

Projekt:Software Engineering

Projekt-Titel:	Manga-Melody
Name:	Rahil Chughtai
Matrikelnummer:	IU14099395
Abschluss:	Informatik, M. Sc.
E-Mail:	rahil.chughtai@iu-study.org
GitHub-Link:	https://github.com/rahilchughtai/manga-melody
Website Domain	https://manga-melody.web.app/
Kursnummer:	DLMCSPSE01_D
Dozent:	Prof. Dr. Markus Kleffmann

Projektdokumentation

Einleitung

Mangas sind eine besondere Form von Comics und Graphic Novels, die ihren Ursprung in Japan haben und durch ihre einzigartigen künstlerischen Stile sowie innovativen Erzähltechniken weltweit an Popularität gewonnen haben. Sie beeinflussen zahlreiche andere Kunstformen und Medienkulturen. Mangas dienen nicht nur der Unterhaltung, sondern bieten häufig tiefgründige Reflexionen über kulturelle, soziale und philosophische Themen. In ihren facettenreichen Geschichten verbinden sie komplexe Charakterentwicklungen mit gesellschaftsrelevanten Fragen und sprechen dadurch ein breites Publikum an – von Kindern bis zu Erwachsenen.

Getrieben durch Faktoren wie der zunehmenden Globalisierung, dem wachsenden Einfluss der Popkultur und technologischen Fortschritten im Bereich des digitalen Publizierens und der Distribution, erreichte der globale Manga-Markt im Jahr 2023 einen Wert von 14,7 Milliarden USD. Prognosen zufolge soll dieser Markt bis 2030 auf 42 Milliarden USD anwachsen (Manga Global Market Report, 2024).

Ein solches Wachstum an Popularität benötigt Plattformen, die dieser Nachfrage gerecht werden. Das folgende Projekt dient als ein Prototyp für eine Applikation für Manga Fans, die dieser Nachfrage gerecht wird, indem eine Plattform bereitgestellt wird, mit der Nutzer ihre Lieblingsmangas einfach finden und erwerben können.

Projektübersicht

Ziel des Projektes ist die Erstellung einer modernen, E-Commerce basierten Web-Applikation “MangaMelody”, die eine Online-Plattform für das Entdecken, Suchen und Kaufen von Mangas bieten soll. Die Applikation soll öffentlich zugänglich sein und vor allem mit einer intuitiven User Experience, einem modernen Look, und mit einer großen Auswahl an Mangas dem Nutzer in die faszinierende Welt der kunstvollen japanischen Illustrationen einladen.

Es folgt eine knappe Übersicht der Hauptfunktionalitäten der Web-Anwendung, die in dem Anforderungsdokument detailliert beschrieben werden:

Hauptfeatures - Kurzüberblick

- Manga Suche
- Manga Favoriten speichern
- Registrierung/Einloggen
- Kauffunktion mit Warenkorb, Bestellung und Rechnung

Risikomanagement

Zunächst werden die Risiken ermittelt und erläutert. Anschließend werden die Risiken in einer Risikomatrix mit einer Eintrittswahrscheinlichkeit und einer Schadenshöhe eingeordnet. Zuletzt werden für jedes Risiko Aktionen dokumentiert, mit denen das Risiko mitigiert werden kann.

Die folgenden Risiken wurden ermittelt:

Risiko 1: Ausfall der Manga-Schnittstelle:

Die Applikation besitzt eine Abhängigkeit zu einer Externen API, die die Manga-Daten liefert. Es besteht ein Risiko, dass diese API temporär oder auch langfristig ausfällt.

Risiko 2: Zeitüberschreitung:

Es besteht ein Risiko, dass die Zeitplanung nicht eingehalten wird, und es somit zu einer Zeitüberschreitung kommt.

Risiko 3: Herausforderungen bei der Integration des Logins:

Da Login-Funktionalitäten oft komplex sind und viele Abhängigkeiten haben, besteht ein Risiko, dass die Implementierung des Logins länger dauert als geplant.

Risikomatrix

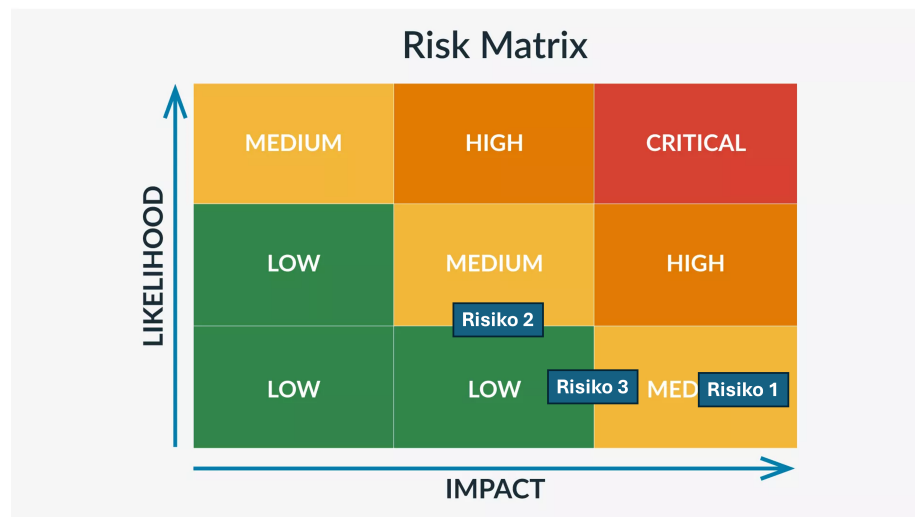


Figure 1: Risikomatrix mit den ermittelten Risiken.

Risiko 1 wurde als relativ unwahrscheinlich bewertet, da es sich um eine wohl dokumentierte und populäre Schnittstelle handelt. Ein Ausfall ohne Mitigation wäre jedoch schwerwiegend, da ein Großteil der Applikation auf die API angewiesen ist.

Mitigation: Bei der Implementierung der Anbindung der Schnittstelle wird ein Mechanismus eingebaut, der im Falle einer Fehler-Response der Schnittstelle eine lokal abgespeicherte Untermenge der Manga Daten zurückgibt. Somit wäre die Applikation noch eingeschränkt nutzbar, bis der Ausfall vorbei ist. Außerdem soll bei der Implementierung auf Error-Handling geachtet werden, um transparent gegenüber dem Nutzer zu sein.

Risiko 2 wurde als relativ unwahrscheinlich bewertet, da der Scope des Projektes klar definiert und eingeschränkt wurde. Die Auswirkungen wären auch mild, da es keine externe Abhängigkeit oder Deadline gibt.

Mitigation: Bei der Zeitplanung wird etwas Puffer berücksichtigt und eingebaut für unerwartete Fehler und Verzögerungen.

Risiko 3 wurde als relativ unwahrscheinlich bewertet, da die Implementation des Logins über externe Anbieter wie Google-Auth gelöst werden können. Falls es zu Verzögerungen kommt, wäre der Schaden ebenfalls mild, da es keine externe Abhängigkeit gibt.

Mitigation: Bei der Implementierung des Logins wird auf externe Anbieter wie Google-Auth zurückgegriffen, um die Komplexität zu reduzieren und die Implementierung zu beschleunigen.

Zeitplanung

Bei der Zeitplanung wurde für jedes Arbeitspaket eine Dauer in Tagen geschätzt. Anschließend wurde ein Gantt-Diagramm erstellt, welches diese Zeitplanung visualisiert.

Arbeitspakete	Dauer in Tagen
Initiales App Set Up	7
Integration der Manga Api	4
Landing-Page	4
Such Funktion	5
Datenbank-Anbindung	4
Login	5
Favoriten	4
Warenkorb	5
Bezahlvorgang	5
Bestellungsübersicht	4

Arbeitspakete	Dauer in Tagen
Testing	5
Puffer für Bug Fixes	7
Puffer für Design und UX	7
Verbesserungen	
App Deployment	5

Gantt-Diagramm

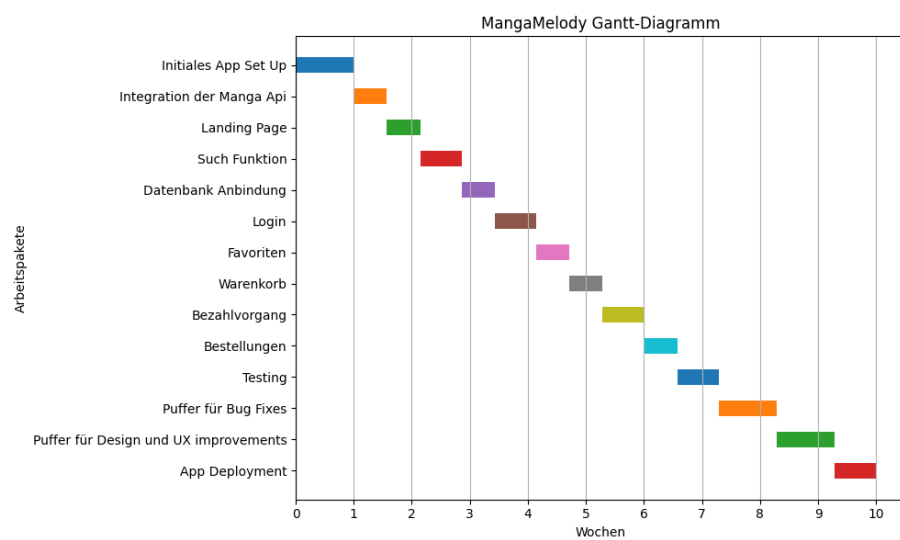


Figure 2: Gantt-Diagramm erstellt mit Python und Matplotlib.

Anforderungsdokument

Im Folgenden werden die Anforderungen näher spezifiziert anhand von

- Einer Stakeholder Analyse
- Einer Dokumentation der funktionalen Anforderungen
- Einer Dokumentation der nichtfunktionalen Anforderungen

Stakeholder-Analyse

Ziel- und Benutzergruppen

Die geplante Applikation richtet sich an mehrere Stakeholder, die in verschiedene Kategorien eingeteilt werden können. Im Folgenden werden die wichtigsten Zielgruppen und ihre jeweiligen Bedürfnisse sowie Interessen beschrieben:

Manga-Fans Die primäre Zielgruppe sind Manga-Enthusiasten, die regelmäßig Mangas konsumieren und nach einer Plattform suchen, die das Auffinden und den Erwerb von Titeln vereinfacht. Diese Gruppe ist breit gefächert und umfasst:

- **Gelegenheitsleser:** Personen, die Manga als gelegentliche Unterhaltung genießen.
- **Hardcore-Fans:** Leidenschaftliche Sammler und regelmäßige Leser, die über Neuerscheinungen, Raritäten und spezial Editionen informiert bleiben möchten.

Demografische Merkmale

Laut Media Guide, 2019 sind 90 % der Manga-Leser männlich und 10 % weiblich. Die Altersverteilung der Manga-Leser ist gezeigt in der folgenden Grafik.

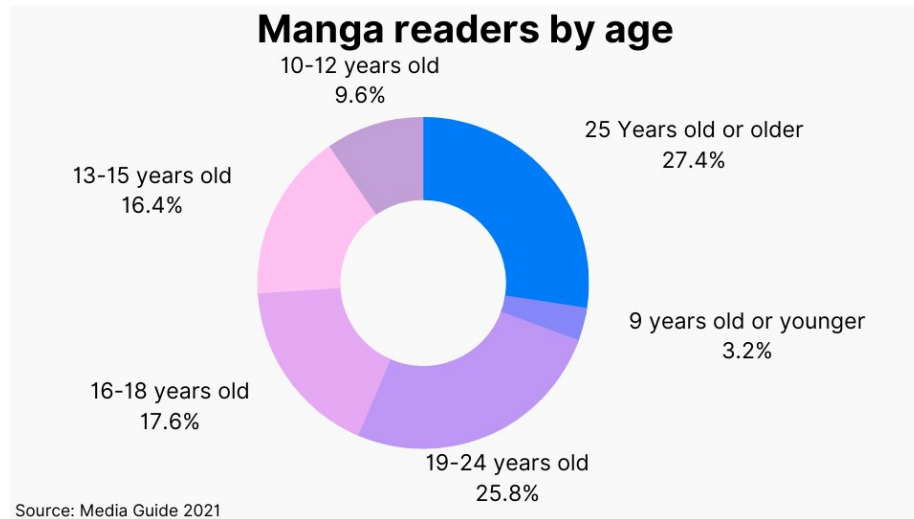


Figure 3: Manga-Leser nach Altersgruppe (Ferjan, 2024).

Im Folgenden ebenfalls dargestellt als Tabelle (Ferjan, 2024).

Altersgruppe	Anteil
25 years old or older	27.4 %
19-24 years old	25.8 %
16-18 years old	17.6 %
13-15 years old	16.4 %
10-12 years old	9.6 %
9 years old or younger	3.2 %

Somit ist die Hauptzielgruppe der Applikation männlich und zwischen 19 und 24 Jahren alt.

Bedürfnisse und Interessen:

- Schnelle und einfache Such- und Filteroptionen (z. B. nach Genre, Autor, Erscheinungsdatum).
- Bequeme Kaufmöglichkeiten.
- Modernes und zeitgemäßes Design.

Künstler und Autoren Manga-Schaffende, darunter Mangaka und Illustratoren, sind indirekte Stakeholder, die von der Verbreitung ihrer Werke profitieren.

Bedürfnisse und Interessen:

- Sichtbarkeit ihrer Werke bei einem breiten Publikum.

- Feedback und Interaktion mit den Fans.
- Faire Monetarisierung.

Zusammenfassung der Hauptinteressen

Stakeholder	Interessen/Bedürfnisse
Manga-Fans	Bequemer Zugang, Personalisierung, Kauf- und Lesemöglichkeiten
Künstler und Autoren	Sichtbarkeit, Fan-Feedback, faire Monetarisierung

Diese Analyse hilft, die Anforderungen aller Stakeholder besser zu verstehen und die Applikation optimal an die Erwartungen anzupassen.

Funktionale Anforderungen

Im folgenden werden die in der Projektdokumentation genannten Features erläutert und beschrieben in Form von User-Stories mit der Struktur

„Als [Rolle]
möchte ich [Aktion/Wunsch],
damit [Nutzen/Ziel]“.

Manga-Suche

- Als **Manga-Fan möchte ich** effizient nach Mangas suchen können, **damit** ich schnell die gewünschten Titel finde.
- Als **Manga-Fan möchte ich** Filteroptionen nutzen können, **damit** ich die Suchergebnisse nach meinen Vorlieben eingrenzen kann.
- Als **Manga-Fan möchte ich** eine Pagination-Funktion haben, **damit** ich bequem durch die Suchergebnisse blättern kann.

Favoriten

- Als **Manga-Fan möchte ich** meine Lieblingsmangas als Favoriten speichern können, **damit** ich sie schnell wiederfinden kann.

Sign Up/Log in

- Als **neuer Benutzer möchte ich** mich mit meiner E-Mail oder meinem Google-Account anmelden können, **damit** ich ein Nutzerprofil erstellen kann.
- Als **Benutzer möchte ich** ein Nutzerprofil erstellen können, **damit** meine persönlichen Daten und Präferenzen gespeichert werden.

- Als **Benutzer möchte ich**, dass meine Nutzerdaten persistent gespeichert werden, **damit** ich bei zukünftigen Anmeldungen darauf zugreifen kann.
- Als **Benutzer möchte ich** meine Nutzerdaten ändern können, **damit** ich meine Informationen aktuell halten kann.

Kauffunktion

- Als **Manga-Fan möchte ich** einen (simulierten) Kaufprozess durchlaufen können, **damit** Mangas kaufen kann.
- Als **Manga-Fan möchte ich** eine Warenkorb-Funktion nutzen können, **damit** ich mehrere Mangas gleichzeitig kaufen kann.
- Als **Manga-Fan möchte ich** eine Rechnung für meine Käufe erhalten, **damit** ich eine Übersicht über meine Ausgaben habe.
- Als **Manga-Fan möchte ich**, dass die Rechnungen in meinem Nutzerprofil gespeichert werden, **damit** ich sie jederzeit einsehen kann.

Use-Case Diagram

Die nachfolgende Grafik zeigt die Funktionalitäten der Applikation in einem Use-Case Diagramm:

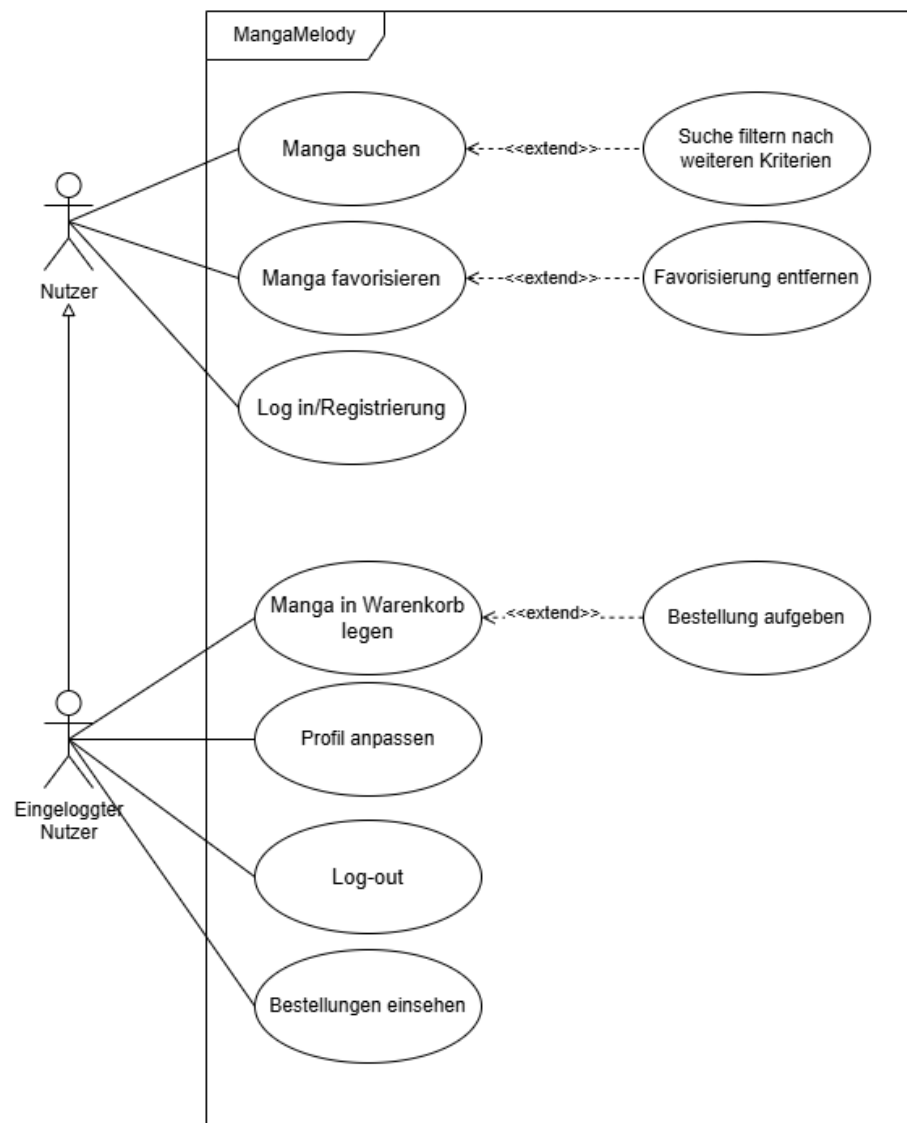


Figure 4: Use-Case Diagram der Funktionalitäten.

Bei diesem wird ein Nutzer wird unterteilt in allgemeine Nutzer, und Nutzer, die eingeloggt sind

Nichtfunktionale Anforderungen

Benutzerfreundlichkeit und Design

- Die Anwendung sollte eine intuitive Benutzeroberfläche haben, die es den Benutzern ermöglicht, die Funktionen einfach zu finden und zu nutzen.
- Die Anwendung wird sich hierbei nach dem Material Design von Google richten, um eine konsistente und moderne Benutzererfahrung zu bieten.

Performance und Robustheit

- Die Anwendung sollte schnell und zuverlässig sein, um eine reibungslose Benutzererfahrung zu gewährleisten.
- Die Ladezeiten sollten bei stabiler Verbindung unter 3 Sekunden sein, um die Benutzer nicht zu verärgern.
- Im Falle von einer Unterbrechung der Verbindung sollte die Anwendung eine Fehlermeldung anzeigen und den Benutzer über die Situation informieren. Zusätzlich sollte die Anwendung dem Benutzer eine eingeschränkte Anzahl an Manga-Titeln anzeigen, die lokal gespeichert sind.

Responsiveness

- Die Anwendung sollte auch auf mobilen Geräten mit kleineren Bildschirmgrößen gut funktionieren und eine reibungslose Benutzererfahrung bieten.
- Die hierbei minimal unterstützte Bildschirmbreite sollte 360 Pixel betragen.
- Alle relevanten Informationen und Features sollten auch auf kleineren Bildschirmen gut lesbar und bedienbar sein.

Sicherheit

- Die Anwendung sollte die Nutzerdaten sicher speichern und übertragen, um die Privatsphäre der Benutzer zu schützen.
- Nutzer sollten nur in der Lage sein, die eigenen Profilinformationen und Bestellungen einzusehen.

Glossar

Begriff	Beschreibung
API	Application Programming Interface, eine Schnittstelle zur Kommunikation zwischen Softwareanwendungen.
Jikan-API	Eine RESTful API, die Daten von MyAnimeList bereitstellt, um Informationen über Manga und Anime abzurufen.
Manga	Japanische Comics oder Graphic Novels, die in einem spezifischen Stil gezeichnet sind.
REST	Representational State Transfer, ein Architekturstil für verteilte Systeme, insbesondere für Webservices.

Begriff	Beschreibung
UI	User Interface, die Schnittstelle, über die Benutzer mit einer Anwendung interagieren.
UX	User Experience, das Gesamterlebnis eines Benutzers bei der Interaktion mit einer Anwendung.

Spezifikationsdokument

Datenmodell

Bei der Modellierung der Daten konzentrieren wir uns zunächst auf die Hauptentitäten. Hierzu gehören

- **Kunde:** Der Kunde ist der Hauptakteur der Anwendung. Er kann sich einloggen, Bestellungen tätigen, Favoriten speichern und sein Profil verwalten.
- **Manga:** Die Mangas sind die Hauptprodukte der Anwendung. Sie enthalten Informationen wie Titel, Autor, Genre, Preis etc.
- **Bestellung:** Die Bestellungen enthalten Informationen über die gekauften Mangas, den Gesamtpreis, das Bestelldatum und die Lieferadresse.
- **Warenkorb:** Der Warenkorb enthält die ausgewählten Mangas, die der Kunde kaufen möchte.
- **Favoriten:** Die Favoriten sind eine Liste von Mangas, die der Kunde als besonders interessant markiert hat.

Das folgende UML-Klassendiagramm zeigt die Beziehungen zwischen den Entitäten und ihre Attribute.

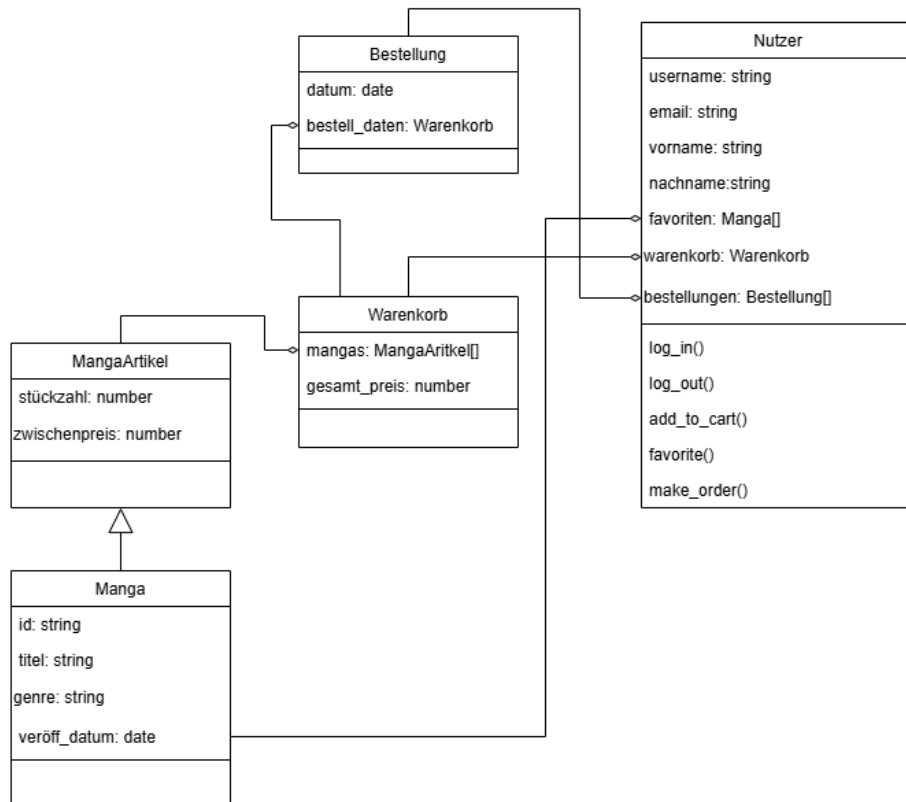


Figure 5: UML-Klassendiagramm der wichtigsten Entitäten.

Zu sehen ist eine Modellierung, die die Hauptentitäten und ihre Beziehungen zueinander darstellt. Einige zusätzliche Entitäten wurden zur Modellierung der Funktionalitäten erstellt. Im Folgenden wird die Modellierung der Hauptentitäten genauer erläutert. Hierbei wird vor allem auf die Beziehungen zwischen den Entitäten eingegangen, die Attribute und Methoden sind weniger relevant.

Ein Nutzer kann Lieblingsmangas haben. Diese werden als Liste gespeichert. Somit besteht eine Aggregation zwischen Nutzer und Mangas. Das Gleiche gilt für Bestellungen. Zudem hat der Nutzer einen Warenkorb.

Eine Bestellung besteht aus einem Datum, und den Bestelldaten, die hier als eine Aggregation zu Warenkorb modelliert wurden.

Ein Warenkorb besteht aus einer Liste der Manga-Items und einem Gesamtpreis. Hierfür wurde ebenfalls eine Aggregation gewählt. In einigen Modellierungen hätte hierfür auch eine Komposition gewählt werden können, was ausdrücken würde, dass der Warenkorb ohne Artikel nicht existieren kann und somit eine Existenzabhängigkeit besteht. In diesem Fall wurde jedoch eine Aggregation

gewählt, da ein leerer Warenkorb als valide angesehen wird. Hierbei wäre die Liste leer und der Gesamtpreis 0.

Ein Manga-Artikel ist eine Generalisierung von Manga. Die **MangaArtikel** Entität ergänzt die Manga-Klasse um die zusätzlichen Informationen der Stückzahl und des Zwischenpreises. Diese Entität wird benötigt, um den Gesamtpreis eines Artikels im Warenkorb zu berechnen.

Geschäftsprozesse

Im Folgenden werden die wesentlichen Geschäftsprozesse der Anwendung erläutert.

Favorisierung

Der Nutzer kann auch, ohne eingeloggt zu sein, bereits Favoriten einspeichern. Ist der Nutzer nicht eingeloggt, werden die Favoriten in dem Speicher des Browsers (**localStorage**) persistiert. Das hat den Vorteil, dass die Favoriten auch nach dem Schließen und Öffnens des Fensters oder nach einem Refresh weiterhin erhalten bleiben. Die Einschränkung hierbei ist jedoch, dass die Daten nicht über mehrere Geräte hinweg gespeichert werden, sondern nur in dem spezifischen Browser.

Ist der User jedoch eingeloggt, werden diese Daten in der Datenbank gespeichert, sodass auf die Favoriten von überall zugegriffen werden können.

Als Pseudocode lässt sich dieser Prozess wie folgt beschreiben:

```
IF User.isLoggedIn
THEN Database.save(User.favorites)
ELSE LocalStorage.save(User.favorites)
```

Login-Prozess

Um alle Funktionen der Applikation zu nutzen, ist eine Anmeldung des Nutzers notwendig. Hierfür soll die Applikation die Möglichkeit bereitstellen, sich entweder klassisch über E-Mail und Passwort anzumelden, oder ein Google-Login zu nutzen.

Bestellungsprozess

Um einen Bestellungsprozess erfolgreich abzuschließen, muss der Nutzer die folgenden Schritte absolvieren. Dieser Prozess wird in dem folgenden UML-Aktivitätsdiagramm visuell dargestellt. Eine Besonderheit hierbei ist, dass der Login Prozess nicht explizit dargestellt wird, da dieser bereits zuvor beschrieben wurde. Innerhalb des Diagramms wird dieser Prozess also als eine Art Blackbox betrachtet.

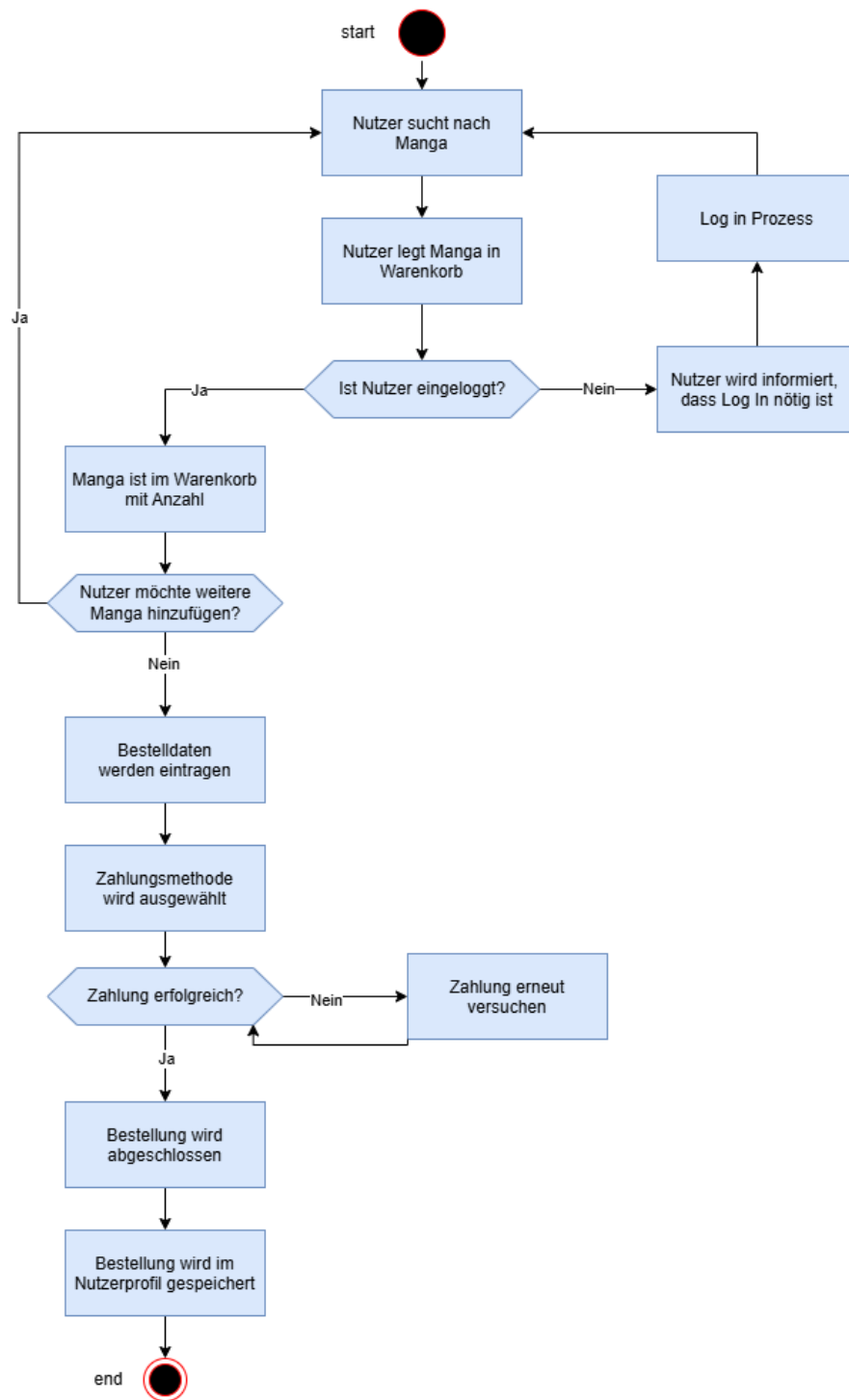


Figure 6: UML-Aktivitätsdiagramm des Bestellungsprozesses.

Systemschnittstellen

Manga-Schnittstelle

Die Hauptschnittstelle wird die Jikan API sein. Mit dieser wird die Applikation direkt über HTTP Requests kommunizieren. Das Datenformat für die Kommunikation ist JSON.

Firebase Datenbank

Für die Persistenz bestimmter Daten wie Nutzernamen, Bestellungen etc. und auch für Funktionalitäten für das Login verwendet die Applikation die Echtzeit-Datenbank Firebase von Google. Das Kommunikationsprotokoll hierfür ist Websocket Communication.

Geschäftsregeln

Nutzer

- Nutzer können nur ihre eigenen Daten und Bestellungen einsehen
- Nur eingeloggte User können Artikel in den Warenkorb legen
- Nur eingeloggte User können Bestellungen tätigen

Bestellungen

- Die maximale Bestellmenge pro Manga ist 50 Stück
- Das Bestelldatum darf nicht in der Zukunft liegen, sondern muss dem aktuellen Tag entsprechen
- Die Anzahl der Waren im Warenkorb darf nicht unter 0 sein
- Jede getätigte Bestellung muss gespeichert werden und für den Nutzer im Nachhinein verfügbar sein

Preise

- Der Preis (sowohl pro Stück als auch gesamt) darf nie negativ sein
- Der angezeigte darf nur maximal zwei Nachkommastellen haben (keine Darstellungsfehler durch Rundungsfehler)

Benutzerschnittstellen

Im Folgenden werden Skizzen der Benutzerschnittstellen gezeigt. Diese sind nur grobe Entwürfe und dienen dazu, die Struktur der GUI zu visualisieren.

Bei dem Betreten der Seite wird der Nutzer auf die Landing-Page geleitet. Als “Hero” Komponente bezeichnet man eine große, auffällige Komponente, die bei dem Nutzer Interesse wecken soll. Von hier aus kann der Nutzer über eine Navigationsleiste auf die verschiedenen Seiten der Applikation navigieren.

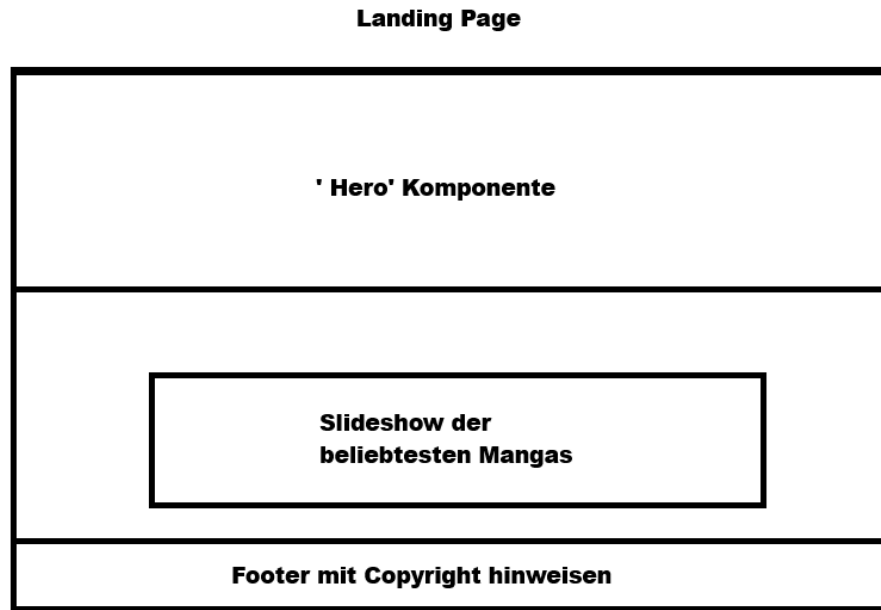


Figure 7: Skizze der GUI der Landing-Page.

Auf der Login/ Sign-up Seite kann der Nutzer sich entweder einloggen oder ein neues Konto erstellen. Hierbei wird unterschieden, ob der Nutzer bereits ein Konto hat oder nicht. Die GUI ist in zwei Teile aufgeteilt, die sich durch Tabs wechseln lassen.

Hierbei soll es eine Eingabvalidierung geben, die sicherstellt, dass die E-Mail-Adresse gültig ist und das Passwort bestimmten Sicherheitsanforderungen entspricht. Zudem muss bei der Registrierung das wiederholte Passwort mit dem ersten Passwort übereinstimmen. Auch die Adressdaten müssen validiert werden.

Log in/Sign up Page im Log in Tab

Log in
Sign up

passwort vergessen?
oder

Login with Google

Log in/ Sign up page im Sign up Tab

Log in
Sign up

oder

Sign up with Google

Figure 8: Skizze der GUI der Login/ Sign-up Ansicht.

Die Profil-Ansicht zeigt dem Nutzer seine persönlichen Daten, wie z.B. Name, E-Mail, Adresse etc.

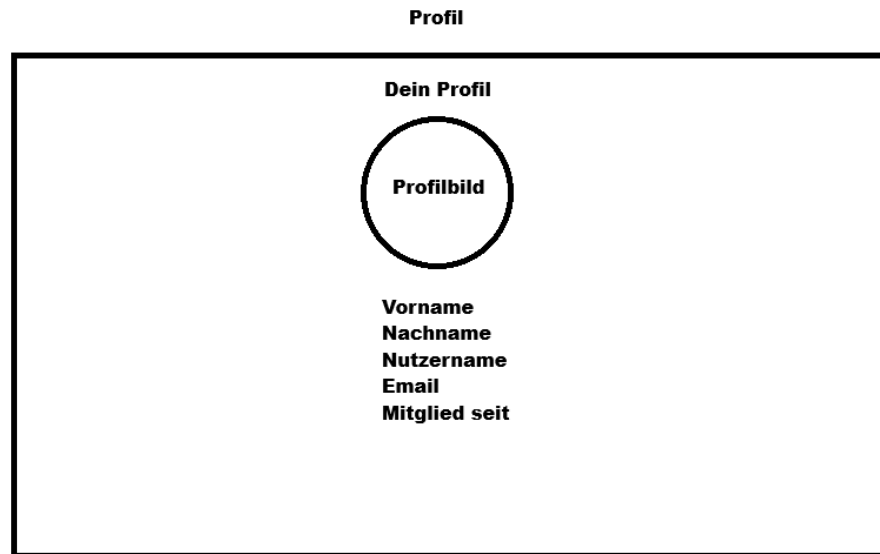


Figure 9: Skizze der GUI der Profil-Ansicht.

Die Manga-Suche ermöglicht es dem Nutzer, nach Mangas zu suchen. Hierbei kann der Nutzer nach verschiedenen Kriterien filtern, wie z.B. Genre, Autor, Preis etc. Die Manga Suche ist das Hauptfeature der Applikation und sollte daher einfach und intuitiv zu bedienen sein. Durch das anklicken eines Mangas wird der Nutzer auf die Detailansicht des Mangas weitergeleitet. Es soll dem Nutzer auch möglich sein, Mangas zu favorisieren und in den Warenkorb zu legen.

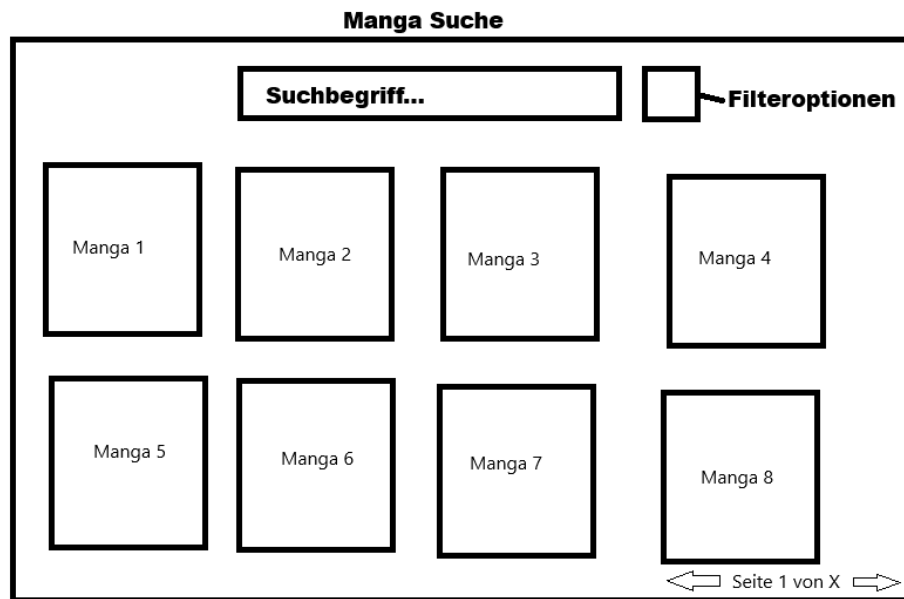


Figure 10: Skizze der GUI der Manga Suche.

Die Favoriten-Ansicht zeigt dem Nutzer eine Liste seiner favorisierten Mangas. Hierbei kann der Nutzer die Mangas auch aus der Liste entfernen.

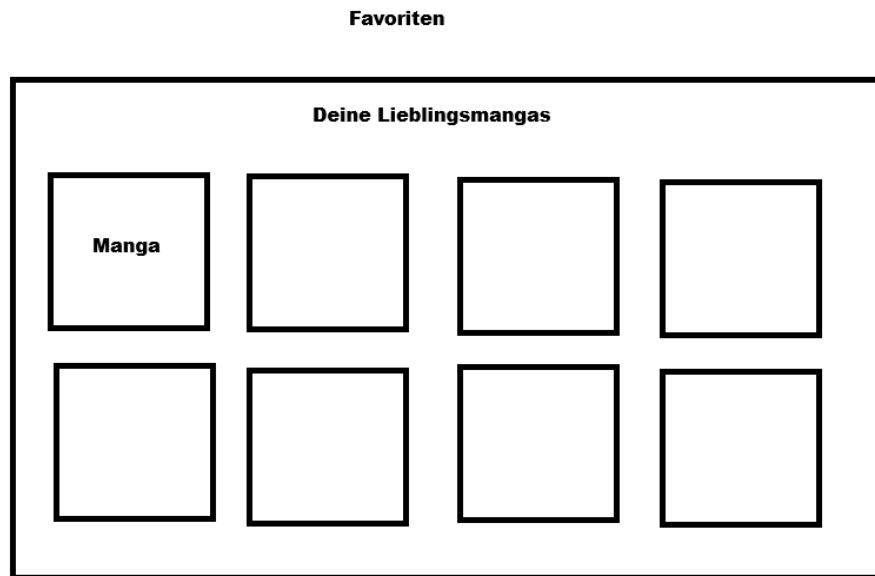




Figure 11: Skizze der GUI der Favoriten.

Die Warenkorb-Ansicht zeigt dem Nutzer eine Liste der Mangas, die er kaufen möchte. Hierbei kann der Nutzer die Anzahl der Mangas ändern oder den Manga aus dem Warenkorb entfernen. Der Gesamtpreis wird automatisch berechnet.

Warenkorb

	Manga 1 Anzahl x Stückpreis y
	Manga 1 Anzahl x Stückpreis y

Gesamtpreis: €100




Figure 12: Skizze der GUI des Warenkorbs.

Die Bestellungen-Ansicht zeigt dem Nutzer eine Übersicht seiner bereits getätigten Bestellungen.

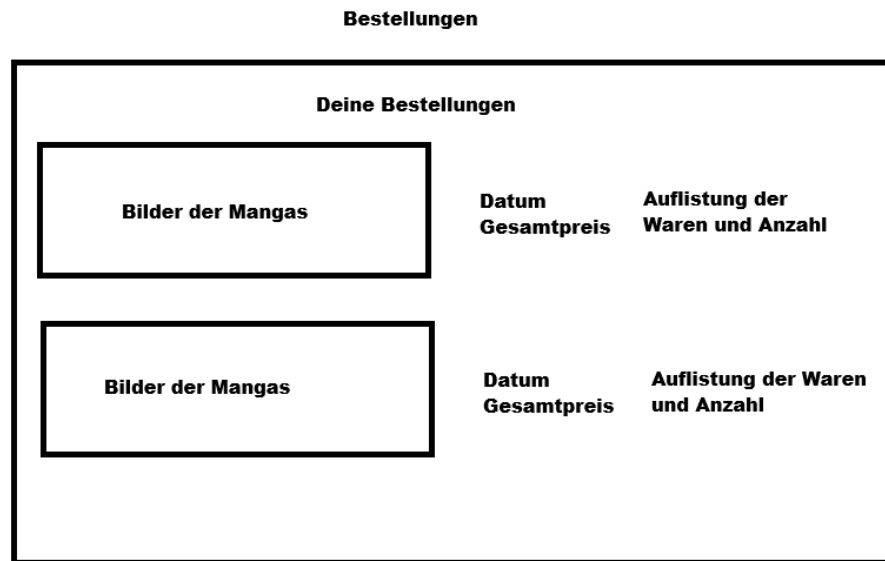


Figure 13: Skizze der GUI der Bestellungen-Anischt.

Architekturdokument

In dieser Projektphase werden Architektur und technische Details festgelegt sowie die Anwendung durch Schreiben des Quellcodes implementiert. Zunächst soll hierzu ein Architekturdokument erstellt werden. Dieses umfasst die folgenden Punkte:

Technologieübersicht



Figure 14: Offizielles Angular Logo.

Angular ist ein Open-Source-Framework zur Erstellung von Webapplikationen. Entwickelt und gewartet wird das Framework von Google und der Open-Source Entwicklercommunity. Angular nutzt, wie viele modernen UI Libraries, ein komponentenbasiertes Architekturmodell. Mit diesem können komplexe Anwendungen nach dem Baustein-Prinzip, durch die Komposition einzelner, modularer Komponenten, aufgebaut werden. Dadurch wird die Wiederverwendbarkeit und Wartbarkeit des Quellcodes erhöht. Auch für die Skalierbarkeit des Projektes ist diese Architektur vom Vorteil, da neue Features leicht hinzugefügt werden können, indem neue Komponenten erstellt, oder bestehende erweitert werden, ohne dass andere Teile des Systems stark angepasst werden müssen.

Eine Angular Komponente besteht typischerweise aus einem Template (.html Datei), einem Stylesheet (.scss Datei), und der Logik, die in einer Typescript (.ts) Datei verwaltet wird. Dadurch gibt es eine saubere Trennung zwischen der Logik und der Darstellung.

Somit bietet Angular eine robuste Grundlage für die Entwicklung skalierbarer Anwendungen, da es eine klare Struktur, Wiederverwendbarkeit von Komponenten und eine effiziente Zusammenarbeit zwischen mehreren Entwicklungsteams durch die Modularität und Isolation der Komponenten ermöglicht.



Figure 15: Offizielles Angular Material Logo.

Angular Material ist eine offizielle UI-Komponentenbibliothek für Angular, die auf den Designprinzipien von Googles Material Design basiert. Sie bietet eine Sammlung vorgefertigter und anpassbarer UI-Komponenten wie Buttons, Formulare, Tabellen, Dialoge, Toolbars und mehr. Angular Material verfolgt das Ziel, eine konsistente, benutzerfreundliche und visuell ansprechende Benutzeroberfläche zu ermöglichen. Für das Projekt verbessert es die Entwicklungsproduktivität, fördert ein einheitliches Design und integriert sich nahtlos in das Angular-Framework ein.

Backend

Firebase ist eine umfassende Backend-Plattform von Google, die Funktionen wie Authentifizierung, Datenbankmanagement (Firestore), Cloud-Funktionen, Hosting und mehr bietet. Firebase beschleunigt die Entwicklung, da es viele der üblichen Backend-Aufgaben übernimmt und skalierbare, serverlose Architekturen unterstützt.

AngularFire ist die offizielle Bibliothek für die Integration von Firebase-Diensten in Angular-Anwendungen. Sie bietet eine einfache Möglichkeit, Firebase-Funktionen wie Authentifizierung, Datenbanksynchronisierung und Dateispeicherung in Angular-Projekte einzubinden.

Programmiersprachen

TypeScript ist ein Supersatz von JavaScript, welcher statische Typisierung hinzufügt. Es ermöglicht eine bessere Fehlerprüfung während der Entwicklungszeit und erleichtert das Schreiben und Verstehen von komplexem Code. Angular selbst ist in TypeScript geschrieben und bietet umfangreiche Unterstützung für diese Sprache.

SCSS (Sassy CSS) ist eine präprozessorische Skriptsprache, die zu CSS (Cascading Style Sheets) kompiliert wird. SCSS erweitert die Möglichkeiten von CSS mit Funktionen wie Variablen, verschachtelten Regeln und Mixins, was die Verwaltung und Wiederverwendung von Stilvorlagen deutlich vereinfacht.

Sonstige Tools Architektur-Pattern und Paradigmen

REST-API: Um an die Manga-Daten zu kommen, verwendet das Projekt die Open Source Schnittstelle der Jikan API verwendet, die auf dem REST

Paradigma basiert. Dabei steht REST für

- **RE**presentational
- **S**tate
- **T**ransfer

Es basiert auf einer zustandslosen Kommunikation und verwendet HTTP-Methoden wie **GET**, **POST**, **PUT** und **DELETE**, um Ressourcen zu manipulieren. RESTful APIs sind darauf ausgelegt, einfach und skalierbar zu sein, indem sie standardisierte URLs und Datenformate wie JSON oder XML verwenden. In diesem Fall wird für die Kommunikation JSON verwendet, mit welchem sich leicht in modernen Programmiersprachen arbeiten lässt.

Reaktive Programmierung

Reaktive Programmierung ist ein Programmierparadigma, das sich auf die asynchrone Verarbeitung und den Datenfluss konzentriert. Dabei werden Daten als Streams betrachtet, die kontinuierlich verarbeitet und transformiert werden können. Neue Werte in der Datenquelle propagieren automatisch durch das System, ohne dass explizite Events oder manuelle Abfragen nötig sind. Dies unterscheidet sich zu anderen Ansätzen, wie der imperativen Programmierung, bei der die einzelnen Schritte nacheinander definiert werden. Dies hat den Nachteil, dass mehr Code nötig ist und dieser schwieriger zu warten ist.

Signals: sind eine von verschiedenen konkreten Kommunikationsmethoden, für die Implementierung einer reaktiven Anwendung in Angular. Sie ermöglichen eine effiziente und leistungsstarke Zustandsverwaltung, indem sie Änderungen an Daten automatisch verfolgen und die betroffenen Komponenten bei Bedarf neu rendern.

Das folgende Listing zeigt ein Beispiel für eine einfache Komponente, die Signals nutzt.

```

Einfaches Beispiel für Signals

import { Component, signal } from '@angular/core'

@Component({
  selector: 'app-counter',
  template: `
    <h1>Counter: {{ count() }}</h1>
    <button (click)="increment()">+</button>
    <button (click)="decrement()">-</button>
  `,
})
export class CounterComponent {
  count = signal(0) // Signal mit initialem Wert 0

  increment() {
    this.count.set(this.count() + 1) // Wert setzen
  }

  decrement() {
    this.count.set(this.count() - 1)
  }
}

```

Figure 16: Code-Beispiel für Signals in Angular.

Mit dem `signal(0)` Befehl wird ein neues Signal `count` deklariert. Dieses kann dann mit `count.set()` einem neuen Wert zugeschrieben werden. In dem Template kann dieser reaktive Wert dann sehr leicht über den `count()` Syntax genutzt werden. Der Vorteil ist hier, dass das User Interface sehr präzise und performant erkennt, wenn sich der `count` Wert verändert hat, wodurch nicht die gesamte Komponente, sondern nur der relevante Bereich der Seite neu gerendert werden muss.

Durch die Nutzung von Signals in Angular kann man also sehr einfach das reaktive Programmierparadigma umsetzen, und bekommt dazu noch bessere Performanz und Wartbarkeit.

Architekturübersicht

Insgesamt verwendet die Applikation eine **serverlose Architektur**. Das Merkmal dieser Architektur besteht nicht darin, dass es überhaupt keine Server gibt, sondern, dass die Verwaltung, die Bereitstellung der Infrastruktur, und die Skalierung der Server an einen Drittanbieter (z.B. Google, Amazon oder Microsoft) ausgelagert wird. Bei der Nutzung einer solchen Architektur muss sich also nur noch um den eigenen Quellcode gekümmert werden. Dies bietet

den Vorteil, dass enorm an Aufwand, Zeit und Ressourcen gespart werden kann, da das Aufsetzen und Verwalten von Servern sehr aufwendig ist.

In diesem Fall ist Google der Anbieter, durch das Firebase Produkt, welches von der Anwendung als Backend-as-a-service (Cloud Firestore) und zur Authentifizierung (Firebase Auth) verwendet wird. Somit besteht die Applikation aus einem Frontend Client, einem Backend-as-a-service (CI) und der Jikan REST-API, die die Manga-Daten liefert.

Das folgende Diagramm zeigt eine abstrakte Darstellung der beschriebenen Architektur.

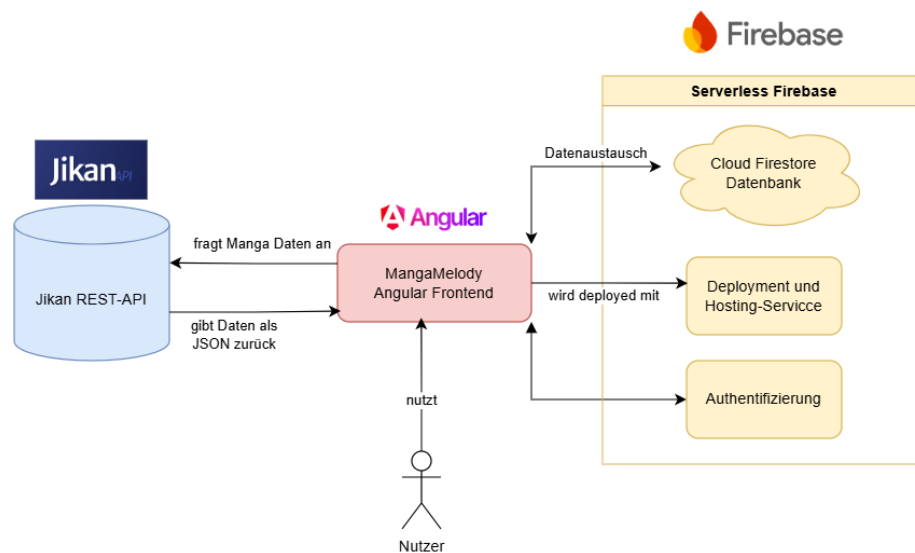


Figure 17: Externes Architekturdiagramm der Anwendung und ihrer Komponenten.

Struktur

Das folgende UML-Klassendiagramm zeigt die Hauptkomponenten der Anwendung sowie die Beziehungen zueinander.

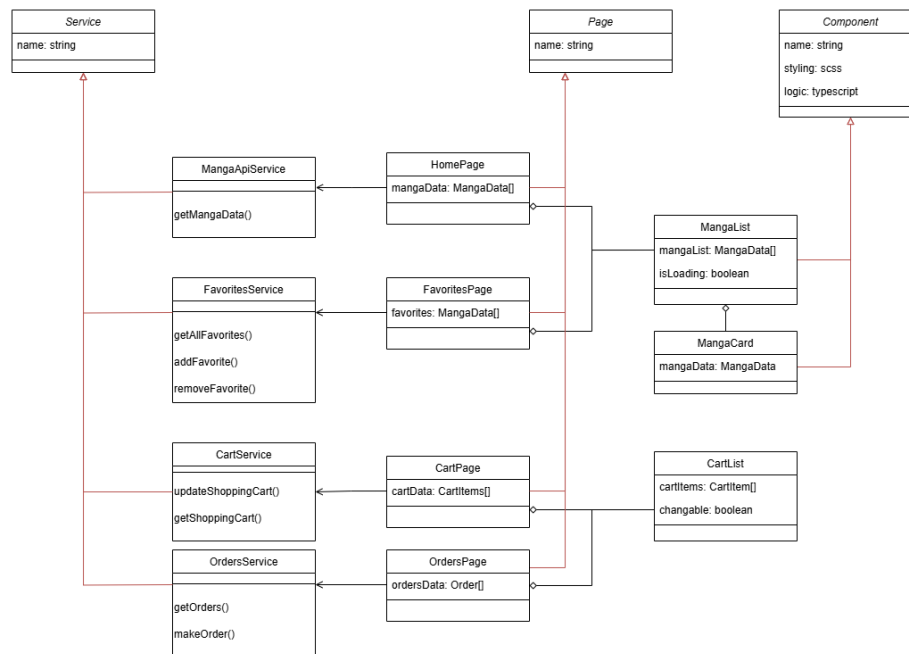


Figure 18: UML-Klassendiagramm der Hauptkomponenten der Anwendung.


Verhalten

Im folgenden wird das Verhalten der Anwendung beschrieben, wenn ein Nutzer auf der Manga-Details Seite, einen Manga in den Warenkorb legt.

MangaMelody

Manga Details

JoJo no Kimyou na Bouken Part 7: Steel Ball Run



Ratings

Score: 9.32 / 10 ★

Popularity: #23 🏆

Members: 285776 👤

Genres

Action Adventure Mystery Supernatural

Synopsis

In the American Old West, the world's greatest race is about to begin. Thousands line up in San Diego to travel over six thousand kilometers for a chance to win the grand prize of fifty million dollars. With the era of the horse reaching its end, contestants are allowed to use any kind of vehicle they wish. Competitors will have to endure grueling conditions, traveling up to a hundred kilometers a day through uncharted wastelands. The Steel Ball Run is truly a one-of-a-kind event. The youthful Johnny Joestar, a crippled former horse racer, has come to San Diego to watch the start of the race. There he encounters Gyro Zeppeli, a racer with two steel balls at his waist instead of a gun. Johnny witnesses Gyro using one of his steel balls to unleash a fantastical power, compelling a man to fire

Price per Manga: 7,00 € EUR

Manga Volume
Volume No: 1

Quantity
2

Add to Cart

Figure 19: Auf dieser Seite kann der Nutzer einen Manga zum Warenkorb hinzufügen.

Add Manga to Cart Process

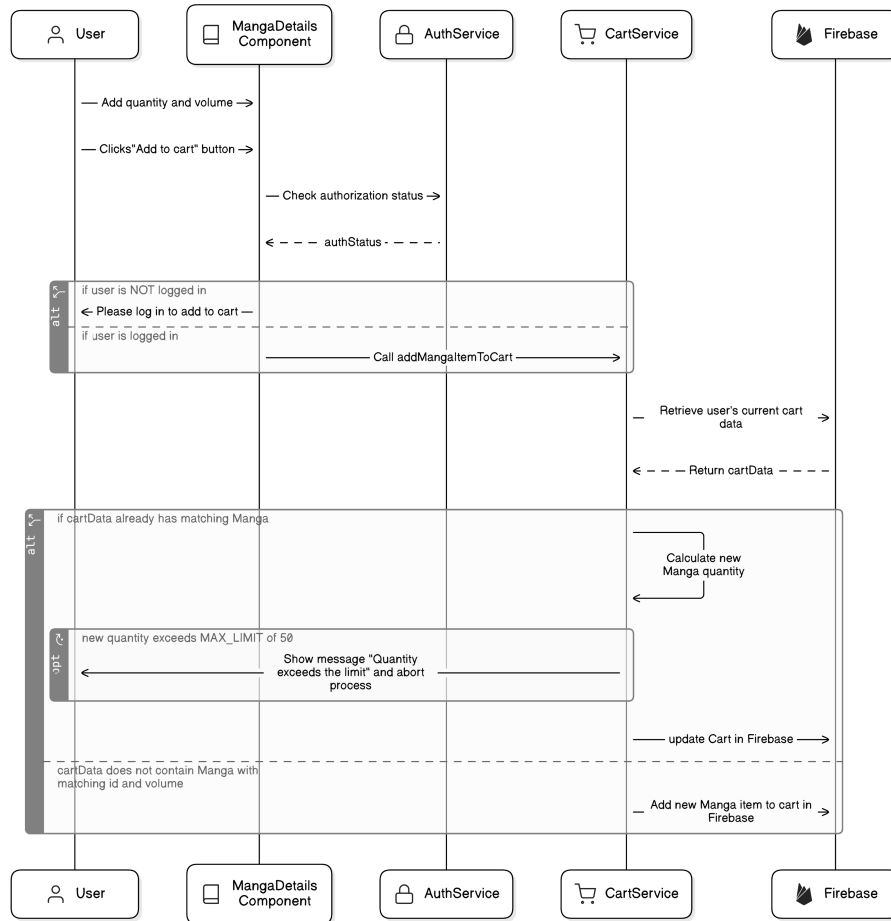


Figure 20: UML-Sequenzdiagramm, welches den Prozess zeigt, mit dem ein Nutzer einen Manga in den Warenkorb legt.