Arc42

Inhaltsverzeichnis

1. Einführung und Ziele

2. Randbedingungen

3. Kontextabgrenzung

4. Lösungsstrategie

5. Bausteinsicht

6. Laufzeitsicht

7. Verteilungssicht

8. Querschnittliche Konzepte

9. Architekturentscheidungen

10. Qualitätsanforderungen

11. Risiken und technische Schulden

12. Glossar

Kapitel 1 - Einführung und Ziele

1.1 Motivation

In einer Welt, die von Hektik und Zeitmangel geprägt ist, bleibt das Kochen oft auf der Strecke, insbesondere für diejenigen, die sich in der Küche noch nicht sicher fühlen. Unser Projekt „Rezepte für Dummies“ zielt darauf ab, das Kochen zu einem erlebnisreichen Lernprozess zu machen, der sowohl Anfängern den Einstieg ins Kochen erleichtert und auch Spaß dabei macht.

1.2 Qualitätsziele

- Die App sollte innerhalb von 1,5 Sekunden starten.
- Die Ladezeiten bei Benutzereingaben sollten innerhalb von 0,2 Sekunden geschehen.
- Die App soll eine 99 % zuverlässige und konsistente Datenverarbeitung vorweisen.
- Die App sollte höchstens einmal mal in 1.000.000 Fällen abstürzen

1.3 Stakeholder

Stakeholder	Entwicklungsteam
Herr Harald Ichters, Anspruch ist, dass allen Aufgaben unter Beachtung des Aspektes des Software Engineerings zielgerichtet und strukturiert nachgegangen wird und die Abgabetermine eingehalten werden.	Die App soll qualitativ hochwertig sein und für den Endnutzer ansprechend. Deshalb wird ein hoher Standard bei der Entwicklung und Programmierung vorausgesetzt.

1.4 Defintionen
1.5 Referenzen

Kapitel 2 – Randbedingungen

Dieses Projekt wird nach den Verfahren und Vorgaben, die in der Vorlesung Software Engineering an der DHBW Karlsruhe besprochen werden, entwickelt. Es ist stets darauf zu achten, dass diese Vorhaben und Verfahren konsequent eingehalten werden.

Kapitel 3 – Kontextabgrenzung

3.1 Formaler Kontext

Die Android-App kann direkt von Endkunden genutzt werden um Rezepte einzusehen und nach zu kochen.

3.2 Technischer Kontext

Es ergeben sich folgende Abhängigkeiten zu externen Systemen:

- Firebase - Firebase ist eine Entwicklungs-Plattform für Apps und Webanwendungen und hostet unsere Echtzeitdatenbank für die Rezepte
- Picasso - Picasso ist eine Image Loading Library die die Bilder für die Rezepte direkt aus dem Internet laden kann
- Material Design - Material Design ist ein entwickeltes Designsystem um konsistente und ansprechende Benutzeroberflächen zu schaffen

Kapitel 4 – Lösungsstrategie

Im Nachfolgenden sollen alle Lösungsstrategien erläutert werden.

4.1 Technologieentscheidungen

Die Android App wird als Native Android App mit Android Studio entwickelt. Außerdem haben wir uns entschieden Kotlin als Programmiersprache zu verwenden, da diese von Google empfohlen wird. Als Quellcodeverwaltungsprogramm nutzen wir GitHub, da GitHub viele Features anbietet und dazu noch Kostenlos ist. Als Code Editor verwenden wir Android Studio Giraffe, die von Google empfohlene IDE für Android development.

4.2 Architekturmuster

Dieses Projekt verwendet das MVC-Entwurfsmuster für die Android App, entwickelt mit Kotlin. Die klare Trennung von Geschäftslogik und Darstellung erleichtert die Entwicklung und Wartung des Codes.

4.3 Erreichen wichtiger Qualitätsmerkmale

Wir haben die Einhaltung von Programmierstandards und Best Practices versucht, um einen konsistenten und gut lesbaren Code zu gewährleisten. Darüber hinaus haben wir Code-Analysewerkzeugen zur Identifizierung von potenziellen Fehlern und Verbesserung der Codequalität genutzt.

4.4 Organisatorische Entscheidungen

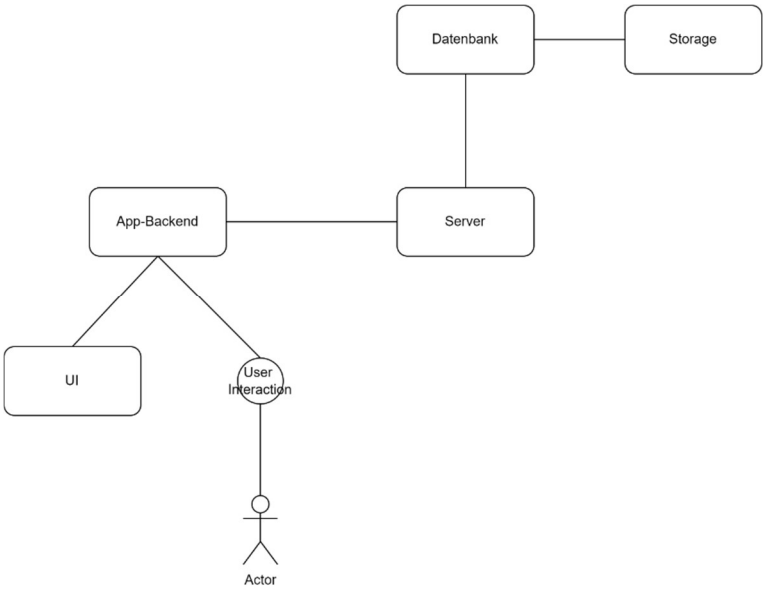
Die Dokumentation des Projektmanagements findet über unser GitHub Discussion Board statt. Unser Projekt haben wir über Jira gemanaged.

Kapitel 5 – Bausteinsicht

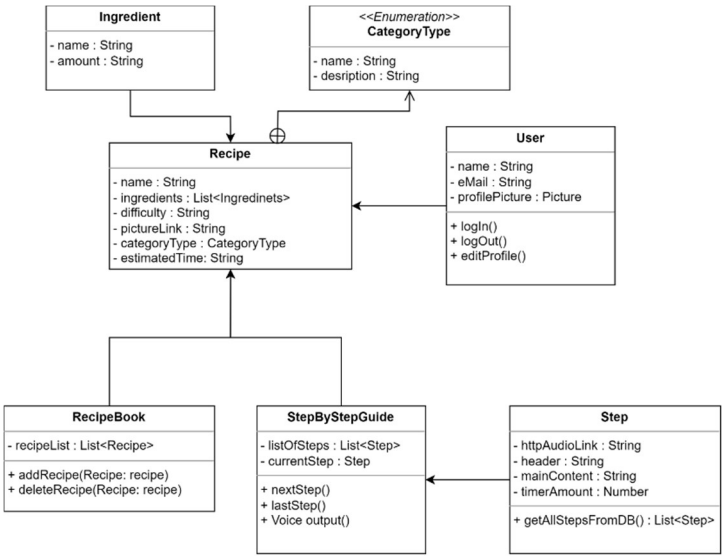
5.1 Kontextabgrenzung

Baustein	
Android App	Die Android App ist für die Darstellung und Interaktion mit dem Benutzer verantwortlich
Firebase Datenbank	Die Datenbank speichert die Rezepte und Bilder
Server	Der Server kommuniziert mit der Android App und der Datenbank, um die angefragten Daten in der App darzustellen

5.2 Ebene 1 – Komponentendiagramm

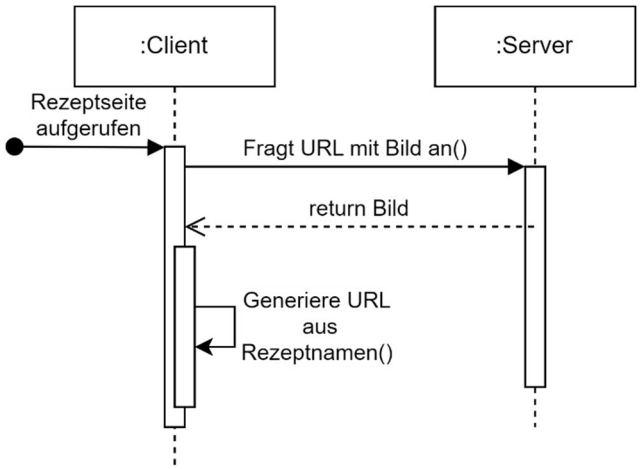


5.3 Ebene 2 - Klassendiagramm

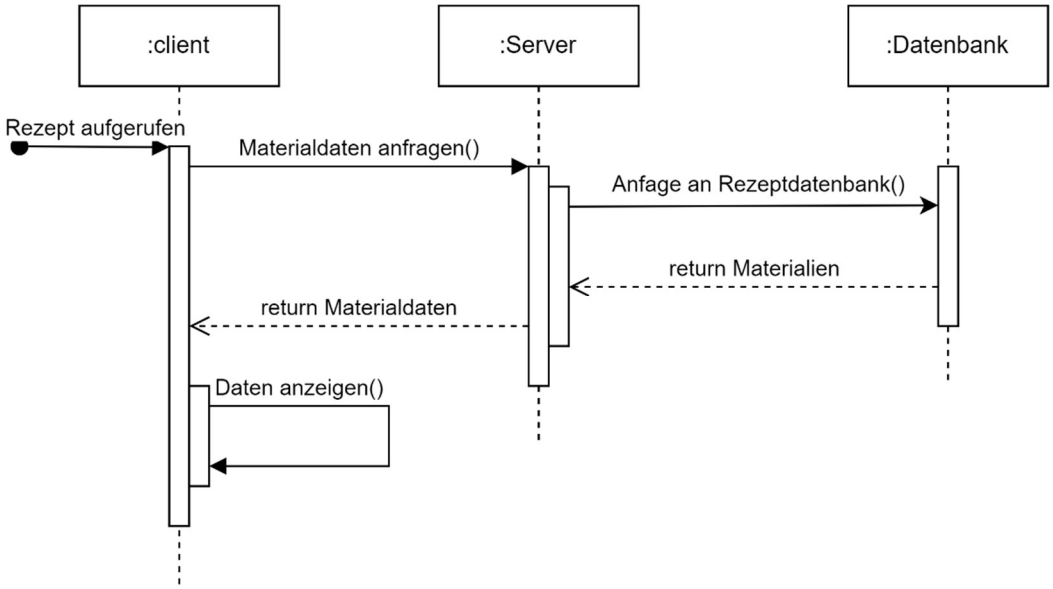


Kapitel 6 – Laufzeiteinsicht

Sobald eine Rezeptseite aufgerufen wird schickt der Client eine Bildabfrage an den Server. Der Server gibt dann ein Bild zurück. Zum Schluss generiert der Client eine URL aus dem Rezeptnamen. Sequenzdiagramm um ein Bild für ein Rezept anzuzeigen.



Sobald ein Rezept aufgerufen wird, leitet der Client die Anfrage an den Server weiter, welcher von der Datenbank die Materialien und Bilder holt. Daraufhin schickt der Server diese zurück an den Client welche dieser dann darstellen kann.



Kapitel 7 - Verteilungssicht

Der Quellcode der Android App kann über GitHub geklont werden. Der Benutzer kann diese App daraufhin selber bauen oder alternativ kann er diese direkt aus dem Google Play Store herunterladen.

Kapitel 8 - Querschnittliche Konzepte

Wichtige Architekturentscheidungen werden innerhalb dieses Dokumentes jeweils nach Erwähnung begründet.

Kapitel 9 - Architekturentscheidungen

Zusammenfassung von Architekturentscheidungen und Designmustern:

1. Strategien:
    - Verwendung von asynchronem Laden für verbesserte Leistung.
    - Einsatz von Caching-Mechanismen für beschleunigte Datenabrufe.
2. Architekturentscheidungen:
    - Integration von Firebase für Echtzeit-Datenaktualisierungen und Skalierbarkeit.

Kapitel 10 - Qualitätsanforderungen

Qualitätsbaum**:**

Qualitätsattribute	Verfeinerung	Szenarien für Qualitätsattribute (Geschäftswert, technisches Risiko)
Leistung	• Starten der App • Ladezeiten bei Benutzereingaben	• Schnelle Ladezeit der App beim Start(M,M) • Minimale Verzögerung bei der Reaktion auf(H,M) Benutzerinteraktionen
Benutzerfreundlichkeit	• Klare Navigation zwischen den Seiten • Intuitives Design der App	• Intuitive Navigation zwischen App-Bildschirmen(H,H) • Klare und leicht verständliche Benutzeroberfläche(H,M)
Skalierbarkeit	• Effizienz • Größere Datenmengen	• Effiziente Handhabung von steigenden Benutzerzahlen(M,M) • Skalierbare Datenbankarchitektur für wachsende Datenmengen(M,M)
Zuverlässigkeit	• Keine Datenverluste • App läuft konsistent	• Konsistente Datenverarbeitung ohne Ausfälle(M,H) • Wiederherstellung nach Netzwerkunterbrechungen oder App-Abstürzen(M,M)
Wartbarkeit	• Neue Features hinzufügen	• Neue Features können ohne Probleme implementiert werden(M,M)

Kapitel 11 - Risiken und technische Schulden

Risiko	Gegenmaßnahme
Firebase fällt aus	Kundensupport anrufen
Langsame Reaktionszeiten und hoher Ressourcenverbrauch	Optimierung von Code, Bilder und Ressourcen sowie sorgfältiges Ressourcenmanagement

Kapitel 12 – Glossar

Akronym	Bedeutung
API	**A**pplication **P**rogrammer **I**nterface (Schnittstelle)