repository, die weiterhin bearbeitet werden können, um die Plattform weiter zu optimieren und zusätzliche Funktionen zu integrieren. Der agile Entwicklungsansatz hat dabei nicht nur eine schnelle Umsetzung ermöglicht, sondern auch die Flexibilität bewiesen, auf Herausforderungen effizient zu reagieren.

Zusammenfassend lässt sich feststellen, dass das Projekt eine solide Grundlage für die Umsetzung eines skalierbaren und sicheren Webportals für spielerische Elemente und Wettbewerbsmechanismen bietet. Für die Zukunft sind regelmäßige Wartungs- und Erweiterungsarbeiten erforderlich, um das System aktuell zu halten und den wachsenden Anforderungen gerecht zu werden.

[16] der Instagramm Account heißt: studygamb1ing (https://www.instagram.com/studygamb1ing?igsh=MWMwYWE1Z3Nnd2VjMg==)

Literaturverzeichnis

[1] S. Dhadil *u. a.*, „Django Unleashed: A Deep Dive into the Features and Advantages of the Django Framework“, *ResearchGate preprint*, 2024, [Online]. Verfügbar unter: https://www.researchgate.net/publication/384557744_Django_Unleashed_A_Deep_Dive_into_the_Features_and_Advantages_of_the_Django_Framework

[2] React, „React DOM Components“, *React*, 2025, [Online]. Verfügbar unter: https://react.dev/reference/react-dom

[3] Vite, „Vite: Next Generation Frontend Tooling“, *Vite*, 2025, [Online]. Verfügbar unter: hhttps://vite.dev/guide/

[4] SQLite, „About SQLite“, *SQLite*, 2025, [Online]. Verfügbar unter: https://www.sqlite.org/about.html

[5] SuperSimpleDev, „Backend web development - a complete overview“, *Youtube*, 2021, [Online]. Verfügbar unter: https://www.youtube.com/watch?v=XBu54nfzxAQ

[6] Programmieren-lernen, „Django Python Tutorial für Anfänger | Frontend + Backend erstellen in 60 Minuten“, *Youtube*, 2022, [Online]. Verfügbar unter: https://www.youtube.com/watch?v=Tl0RsoMMSic

[7] i. Janssen, „App“, *ingo-janssen*, 2022, [Online]. Verfügbar unter: https://ingo-janssen.de/

[8] F. Deinhard, „Was ist Object-Relational Mapping (ORM)“, *it-schulungen*, 2025, [Online]. Verfügbar unter: https://www.it-schulungen.com/wir-ueber-uns/wissensblog/was-ist-object-relational-mapping-orm.html

[9] IONOS_Redaktion, „Cookies“, *IONOS*, 2022, [Online]. Verfügbar unter: https://www.ionos.de/digitalguide/hosting/hosting-technik/was-sind-cookies/

[10] MDN, „Kritischer Rendering Pfad“, *MDN Web Docs*, 2024, [Online]. Verfügbar unter: https://developer.mozilla.org/de/docs/Web/Performance/Guides/Critical_rendering_path

[11] P. Müller, „Einstieg in HTML und CSS“, *pmüller*, 2022, [Online]. Verfügbar unter: https://pmueller.de/semantisches-html-gibt-inhalten-bedeutung/

[12] M. Vilas, „HTML & CSS“, *DHBW-Stuttgart*, [Online]. Verfügbar unter: https://wwwlehre.dhbw-stuttgart.de/~vilas/webeng-I/WebEng_HTML-CSS.pdf

[13] P. Ackermann, *JavaScript*. Rheinwerk Verlag, 2023, S. S.29.