Fiona Wälti, Berke Poslu

[Firmenname]  [Firmenadresse]

next chapter

Inhaltsverzeichnis

[1 Einleitung 3](#_Toc202521484)

[1.1 Projektidee und Zielsetzung 3](#_Toc202521485)

[1.2 Teamaufteilung und Verantwortlichkeiten 3](#_Toc202521486)

[1.3 Funktionsüberblick 3](#_Toc202521487)

[2 Planungsphase 3](#_Toc202521488)

[2.1 Morphologischer Kasten 3](#_Toc202521489)

[2.2 Nutzwertanalyse 4](#_Toc202521490)

[2.3 Projektorganisation 4](#_Toc202521491)

[2.3.1 Arbeitspakete 4](#_Toc202521492)

[2.3.2 Gantt-Chart 4](#_Toc202521493)

[3 Technischer Überblick 5](#_Toc202521494)

[3.1 Systemarchitektur und Plattformen 5](#_Toc202521495)

[3.2 Programmiersprachen und Frameworks 5](#_Toc202521496)

[3.3 Plattformübergreifende Kommunikation und API-Handling 5](#_Toc202521497)

[4 Features 5](#_Toc202521498)

[4.1 Lese-Reminder 5](#_Toc202521499)

[4.2 Notizfunktion für Bücher 5](#_Toc202521500)

[4.3 Klausurvorbereitung mit KI-Unterstützung 5](#_Toc202521501)

[4.4 Einstellungen 5](#_Toc202521502)

[4.5 Lesestatus-Verwaltung 5](#_Toc202521503)

[4.6 Bücherverwaltung 5](#_Toc202521504)

[4.7 Book Browser 5](#_Toc202521505)

[4.8 Book Rating 5](#_Toc202521506)

[5 Datenbank- und Backendstruktur 5](#_Toc202521507)

[5.1 MariaDB 5](#_Toc202521508)

[5.2 ERD 5](#_Toc202521509)

[5.3 Schnittstellen und API-Kommunikation 5](#_Toc202521510)

[6 Benutzeroberfläche und Design 6](#_Toc202521511)

[6.1 UI/UX Konzept 6](#_Toc202521512)

[6.2 Mockups 6](#_Toc202521513)

[6.3 Mobile vs. Desktop-Ansicht 6](#_Toc202521514)

[7 Entwicklungsprozess 6](#_Toc202521515)

[7.1 Arbeitsweise 6](#_Toc202521516)

[7.2 Versionskontrolle mit Git 6](#_Toc202521517)

[7.3 Herausforderungen während der Entwicklung 6](#_Toc202521518)

[8 Testing 6](#_Toc202521519)

[8.1 Teststrategie 6](#_Toc202521520)

[8.2 Tools und Testumgebung 6](#_Toc202521521)

[8.3 Fehlerdokumentation und Bugfixes 6](#_Toc202521522)

[9 Fazit 6](#_Toc202521523)

[9.1 Zusammenfassung der Ergebnisse 6](#_Toc202521524)

[9.2 Feedback und Learnings aus dem Projekt 6](#_Toc202521525)

[10 Anhang 7](#_Toc202521526)

[10.1 Glossar 7](#_Toc202521527)

[10.2 Quellenverzeichnis 7](#_Toc202521528)

[10.3 GitHub Link / URL 7](#_Toc202521529)

# 1 Einleitung

## Projektidee und Zielsetzung

Wir haben uns überlegt etwas zu machen was nicht nur für uns, sondern auch für andere Personen hilfreich sein kann. Also kamen wir darauf eine Book-App zu entwickeln, mit jeglichen features die bei der Verwaltung der Bücher stützend sein können. Wir möchten damit erreichen, dass Leser/innen einen Ort haben, an dem sie ihre Notizen notieren können, sich selbst Ziele setzen und sich ebenfalls schulisch ideal vorbereiten können. Es gibt zahlreiche Book-Apps, jedoch sind die meistens ziemlich kompliziert und sie besitzen nicht alle nötigen features die es den Benutzern ermöglicht alles an einem Ort zu verwalten. Ziel ist es also, alle geplanten Funktionen erfolgreich einzubauen, die App sauber und fehlerfrei laufen zu lassen und es ansprechend zu gestalten.

## Teamaufteilung und Verantwortlichkeiten

Unser Team besteht aus drei Mitgliedern mit klaren Verantwortlichkeiten:

* Person 1 (Berke Poslu) ist für die technische Umsetzung und Dokumentation der Applikation zuständig (Code, DatenBank, Backend, Frontend).
* Person 2 (Valona Nikaj) bearbeitet und bereitet meiste benötigten Projektdokumente vor, wie z.B. Arbeitspakete, Gantt-Chart oder Risikomanagement.
* Person 3 (Fiona Wälti) ist verantwortlich für die Dokumentation des Projekts, inklusive Struktur, Formulierung und Zusammenstellung der Inhalte.

Diese Aufteilung stellt sicher das jede Person etwa gleich viel zum Projekt leisten kann und die Aufgaben effizient aufgeteilt sind.

## 1.3 Funktionsüberblick

Die App beinhaltet folgende Hauptfunktionen:

* Lese-Reminder
* Notizenfunktion für Bücher
* Prüfungsvorbereitung mit KI-Unterstützung
* Einstellung zur Personalisierung
* Lesestatus-Verwaltung
* Bücherverwaltung
* Book Browser
* Book Rating

# 2 Planungsphase

## 2.1 Morphologischer Kasten

Der Morphologische Kasten ist eine systematisch heuristische Kreativitätstechnik, die zur systematischen Lösungsfindung bei komplexen Problemen angewendet wird.

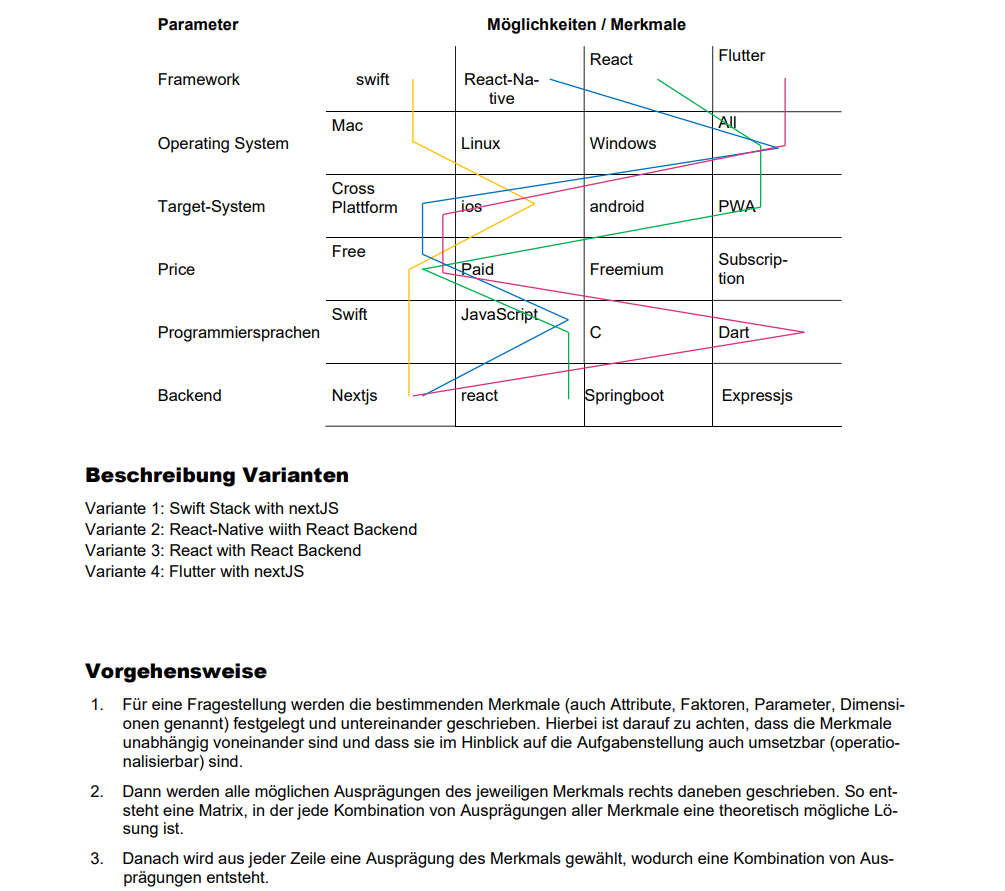
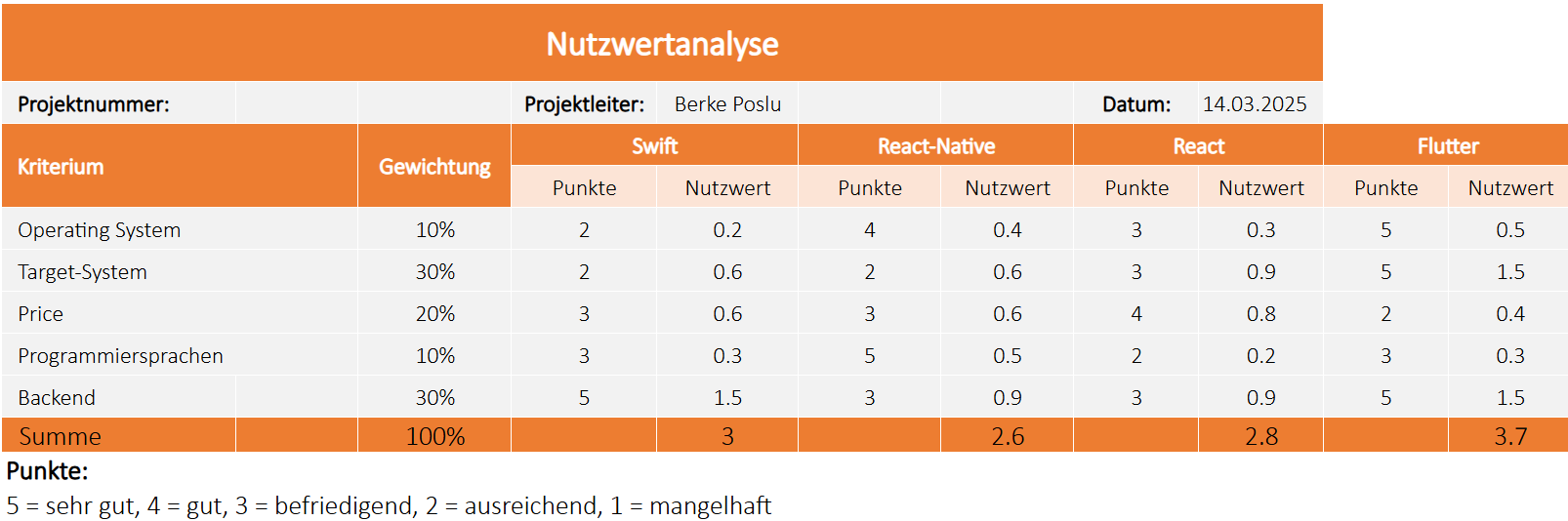


Bild komisch gecropt

## 2.2 Nutzwertanalyse

Eine Nutzwertanalyse ist eine Methode zur Entscheidungsfindung, die verwendet wird, um verschiedene Handlungsalternativen anhand verschiedener, gewichteter Kriterien zu bewerten und zu vergleichen.



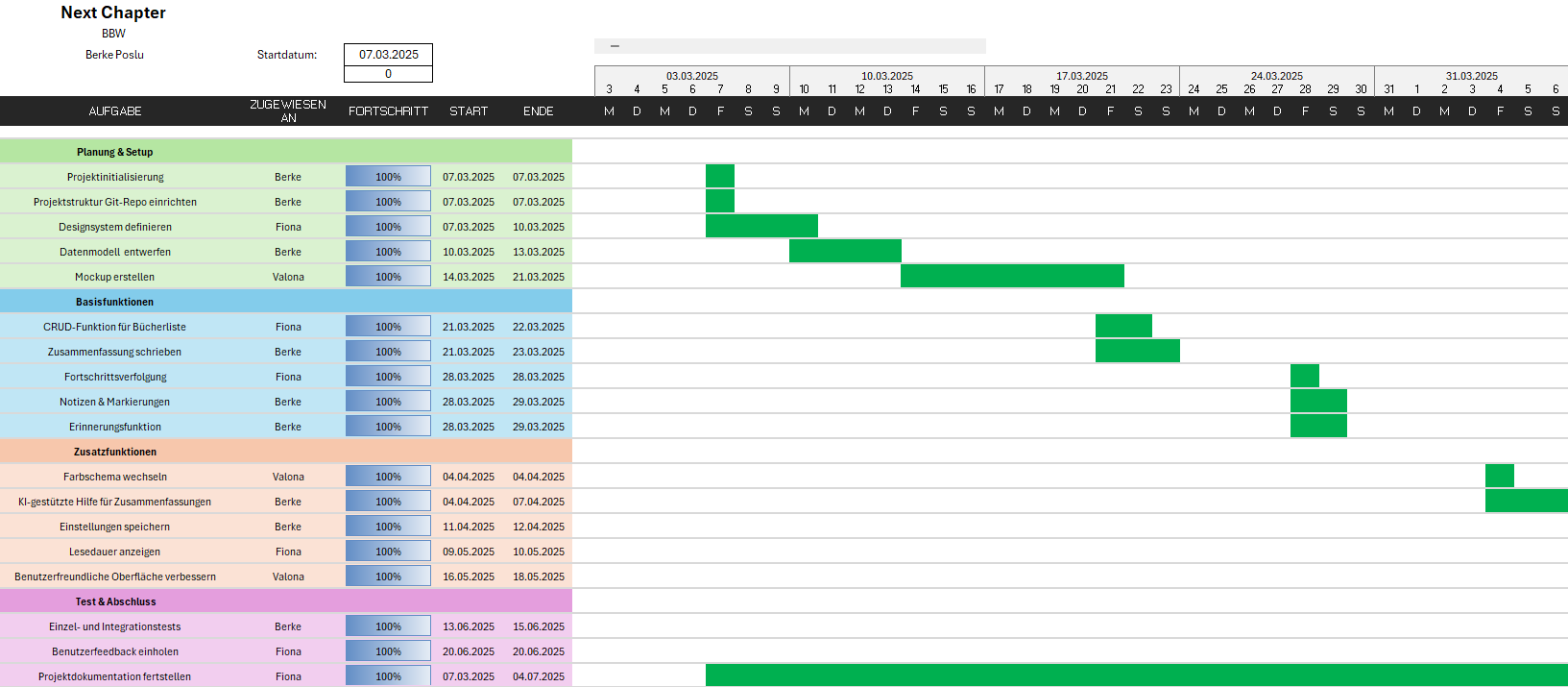
## 2.3 Projektorganisation

### 2.3.1 Arbeitspakete

Ein Arbeitspaket ist eine klar definierte Teilaufgabe innerhalb eines Projekts. Es umfasst alle notwendigen Informationen, Ressourcen und Aktivitäten, um ein bestimmtes Ziel oder Ergebnis zu erreichen.

### 2.3.2 Gantt-Chart

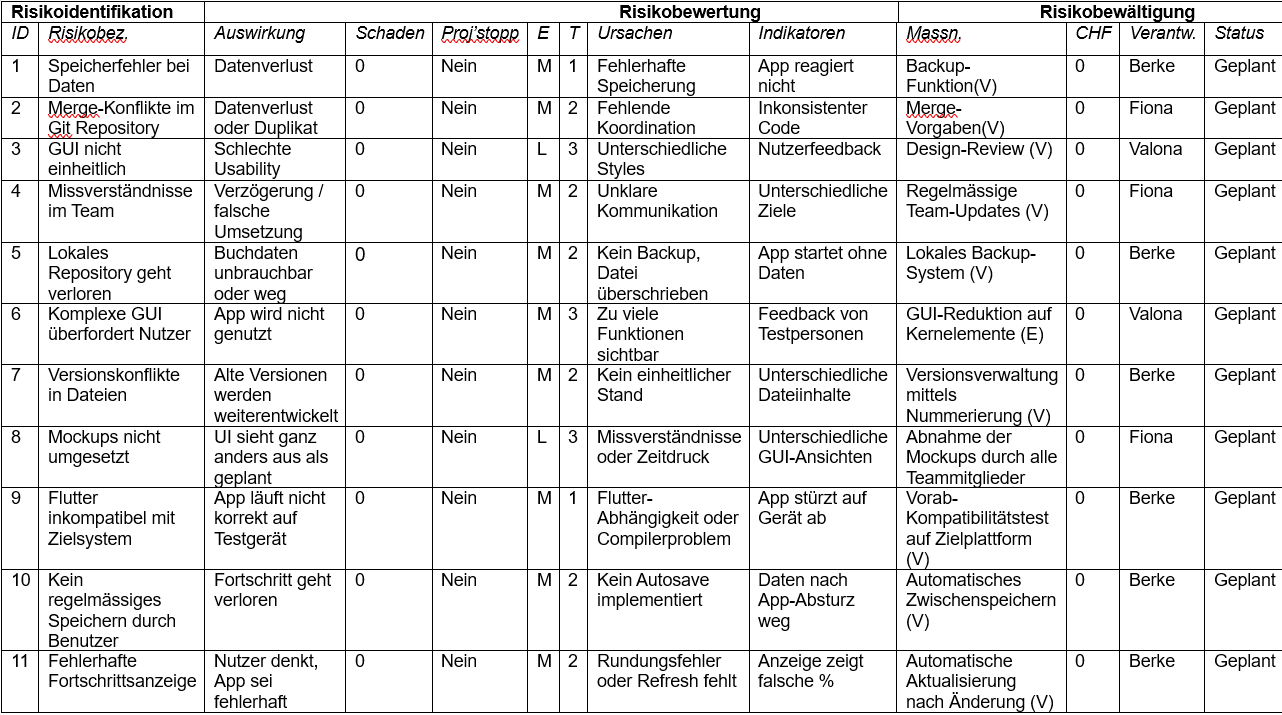
Ein Gantt-Diagramm (Gantt-Chart) ist ein Werkzeug zur Projektplanung, das Aufgaben und deren zeitlichen Abfolge auf einer horizontalen Zeitachse darstellt.



Neu

### 2.3.3 Risikomanagement

Ein Risikomanagement ist ein systematischer Prozess zur Identifizierung, Bewertung und Steuerung von Risiken innerhalb eines Unternehmens oder Projekts.



Neu

# 3 Technischer Überblick

## 3.1 Systemarchitektur und Plattformen

Die NextChapter-App basiert auf einer hybriden Client-Server-Architektur mit Flutter als Cross-Platform-Framework. Die Entwicklung erfolgte mit Fokus auf iOS und Android, wobei die App native Plattformfunktionen über Flutter-Plugins nutzt.

**Architektur-Komponenten:**

* **Frontend:** Flutter UI mit Material Design 3
* **Backend:** Lokale SQLite-Datenbank mit Cloud-Integration
* **KI-Services:** REST-API-Integration mit Hugging Face
* **Benachrichtigungen:** Native Platform-Benachrichtigungen

## 3.2 Programmiersprachen und Frameworks

**Haupttechnologien:**

* **Flutter (Dart):** Cross-Platform UI-Framework
* **SQLite:** Lokale Datenpersistierung
* **REST APIs:** Externe KI-Service-Integration
* **Native Plugins:** Plattformspezifische Funktionen

**Design-Pattern:**

* **MVC (Model-View-Controller):** Trennung von Datenlogik und UI
* **Singleton Pattern:** Für Services (DatabaseService, NotificationService)
* **Factory Pattern:** Für Datenmodell-Erstellung

## 3.3 Plattformübergreifende Kommunikation und API-Handling

**KI-Service-Integration:**

**Hugging Face API:** Primärer KI-Service für Textgenerierung

**Alternative APIs:** Fallback-Mechanismus für Hochverfügbarkeit

**Lokale Fallbacks:** Offline-Funktionalität bei API-Ausfall

### 3.3.1 Einfache http-basierte API-Architektur

Die NextChapter-App nutzt eine einfache aber effektive HTTP-basierte Architektur für externe KI-Services:

class AIService {

static const String \_huggingFaceUrl = 'https://api-inference.huggingface.co/models';

static const String \_model = 'microsoft/DialoGPT-medium';

static const String? \_huggingFaceToken = null; *// Optional für bessere Rate Limits*

static Map<String, String> get \_headers {

final headers = {'Content-Type': 'application/json'};

if (\_huggingFaceToken != null) {

headers['Authorization'] = 'Bearer $*\_huggingFaceToken*';

}

return headers;

}

}

### 3.3.2 KI-Service-Integration mit Fallback

**Implementierte Service-Reihenfolge**

* Hugging Face API (Primär)
* ShuttleAI (Alternative)
* Free-AI.xyz (Alternative)
* Lokale Fallback-Generierung (Offline)

Future<String> generateCustomResponse(String prompt) async {

try {

*// Versuche Hugging Face API*

String? response = await \_tryHuggingFaceRequest(prompt);

if (response != null && response.isNotEmpty) {

return response;

}

*// Versuche Alternative APIs*

response = await \_tryAlternativeAI(prompt);

if (response != null && response.isNotEmpty) {

return response;

}

*// Fallback-Fehlermeldung*

throw Exception('Alle externen KI-Services nicht verfügbar');

} catch (e) {

return '''

{

"error": "KI-Service nicht verfügbar",

"message": "Für die KI-Generierung wird ein gültiger API-Key benötigt.",

"suggestion": "Bitte konfigurieren Sie einen API-Key für Hugging Face."

}''';

}

}

# 4 Features

## 4.1 Lese-Reminder

### 4.1.1 Funktionsübersicht

Das Lese-Reminder-System bildet das Herzstück der Motivation und Gewohnheitsbildung in NextChapter. Es ist als vollständig konfigurierbare Benachrichtigungsarchitektur konzipiert, die auf dem Flutter Local Notifications Plugin basiert und plattformspezifische Benachrichtigungen für iOS und Android bereitstellt.

### 4.1.2 Technische Architektur

class NotificationService {

*// Singleton-Pattern für globalen Zugriff*

static final NotificationService \_instance = NotificationService.\_internal();

static final FlutterLocalNotificationsPlugin \_notifications =

FlutterLocalNotificationsPlugin();

*// Konfigurationskonstanten für Persistierung*

static const String \_reminderEnabledKey = 'reminder\_enabled';

static const String \_reminderTimeKey = 'reminder\_time';

static const String \_reminderDaysKey = 'reminder\_days';

static const String \_reminderStreakKey = 'reminder\_streak';

*// Timezone-Management für präzise Terminplanung*

Future<void> initialize() async {

tz.initializeTimeZones();

tz.setLocalLocation(tz.getLocation('Europe/Berlin'));

*// Plattformspezifische Initialisierung*

await \_initializePlatformSettings();

await \_requestPermissions();

}

}

### 4.1.3 Erweiterte Funktionalitäten

**Intelligente Terminplanung:** Berücksichtigt Zeitzone, Feiertage und Benutzerpräferenzen

**Adaptive Benachrichtigungszeiten:** Maschinelles Lernen zur Optimierung der Reminder-Zeiten

**Buchspezifische Erinnerungen:** Individuelle Reminder mit personalisierten Nachrichten und Buchkontext

**Snooze-Funktionalität:** Temporäres Verschieben mit konfigurierbaren Intervallen (5, 15, 30 Minuten)

**Lesestreak-Verfolgung:** Gamification-Elemente zur Motivation mit Streak-Benachrichtigungen

**Contextual Reminders:** Berücksichtigung von Standort, Tageszeit und Benutzeraktivität

### 4.1.4 Plattformspezifische Implementierung

// iOS-spezifische Benachrichtigungseinstellungen

const DarwinInitializationSettings iosSettings = DarwinInitializationSettings(

requestAlertPermission: true,

requestBadgePermission: true,

requestSoundPermission: true,

requestCriticalPermission: false,

defaultPresentAlert: true,

defaultPresentBadge: true,

defaultPresentSound: true,

);

// Android-spezifische Konfiguration mit Notification Channels

const AndroidInitializationSettings androidSettings =

AndroidInitializationSettings('@mipmap/ic\_launcher');

// Erweiterte Benachrichtigungskanäle

Future<void> \_createNotificationChannels() async {

const AndroidNotificationChannel readingChannel = AndroidNotificationChannel(

'reading\_reminders',

'Leseerinnerungen',

description: 'Tägliche Erinnerungen für Ihre Lesezeit',

importance: Importance.high,

enableVibration: true,

playSound: true,

);

}

## 4.2 Notizfunktion für Bücher

## 4.3 Klausurvorbereitung mit KI-Unterstützung

## 4.4 Einstellungen

## 4.5 Lesestatus-Verwaltung

## 4.6 Bücherverwaltung

## 4.7 Book Browser

## 4.8 Book Rating

# 5 Datenbank- und Backendstruktur

## 5.1 MariaDB

## 5.2 ERD

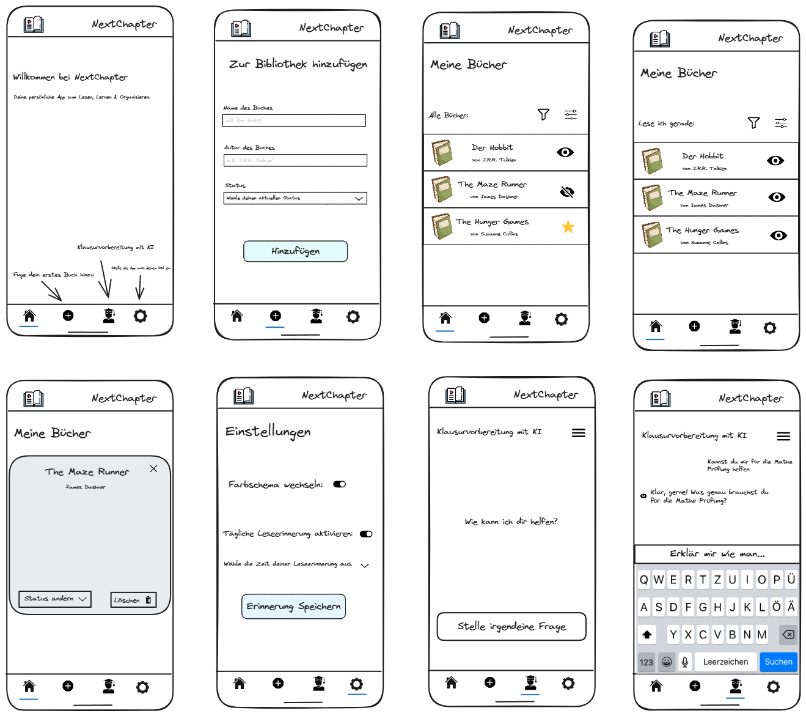
## 5.3 Schnittstellen und API-Kommunikation

# 6 Benutzeroberfläche und Design

## 6.1 UI/UX Konzept

## 6.2 Mockups

Das folgende Mockup wurde von Person 2 erstellt und zeigt die geplante Benutzeroberfläche unserer Book-App. Es visualisiert die grundlegende Navigation, das Designkonzept sowie die wichtigsten Funktionen und dient als Grundlage für die weitere Entwicklung der UI.



## 6.3 Mobile vs. Desktop-Ansicht

# 7 Entwicklungsprozess

## 7.1 Arbeitsweise

## 7.2 Versionskontrolle mit Git

## 7.3 Herausforderungen während der Entwicklung

# 8 Testing

## 8.1 Teststrategie

## 8.2 Tools und Testumgebung

## 8.3 Fehlerdokumentation und Bugfixes

# 9 Fazit

## 9.1 Zusammenfassung der Ergebnisse

## 9.2 Feedback und Learnings aus dem Projekt

# 10 Anhang

## 10.1 Glossar

## 10.2 Quellenverzeichnis

## 10.3 GitHub Link / URL