ISEF Projekt

Online Quizsystem – Meilenstein 4

Systemdokumentation

Zweck:

Die Systemdokumentation richtet sich an Entwickler, IT-Architekten und technisches Personal.
Sie dient der Dokumentation des Designs, der Architektur und der technischen Spezifikationen des Systems.

Zielgruppe:

Softwareentwickler, IT-Architekten, technische Projektleiter, technisches Support-Personal

Inhaltsverzeichnis

I. Abbildungsverzeichnis 3
II. Tabellenverzeichnis 3
1. Einleitung und Ziele des Projektes 3
2. Systemarchitektur 4
2.1 Architekturdiagramm 4
2.2 Beschreibung der Komponenten 4
3. Technologie 5
3.1 Programmiersprachen und Frameworks 5
3.2 Datenbank 6
3.3 Weitere Tools 6
4. Datenbankdesign 7
5. Benutzeroberfläche 7
6. Test 8
6.1 Teststrategie 8
6.2 Äquivalenzklassen 9
6.3 WidgetTest 10
6.4 White-Box Test - Anweisungsüberdeckungstest 11
6.5 Analyse der Testergebnisse 13
7. Sicherheit 13
7.1 Sicherheitsrichtlinien 14
8. Ansprechpartner 14

I. Abbildungsverzeichnis

Abbildung 1 Systemarchitektur 4
Abbildung 2 Datenbankdesign 7
Abbildung 3 Aufbau Benutzeroberfläche 8

II. Tabellenverzeichnis

Tabelle 1 Äquivalenzklassen 9

1. Einleitung und Ziele des Projektes

Unsere Quiz-Webapp ist eine vielseitige und interaktive Plattform, die es ermöglicht Benutzern kollaborativ miteinander zu arbeiten und so für ein Modul zu lernen, sodass diese verschiedene ihr Wissen in verschiedenen Themengebiete erweitern und testen können. Die Web-Anwendung bietet einen Einzelspieler Modus, ein Ranglistensystem und die Möglichkeit an, weitere Fragen in dem Fragenkatalog aufzunehmen. Zu dem Authentifizieren bietet die Anwendung eine Registrierung und ein Anmeldefenster an.

Unser Ziel ist es Studenten der IU im Informatik Fachbereich eine Web-Anwendung zu bieten, in der die Nutzer dazu motiviert werden, ihren Wissenstand regelmäßig und zusammen mit ihren Kommilitonen auszuweiten und zu verbessern. Damit die Studenten optimal vorbereitet sind, um eine Top-Note in der nächsten Prüfung zu erhalten.

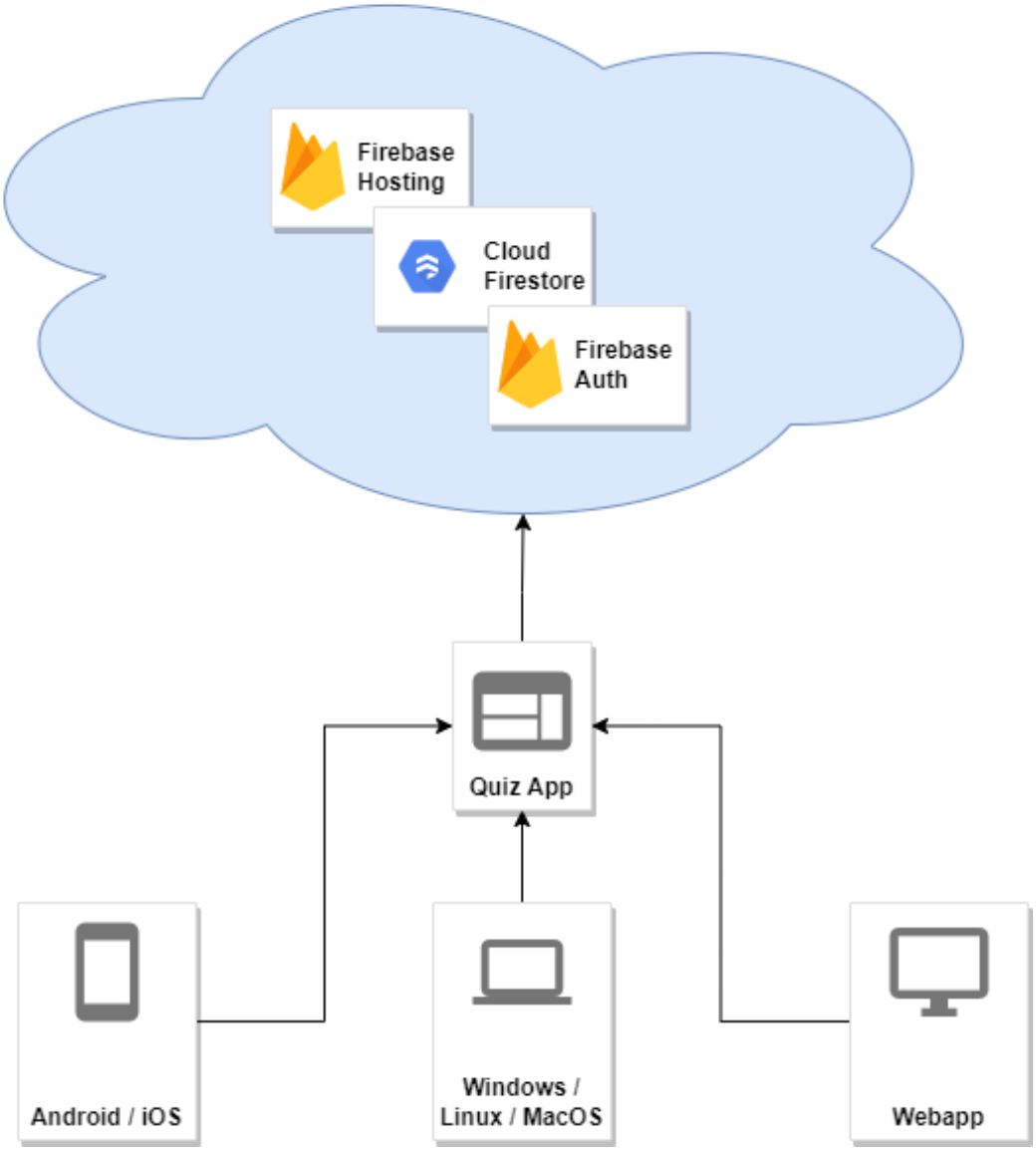
Die Doku zielt darauf ab, die Nutzer über die Systemstruktur zu informieren. Da die Nutzer unserer Applikation größtenteils Studenten sind, fokussiert sich die Dokumentation auf diese Zielgruppe.

Die Systemdokumentation umfasst die systematische Konstruktion unserer Webapp und stellt somit die Architektur dar. Die einzelnen Komponenten, wie auch die Technologien von Frameworks bis Bibliotheken werden beleuchtet. Zudem beinhaltet die Dokumentation das Design der Datenbank und zeigt die API-Endpunkte auf. Außerdem wird eine Beschreibung für die Benutzeroberfläche bzw. Das Frontend unserer Webapplikation geliefert. Abschließend deckt die Systemdokumentation die Testdurchführung und Testergebnisse ab und liefert wertvolle Einblicke über die IT-Sicherheitsrelevanten Maßnahmen.

2. Systemarchitektur

2.1 Architekturdiagramm

Abbildung 1 Systemarchitektur



Quelle: Eigene Darstellung

2.2 Beschreibung der Komponenten

- Clients:
  - Android: Native Apps die auf einem Android Endgerät installiert werden.
  - Webanwendung: Kann über den Browser aufgerufen werden. Der Aufruf kann von jedem Endgerät Internet fähigen Endgerät mit kompatiblem Browser geschehen
- Backend:

    - Firebase Hosting: Bietet die Infrastruktur für die Bereitstellung der Webanwendung und der APIs.
    - Cloud Firestore: NoSQL-Datenbank, in der alle Anwendungsdaten gespeichert werden.
    - Firebase Auth: Authentifizierungssystem für die Benutzeranmeldung und -verwaltung.
- Kommunikation:
    - Alle Clients kommunizieren über APIs mit dem Backend.
    - Die Kommunikation erfolgt über HTTPS und JSON.
- Sicherheit:
    - Alle Kommunikation zwischen Clients und Backend ist über HTTPS verschlüsselt.
    - Firebase Auth bietet robuste Funktionen zur Benutzerauthentifizierung und -autorisierung.
    - Cloud Firestore verfügt über integrierte Zugriffskontrollfunktionen.
- Vorteile der Architektur:
    - Skalierbarkeit, Verfügbarkeit, Wartbarkeit, Flexibilität
- Nachteile der Architektur:
    - Vendor Lock-in, Kosten

3. Technologie

3.1 Programmiersprachen und Frameworks

Im Projekt kommt für die Programmierung der App das Open-Source-Framework Flutter zum Einsatz. Flutter wurde von Google entwickelt, um UI freundliche Cross-Plattform Anwendungen zu ermöglichen. Das Flutter Framework baut auf der Programmiersprache Dart auf. Hierbei handelt es sich um eine objektorientierte, Java nahe Entwicklungssprache. Konstruktiv werden in Flutter Widgets (Ansichten) erstellt welche als UI-Elemente auf der Benutzeroberfläche dargestellt werden. Ziel für unser Projekt war es eine skalierbare und strukturierte Programmiersprache zu verwenden, die sowohl Web-, Mobile- und Desktop-Anwendungen auf möglichst einfache Weise zu ermöglichen. Zwar ist im ersten Rollout nur eine Web-Anwendung vorgesehen, diese lässt sich aber in Zukunft bei Nachfrage erweitern.

Die Kombination von Flutter und Dart ermöglicht eine ideale Voraussetzung für unsere Web-App, da diese eine entwicklerfreundliche Umgebung für die Programmierer bietet, allerdings auch Vorteile wie die plattformübergreifende Entwicklung, eine hohe Leistung durch die Ahead-Of-Time Kompilierung, ein umfangreiches Widget-Framework und aufgrund der Widgets ein einheitliches UI-Design liefert.

Firebase ist eine von Google entwickelte Plattform, die eine breite Palette an Tools und Diensten für die Entwicklung von Webanwendungen und mobilen Anwendungen bietet. Die Plattform bietet ein eigenes Authentication an, welche es Entwicklern ermöglicht, Nutzer zu authentifizieren. Zudem gewährleistet die Firebase Authentication eine einfache Verwaltung von Nutzerkonten. Außerdem unterstützt Firabase den Firestore, welche eine flexible, skalierbare NoSQL-Datenbank ist, die in Echtzeit synchronisiert. Nicht zuletzt bietet Firebase eine Analytics Funktionalität an, die Nutzerdaten und Interaktionen misst und analysiert. Diese umfassenden Funktionalitäten machen Firebase zu einer idealen Umgebung, um unser Webprojekt zu hosten.

3.2 Datenbank

Für unser Web-Projekt verwenden wir die von Google entwickelte Firebase Firestore. Diese ist eine flexible skalierbare NoSQL-Datenbank und ein Teil von der Firebase Plattform. Hauptfunktionalität und Vorteil des Firestore ist eine Echtzeit-Synchronisation der Anwendung. Die Dokumentenbasierte Datenbank speichert die Daten als Dokumente, um die Sammlung an Daten zu organisieren. Diese Struktur ist flexibel und ermöglicht die Speicherung der Daten in einer JSON-ähnlichen Struktur. Aufgrund der einfachen Integration und Verwaltung, eignet sich Firebase besonders gut für eine Quiz-Plattform, bei der sich die Daten problemlos erweitern lassen.

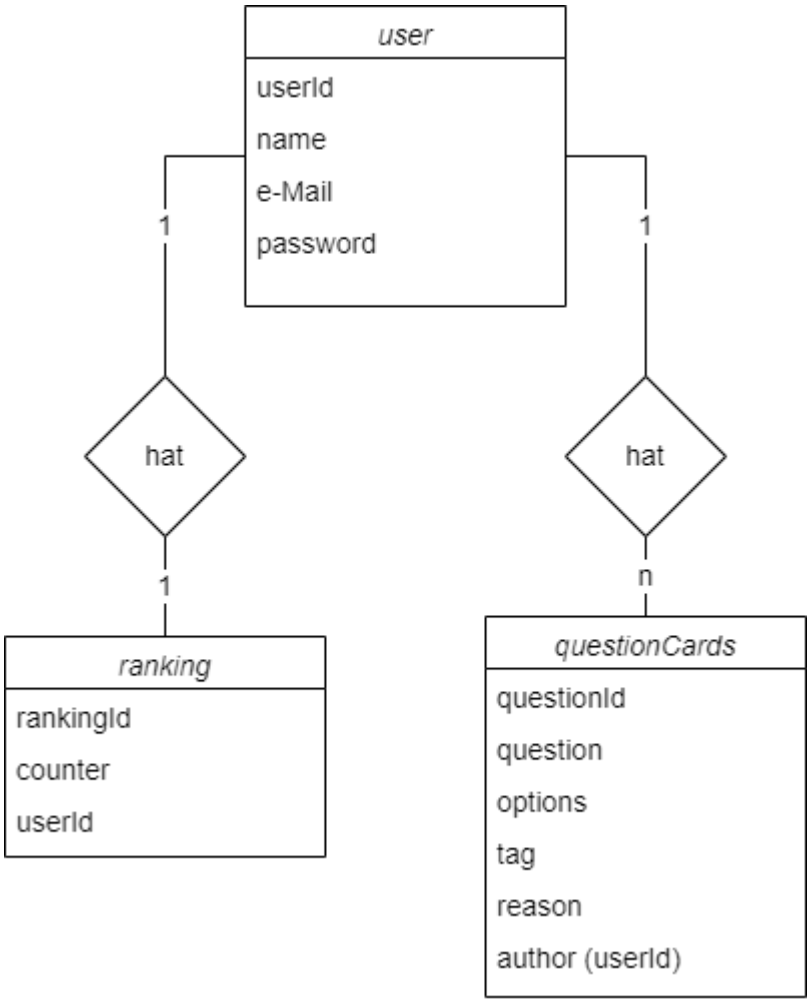
3.3 Weitere Tools

Wir verwenden Github als Versionskontrollsystem, da es mehreren Entwicklern ermöglicht, gleichzeitig an unserem Quiz-Projekt zu arbeiten und das ohne das wir uns gegenseitig beim Programmieren blockieren. Dies ermöglicht eine nahtlose Zusammenarbeit im Team und hilft Konflikte beim Zusammenführen von Code zu vermeiden. Zudem wird jede Änderung am Code protokolliert, sodass jederzeit nachvollzieht werden kann, welcher Entwickler was geändert hat und warum.

Wir verwenden VSCode als IDE zur Programmierung unseres Projektes. VSCode ist ein leistungsstarker und beliebter Code-Editor, welcher sich problemlos mit GitHub integrieren lässt und so eine ideale Ausgangssituation für die Entwicklung bietet. Zudem bietet VSCode eine benutzerfreundliche Oberfläche, in der es leicht ist, sich zu navigieren. Außerdem bietet VSCode eine breite Auswahl an möglichen Erweiterungen und integrierte Git Funktionalitäten. Ausschlaggebend für die Verwendung von diesem Code-Editor ist die kostenlose Nutzung dieses Tools.

4. Datenbankdesign

Abbildung 2 Datenbankdesign



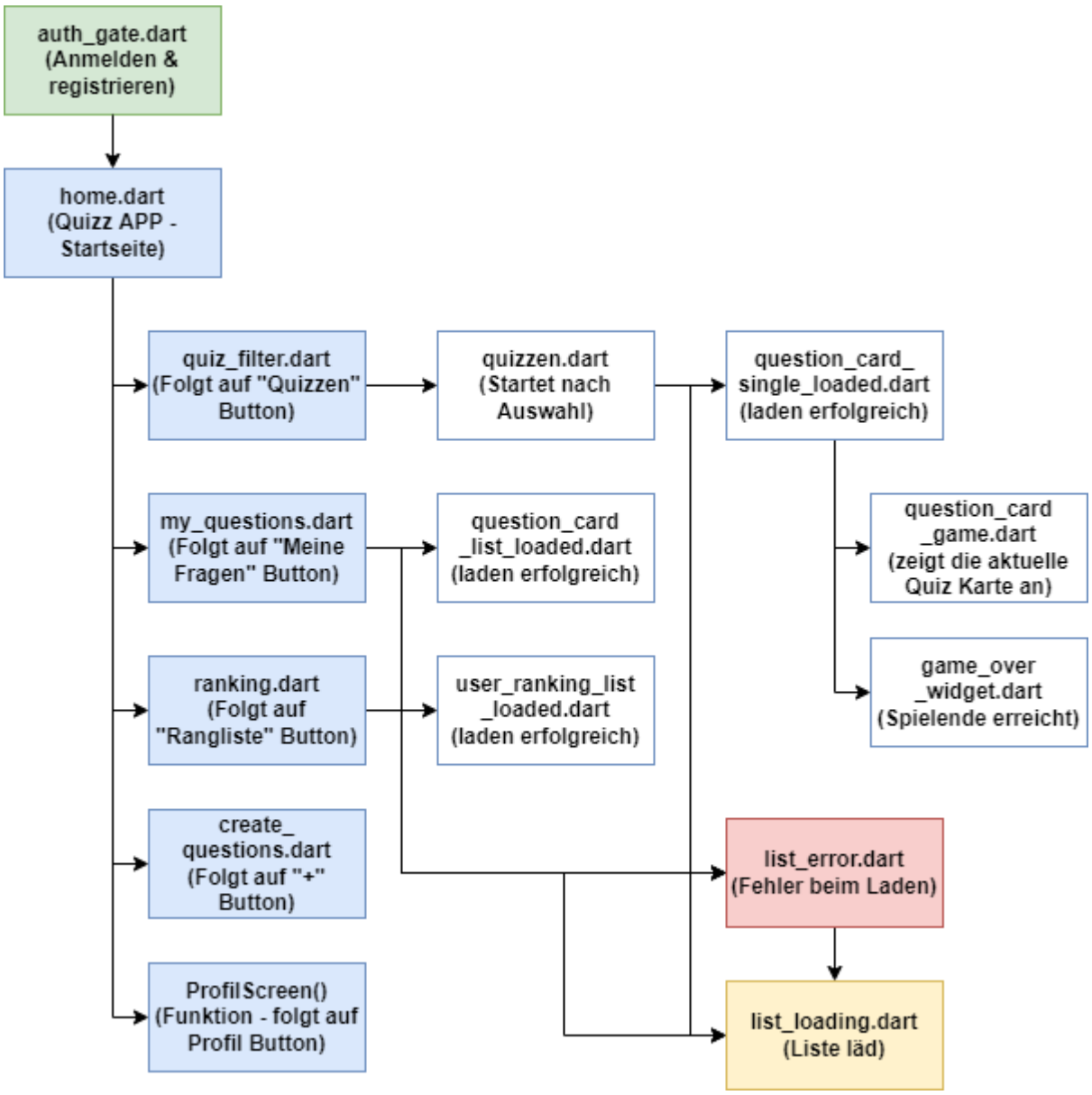
Quelle: Eigene Darstellung

5. Benutzeroberfläche

Die Benutzeroberfläche baut sich aus mehreren Hauptseiten (Screens) und deren Unterseiten (Widgets) zusammen auf:

Die Screens sind hier anhand der grünen und hellblauen Kästchen zu erkennen. Es startet mit der Authentifizierungsseite (auth_gate.dart). Danach folgt die Startseite (home.dart), von hier aus sind über die Buttons die weiteren Screens zu erreichen. Die weiß, gelb und rot hinterlegten Kästchen beschreiben dabei die Unterseiten also Widgets, welche nur von den jeweiligen Screens erreichbar sind. Das gelb markierte Widget bedeutet, dass die Liste lädt und das rot markierte Widget bedeutet, dass es einen Fehler beim Laden einer geforderten Liste gibt. Gelb und rot sind kritische Widgets, die für eine nicht performante Lösung sprechen.

Abbildung 3 Aufbau Benutzeroberfläche



Quelle: Eigene Darstellung

6. Test

6.1 Teststrategie

Damit die Anwendung optimal und in einem wirtschaftlich angemessenen Rahmen getestet wird, sollen 3 verschiedene Bereiche abgedeckt werden. Im ersten Test-Bereich erfolgt die Validierungstestung der einzelnen Text-Boxen. Diese umfasst zum einen, die Authentifizierungsmethoden mit Registrierung und Anmeldung, allerdings auch die Validierung für die Funktionalität Fragen hinzufügen. Der Zweite Testfall bezieht sich auf das Widget Testing. Hierbei wird das Verhalten der UI-Komponenten überprüft, sodass sichergestellt werden kann, dass die Widgets ordnungsgemäß gerendert werden und auf die Benutzerinteraktionen richtig reagieren. Die letzte Testabdeckung beschäftigt sich mit dem Anweisungsüberdeckungstest. Dieser wird verwendet, um das Verhalten der

ganzen Anwendungen zu überprüfen, um sicherzustellen das alle Anweisungen vom Code aus erreichbar sind.

6.2 Äquivalenzklassen

Tabelle 1 Äquivalenzklassen

Nr	Input	Gültige Äquivalenzklasse	Ungültige Äquivalenzklasse	Repräsentant	Soll-Ergebniss	Ist-Ergebniss
1	Strings	Alle Felder ausgefüllt	Mindestens 1 Feld nicht ausgefüllt	Alle Felder ausgefüllt außer das Modul	Frage nicht akzeptiert, Validierung: Bitte Modul auswählen	Frage nicht akzeptiert Validierung: Bitte Modul auswählen
2	Strings	Alle Felder ausgefüllt	Mindestens 1 Feld nicht ausgefüllt	Alle Felder ausgefüllt	Frage akzeptiert	Frage akzeptiert
3	Strings	Alle Felder ausgefüllt	Mindestens 1 Feld nicht ausgefüllt	Alle Felder ausgefüllt außer eine Antwort	Frage nichtakzeptiert, Validierung: Bitte Antwort eingeben	Frage nichtakzeptiert, Validierung: Bitte Antwort eingeben
4	Strings	Alle Felder ausgefüllt	Mindestens 1 Feld nicht ausgefüllt	Alle Felder ausgefüllt außer die Frage	Frage nichtakzeptiert, Validierung: Bitte Frage eingeben	Frage nichtakzeptiert, Validierung: Bitte Frage eingeben
5	Strings	Alle Felder ausgefüllt	Mindestens 1 Feld nicht ausgefüllt	Alle Felder ausgefüllt außer die Erläuterung	Frage nichtakzeptiert, Validierung: Bitte Erläuterung eingeben	Frage nichtakzeptiert, Validierung: Bitte Erläuterung eingeben
6	E-Mail korrekt, Passwort leer	E-Mail + Passwort richtig	E-Mail + Passwort falsch	E-Mail: test@test.de Pw: leer	Validierung: Password is required	Passwort is required
7	E-Mail korrekt, Passwort korrekt	E-Mail + Passwort richtig	E-Mail + Passwort falsch	E-Mail: test@test.de Pw: korrektes Passwort	Anmeldung erfolgreich	Anmeldung erfolgreich
8	E-Mail korrekt, Passwort falsch	E-Mail + Passwort richtig	E-Mail + Passwort falsch	E-Mail: test@test.de Pw: falsches Passwort	Anmeldung fehlgeschlagen, Validierung: The supplied auth credential is incorrect, malformed or has expired.	Anmeldung fehlgeschlagen, Validierung: The supplied auth credential is incorrect, malformed or has expired.
9	Registrierung, Password und confirmPassword stimmen	Password und confirmPassword stimmen überein	Password und confirmPassword stimmen nicht überein	Password und confirmPassword stimmen nicht überein	Registrierung fehlgeschlagen, Validierung: Passwords do not match	Registrierung fehlgeschlagen, Validierung: Passwords do not match

	nicht über-ein					
10	Registrie-rung, Pass-word und confirm-Password stimmen überein	Password und con-firmPass-word stimmen überein	Password und con-firmPass-word stim-men nicht überein	Password und confirmPass-word stimmen überein	Registrierung erfolgreich	Registrierung erfolgreich

6.3 WidgetTest

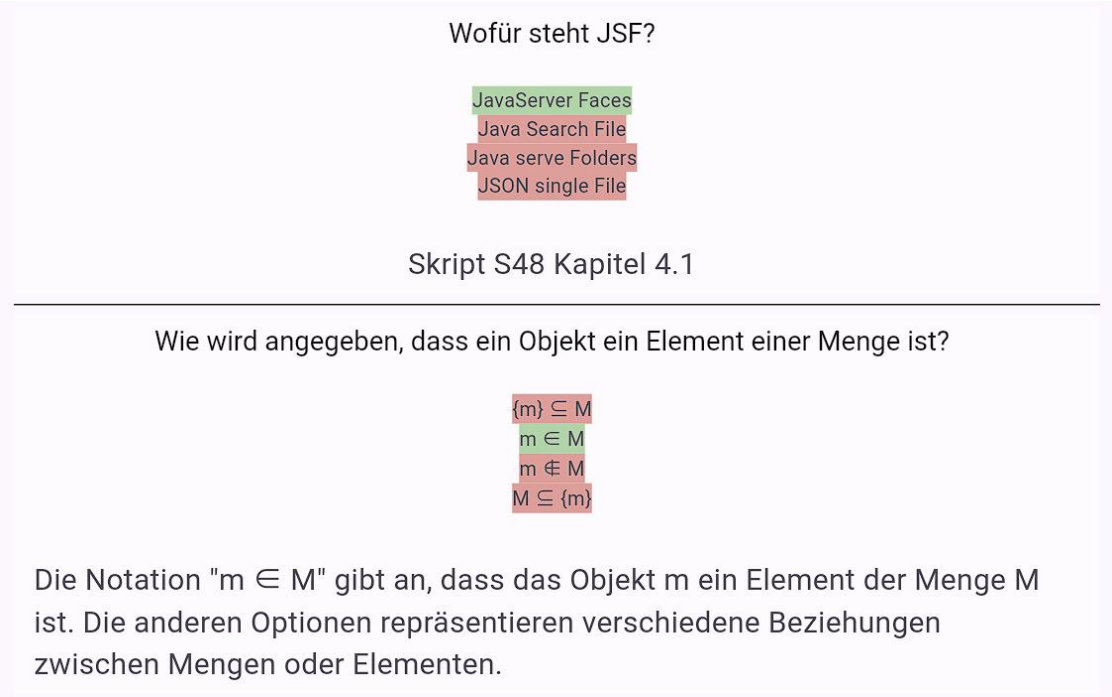
Der Flutter WidgetTest bietet die Möglichkeit zu überprüfen ob die gewünschten Widgets richtig gerendert werden. In diesem Projekt wurde der Test für den Widget Aufruf “Meine Fragen” - question_card_list_loaded.dart durchgeführt. In dem Test wird geprüft, ob das ListView-Widget welches die Einträge aus der Datenbank anzeigt, ausgegeben wird.
In Kombination mit der Erweiterung GoldenTest wird eine *.png Datei generiert, die eine Ausgabe in Blöcken anzeigt:

Abbildung 4 Ausgabe des WidgetTest

Der Schwarze obere Block stellt dabei die Frage aus der ListView wieder. Der grün markierte Blockbereich zeigt die richtige Antwort und die falschen sind in rot markiert.
Den Abschluss unterhalb der farbigen Blöcke stellt den Grund dar. Es ist zu erkennen, dass in dem Test lediglich zwei Testdaten in die ListView eingespielt wurden.

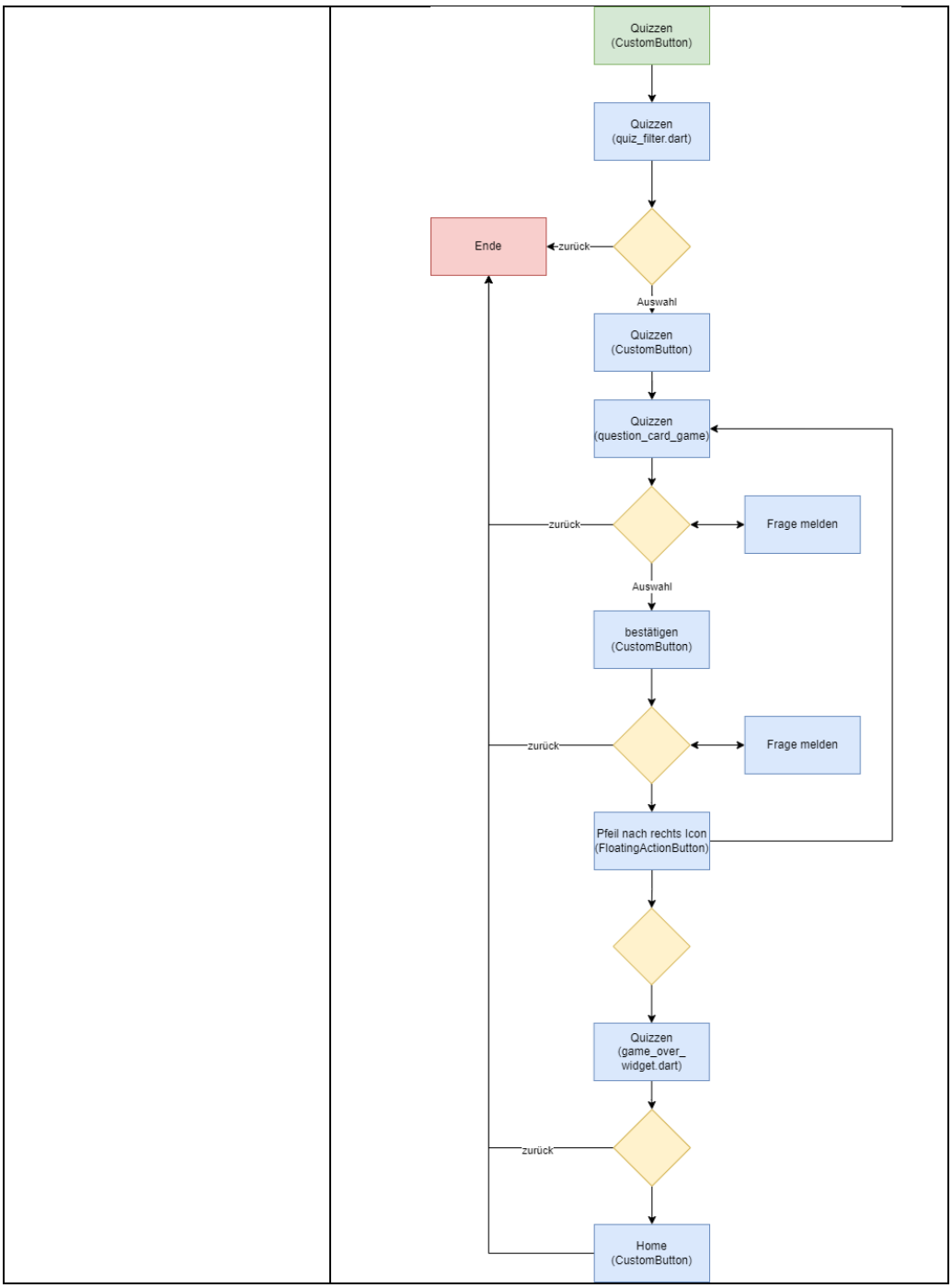
Hier der Vergleich zu den Produktivdatensatz:

Abbildung 5 Produktivdatensatz



6.4 White-Box Test - Anweisungsüberdeckungstest

Der Anweisungsüberdeckungstest stellt sicher, dass die einzelnen Anweisungen durchlaufen werden können. Mithilfe dieses Tests wird die Vollständigkeit des Quellcodes gewährleistet.



Um die Wirtschaftlichkeit des Testens zu gewährleisten, wurden diese Tests in Form von Repräsentanten gewählt. Dies ermöglicht eine effiziente und Zeitsparende Entwicklung, wie auch eine Kosteneffiziente Entwicklung der Webanwendung, da die Tests bekanntlich nicht die Abwesenheit von Softwarefehlern vorzeigen, sondern lediglich potenzielle Fehler darstellt. Mithilfe der Äquivalenzklassen, dem WidgetTest und des Anwendungsüberdeckungstests konnten wir so unsere Anwendung wirtschaftlich angemessen und mit gutem Bewusstsein Testen und den ersten Release anpeilen.

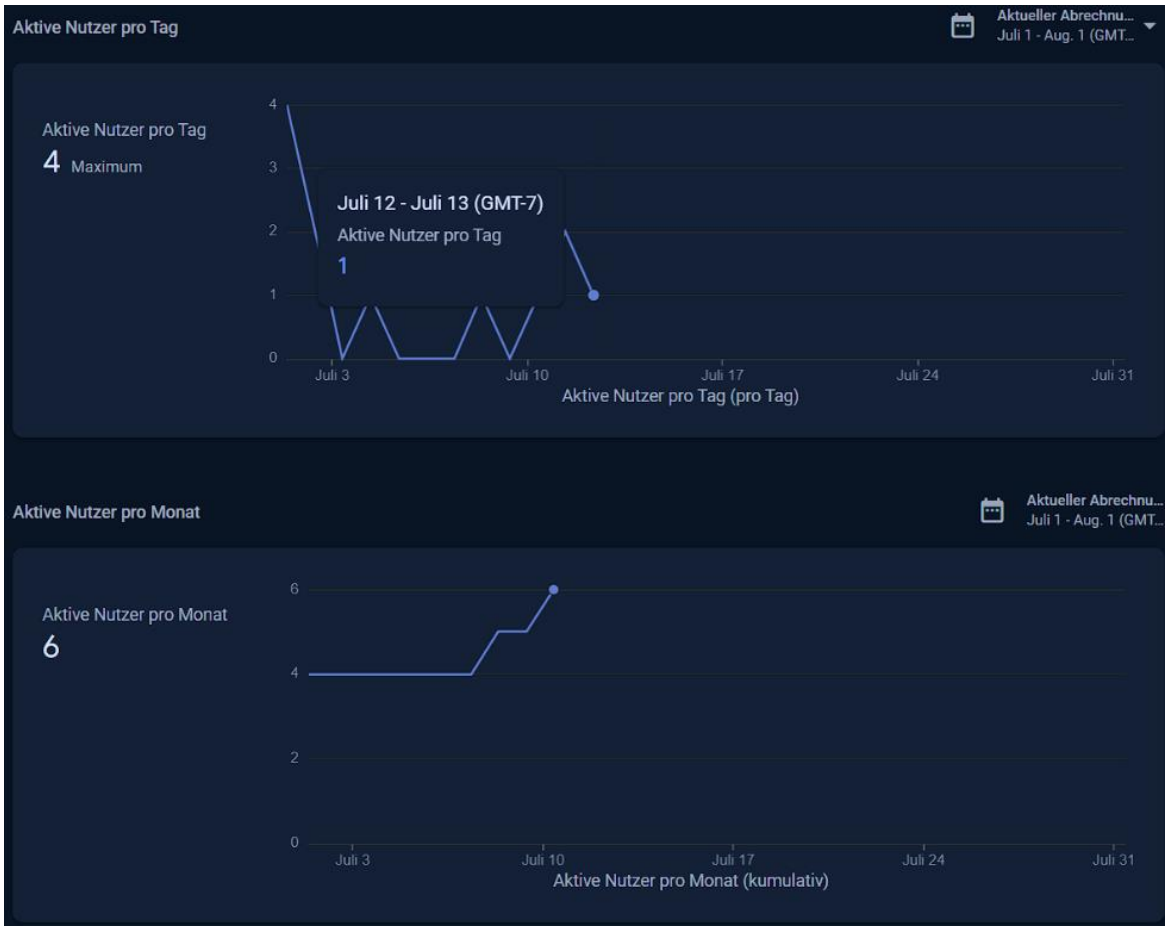
6.5 Analyse der Testergebnisse

Aufgrund der Ergebnisse der Äquivalenzklassen können wir uns sicher sein, dass die Validierung ordnungsgemäß funktioniert. In 10 von 10 Fällen stimmt das Soll-Ergebnis mit dem Ist-Ergebnis überein. Ebenso wurde der WidgetTest, als auch der Anwendungsüberdeckungstest erfolgreich und ohne Komplikationen abgeschlossen.

7. Sicherheit

Eine Anforderung unserer Quiz-App ist eine Benutzerauthentifizierung, sodass wir die Anmeldungen von Nutzern mit unserem Logging Werkzeug nachvollziehen können. Zudem bietet die Benutzerauthentifizierung ein sicheres Anmeldeverfahren.

Abbildung 6 Aktive Nutzer



Quelle: Screenshot aus Firebase

Die Datenverschlüsselung in Firebase Firestore wird mit einer symmetrischen AES-Verschlüsselung mit einer Schlüssellänge von mindestens 256 Bit gewährleistet. Dieser wird in unserem System als kryptografisches Verfahren verwendet.

Eine sichere Kommunikation zwischen Client und Server wird mit einem TLS-Zertifikat auf Firebase gewährleistet.

Mit einer angemessenen Validierung stellen wir sicher, dass keine Daten missbräuchlich abgefragt werden können. Diese umfasst eine Validierung bei der Passwortwahl und der Eingabe in Textfeld, wenn man eine neue Frage hinzufügt, sodass keine gleichen Fragen mehrfach in unserer Datenbank vorkommen können. Zudem werden Fehlermeldungen ausgegeben, wenn Pflichtfelder nicht ausgefüllt werden, sodass keine unvollständigen Datensätze in die Datenbank gelangen.

7.1 Sicherheitsrichtlinien

Passwortrichtlinie: Das Passwort unserer Nutzer beträgt eine Zeichenlänge von sechs beliebiger Zeichen nach UTF-8-Kodierung. Diese wurde getroffen, um die Möglichkeit von Brute-Force Attacken von potentialen Angreifern zuvorzukommen

8. Ansprechpartner

- Technische Verantwortung
  Bei Fragen zum technischen Aufbau und Design der hier vorgestellten Web-Applikation, richten sie ihre Anliegen an flutter-quizz-a9ab4@web.de
- Bei Fragen zu Server-Ausfällen oder Problemen bei der Erreichbarkeit der Website richten sie Ihre Anliegen an flutter-quizz-a9ab4@web.de