

GeoQuiz – Dokumentation

<https://github.com/Lor0l/geoQuiz>

1. Beschreibung der Anwendung

1.1 Projektidee

Die Android-Anwendung GeoQuiz soll die Rahmenbedingungen für das Erstellen sowie das Spielen des Geländespiels Schnitzeljagd unter Nutzung einer OSMDroid Karte bieten.

Es gibt zwei Hauptfunktionen EDIT und PLAY. Erstere umfasst einen Editor, mit dem ein eigenes Spiel erstellt werden kann (Spiele im Editor werden im Folgenden als Projekt bezeichnet).

Im Editormenü können Projekte neu angelegt und bereits Vorhandene zum Bearbeiten ausgewählt oder gelöscht werden. Beim Bearbeiten können auf der angezeigten Karte Markierungen manuell oder automatisch (aktueller Standort) gesetzt werden. Nachdem eine Markierung gesetzt wurde, muss eine ortsbezogene Frage mit entsprechender Lösung eingegeben werden. Hinzugefügte Markierungen erhalten eine fortlaufende Nummer in deren Reihenfolge sie im Spiel freigeschaltet werden. Nachdem die gewünschte Anzahl an Markierungen gesetzt wurde, kann der Editor geschlossen und das Projekt im Editormenü zum Spielen freigegeben (veröffentlicht) werden. Nach dem Veröffentlichen kann ein Spiel nicht mehr editiert werden um Cheaten zu verhindern.

Im Spielmenü kann ein Spiel gestartet und gelöscht werden. Nach dem Start wird die erste Markierung mit der dazugehörigen Frage angezeigt. Wurde die korrekte Antwort eingegeben wird die nächste Markierung freigeschaltet.

Das Spiel ist beendet, wenn die letzte Aufgabe gelöst wurde oder der Spieler aufgibt.

Projekte und Spielstände werden lokal auf dem Gerät gespeichert.

1.2 Benutzeroberfläche

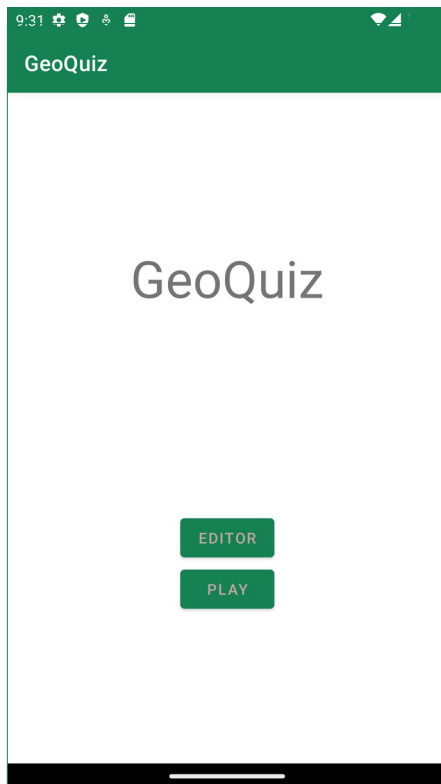


Abb. 1.1 Hauptmenü

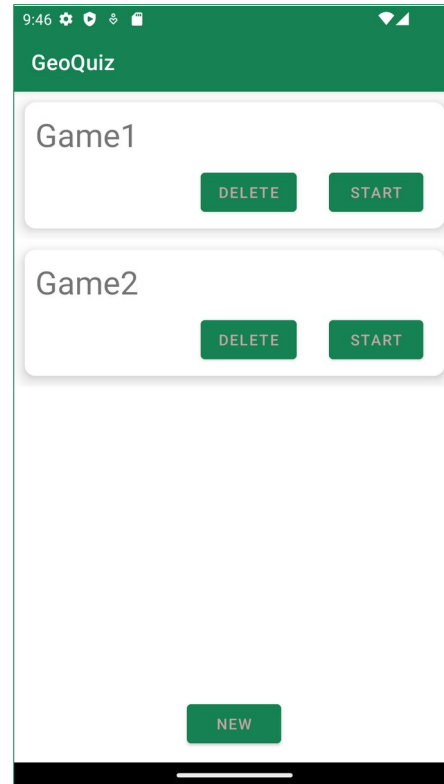


Abb. 1.2.1 Spielmenü

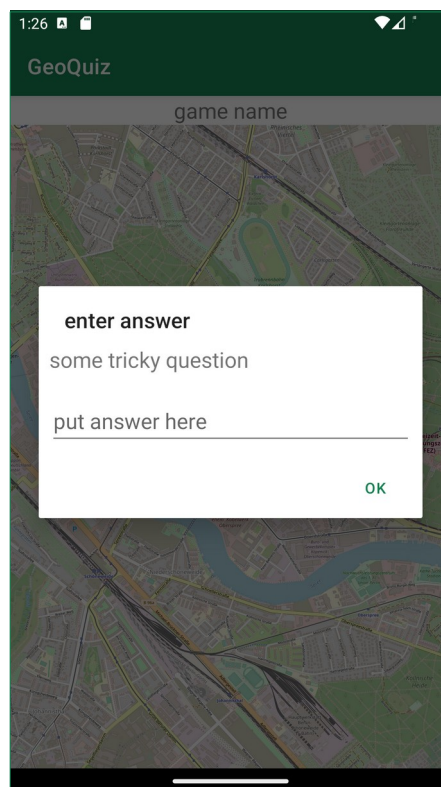


Abb. 1.2.2 im Spiel Antworteingabe

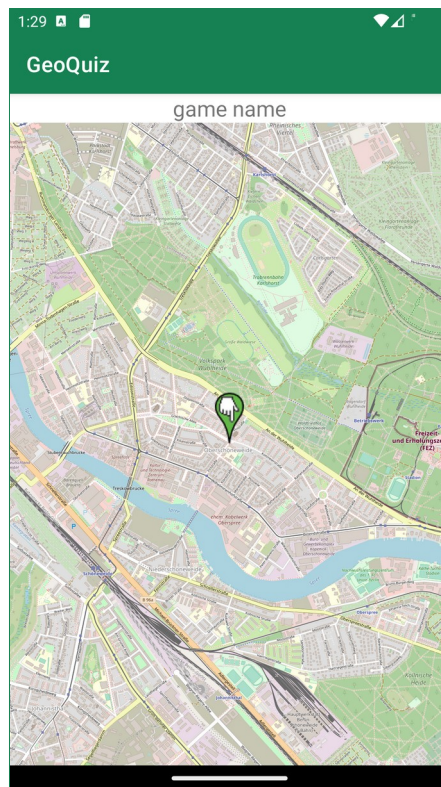


Abb. 1.2.3 im Spiel

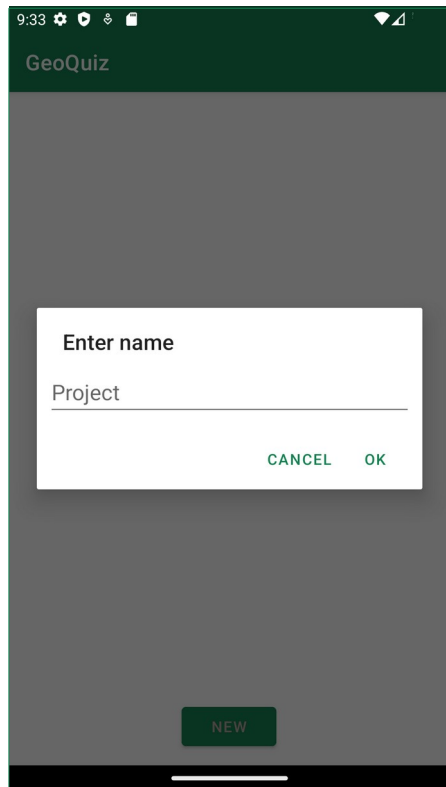


Abb. 1.3.1 Editormenü Projekt erzeugen

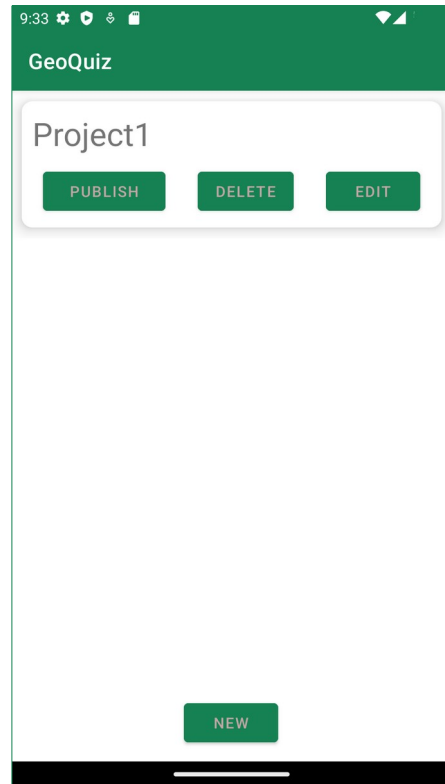


Abb. 1.3.2 Editormenü mit einem Projekt

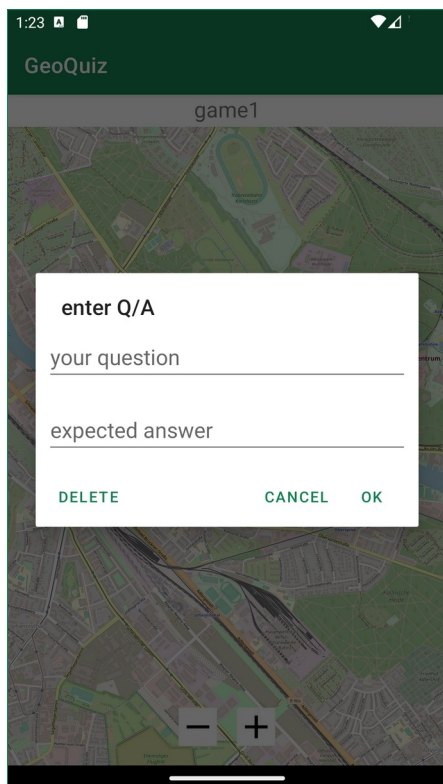


Abb. 1.3.3 Editor mit Q/A Eingabe

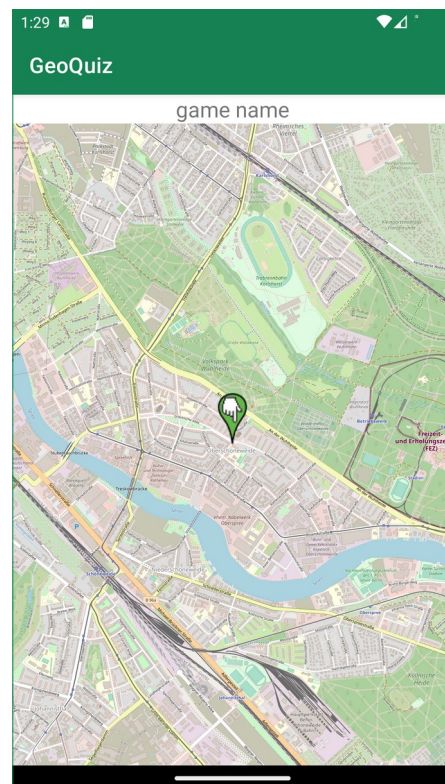


Abb. 1.3.4 Editor mit geöffnetem Projekt

2. Komponenten

2.1 Übersicht

Das Programm setzt sich aus den Komponenten **GeoQuizUI** (Visualisierung und User Input), **GeoQuizEngine** (Anwendungslogik), **MapData** (Datenstruktur und Speicherung) und **LocationProvider** (Standortermittlung) zusammen.

Zur besseren Lesbarkeit wird auf die Bezeichnung der Parameter an dieser Stelle verzichtet. Eine genaue Beschreibung folgt im Interface-Abschnitt der jeweiligen Komponente.

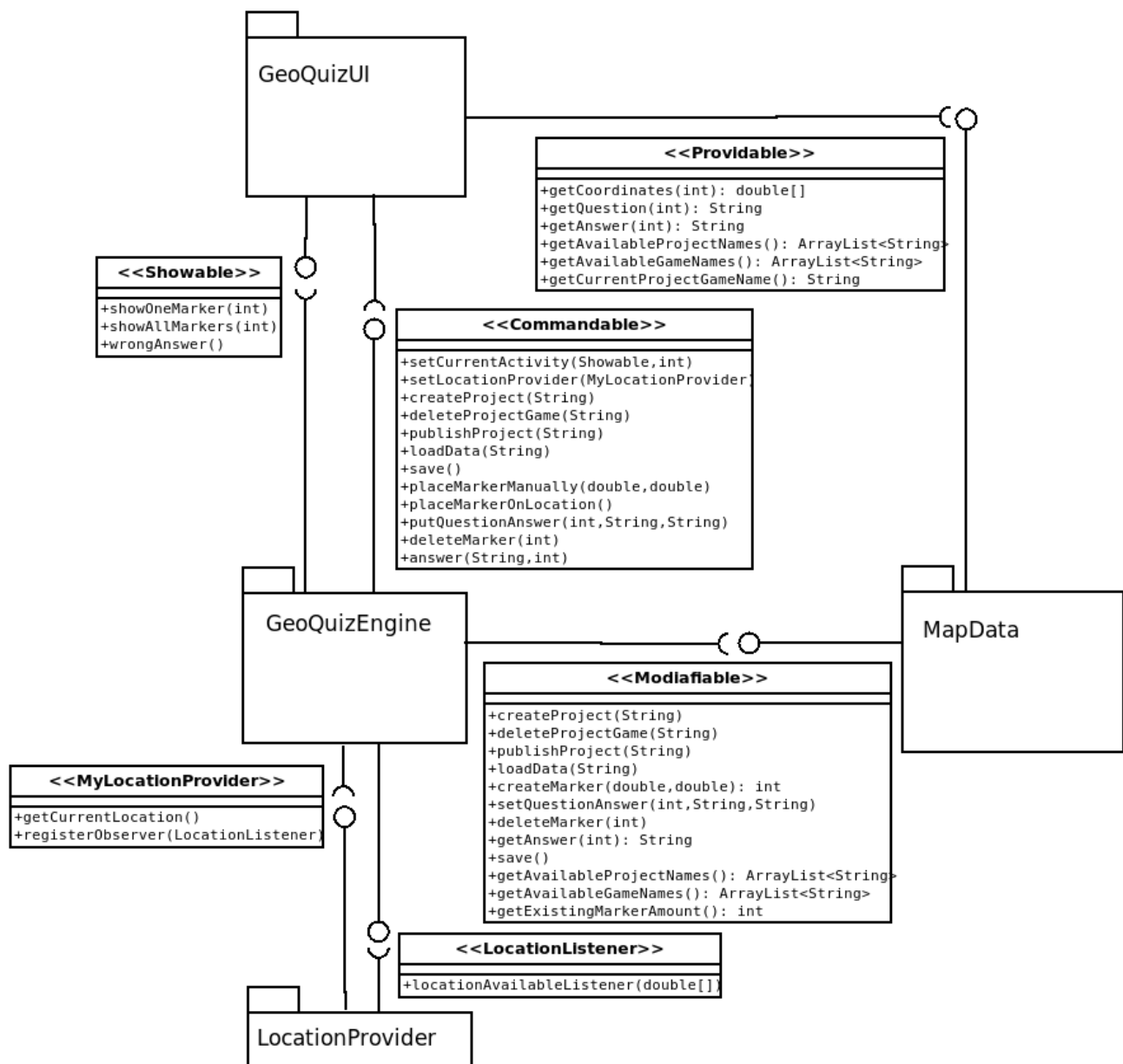


Abb. 2 Komponentendiagramm

2.2 GeoQuizUI

2.2.1 Zusammenfassung

Diese Komponente stellt die grafische Benutzeroberfläche für Android Betriebssysteme bereit und ist im Sinne des MVC Patterns ein View.

2.2.2 Struktur und Technik

Das Hauptmenü ist aus einem Constraint Layout aufgebaut und hat einen Textview mit dem Namen der Anwendung sowie zwei Buttons.

Im Editormenü ermöglicht ein Relative Layout mit Scroll View eine Auflistung von Projekten über die Beschränkung der Bildschirmanzeige hinaus. Beim Anlegen eines neuen Projekts öffnet sich über dem Scroll View ein Relative Layout mit editierbarem Text View für die Eingabe des Projektnamens. Nach Bestätigung schließt sich das Layout wieder und dem Scroll View wird dynamisch zur Laufzeit ein neues Layout mit Card View hinzugefügt. Dieser enthält den eingegebenen Namen und Buttons zum Starten, Veröffentlichen und Löschen.

Das Spielmenü hat ebenfalls einen Scroll View welcher schon beim Öffnen des Menüs alle wählbaren Spiele enthält. Hier kann die Anzahl der Layouts mit Card View nur durch Löschen eines Spiels verändert werden kann.

Für den Editor und das Spiel wird ein Constraint Layout verwendet, welches den Map View aus der OSMDroid Bibliothek und einen Text View enthält. Der Text View zeigt den Namen des geöffneten Projektes bzw. Spiels an. Die Eingabe von Antwort und Frage (Frage nur im Editor) ermöglicht ein Dialog, der angezeigt wird wenn man einen Marker auf der Karte anklickt.

Probleme bestanden beim Arrangieren des Layouts mit Map und Text View. Es ist nicht gelungen die Grenzen der Views in Relation zueinander zu setzen. Durch das Angeben von konkreten Layoutgrenzen in Pixeln ist zu erwarten, dass die GUI auf Geräten mit abweichender Bildschirmgröße bzw. Auflösung nicht wie gewünscht dargestellt wird. Möglicherweise entstehen dabei Probleme bei der Eingabe. Die Layouts wurden an ein Gerät mit 5.0“ Bildschirmdiagonale bei einer 1080x1920px Auflösung erstellt.

2.2.3 Interface

Über das Interface Showable wird das GeoQuizUI nach Veränderungen an MapData von der GeoQuizEngine zur Aktualisierung der Anzeige aufgerufen.

```
/**
 * information for view about changes on business logic so it can update the
 * GUI accordingly
 * EDITOR and GAME indicate the context in which the method supposed to be
 * called
 */
public interface Showable {

    /**
     * EDITOR/GAME
     * add marker to mapview
     * @param markerID
     */
    void showOneMarker(int markerID);

    /**
     * EDITOR
     * adds all markers when opening project
     * @param markerAmount
     */
    void showAllMarkers(int markerAmount);
}
```

```




/**
 * GAME
 * notify user that last answer was wrong
 */
void wrongAnswer();
}

```

2.2.4 Testkonzept

Um das User Interface zu testen werden Integrationstests mit Espresso durchgeführt. Hierbei wird getestet ob alle Komponenten der App richtig zusammen arbeiten also ob auf die in den Szenarios definieren Eingaben die gewünschten Ausgaben folgen. Die in den Tabellen beschriebenen Testszenarien wurden anhand folgender Use Case Kategorien erstellt.

- UC1 – Projekt erstellen/ löschen
- UC2 – Projekt bearbeiten
- UC3 – Projekt veröffentlichen
- UC4 – Spiel spielen

 nicht durchgeführt/
Test nicht erstellt
  erfolgreich
  fehlgeschlagen

UC1: testNewProject() positiv	
Scenario	Projekt mit eindeutigem Namen anlegen
Assertion	- neues Layout mit eingegebenem Namen und Buttons wird im Scroll View angezeigt

UC1: testScrollView() positiv	
Scenario	mehr Projekte anlegen als gleichzeitig angezeigt werden können
Assertion	- erstes und letztes Projekt können gestartet werden

UC1: testDeleteProject() positiv	
Scenario	Projekt aus Liste in Editor löschen
Assertion	- Layout des gelöschten Projekts wird nicht mehr angezeigt - neues Projekt mit gleichem Namen kann erzeugt werden und wird angezeigt

UC1: testNameTakenByProject() negativ	
Scenario	neues Projekt mit an existierendes Projekt vergebenen Namen anlegen
Assertion	- Toast Benachrichtigung - kein neues Layout mit bereits existierendem Namen hinzugefügt

UC1: testNameTakenByGame() negativ	
Scenario	neues Projekt mit an Spiel vergebenen Namen anlegen
Assertion	- Toast Benachrichtigung - kein neues Layout mit bereits existierendem Namen hinzugefügt

UC1: testSameSameButDifferentProject() rand	
Scenario	Projekt anlegen mit Namen der sich nur durch Großbuchstaben nicht in Zeichenfolge von anderem Projektnamen unterscheidet (<i>Eventuell kein echter Randfall?</i>)
Assertion	- neues Layout mit eingegebenem Namen und Buttons wird im Scroll View angezeigt

UC1: testSameSameButDifferentGame() rand	
Scenario	Projekt anlegen mit Namen der sich nur durch Großbuchstaben nicht in Zeichenfolge von anderem Spielnamen unterscheidet (<i>Eventuell auch kein echter Randfall?</i>)
Assertion	- neues Layout mit eingegebenem Namen und Buttons wird im Scroll View angezeigt

UC2: testOpenNewProject() positiv	
Scenario	neues Projekt öffnen
Assertion	- kein Marker, keine Frage und keine Antwort wird angezeigt

UC2: testFirstMarker() positiv	
Scenario	erster Marker mit Frage und Antwort wird manuell eingegeben
Assertion	- genau ein Marker wird auf der Karte an richtiger Position angezeigt - Frage und Antwort werden angezeigt

UC2: testNextMarker() positiv	
Scenario	weiterer Marker mit Frage und Antwort wird manuell eingegeben
Assertion	- Frage und Antwort werden an richtigem Marker angezeigt - andere Marker unverändert

UC2: testChangeQA() positiv	
Scenario	Frage und Antwort werden an existierendem Marker geändert
Assertion	- neue Frage und Antwort werden an richtigem Marker angezeigt - vorherige Frage und Antwort werden nicht mehr angezeigt - andere Marker unverändert

UC2: testMarkerOnLocationWithPermission() positiv	
Scenario	Berechtigung wird erteilt Marker wird mit Frage und Antwort an Standort eingegeben
Assertion	- Frage und Antwort werden an richtigem Marker an richtiger Position angezeigt - andere Marker unverändert

UC2: testDeleteMarker() <i>positiv</i>	
Scenario	mehrere Marker setzen dann einen Marker löschen
Assertion	- richtiger Marker verschwindet von Karte - andere Marker unverändert

UC2: testOpenExistingProject() <i>positiv</i>	
Scenario	bestehendes Projekt öffnen
Assertion	- bereits gesetzte Marker an richtiger Position mit richtiger Antwort und Frage werden angezeigt

UC2: testChangeAndReopenProject() <i>positiv</i>	
Scenario	bestehendes Projekt ändern und erneut öffnen
Assertion	- Marker mit Frage und Antwort werden entsprechend den Veränderungen (nicht mehr) angezeigt

UC2: testChangeAndReopenProject() <i>positiv</i>	
Scenario	bestehendes Projekt ändern und erneut öffnen
Assertion	- Marker mit Frage und Antwort werden entsprechend den Veränderungen (nicht mehr) angezeigt

UC2: testStopRestartProject() <i>positiv</i>	
Scenario	Projekt bearbeiten, zwischen Apps wechseln
Assertion	- Marker unverändert an richtiger Position

UC2: testDestroy() <i>positiv</i>	
Scenario	Projekt bearbeiten, App neu starten, Projekt öffnen
Assertion	- Marker unverändert an richtiger Position

UC2: testMarkerOnLocationNoPermission() <i>negativ</i>	
Scenario	(Berechtigung nicht erteilen) Marke wird an Standort eingegeben
Assertion	- kein neuer Marker wird hinzugefügt

UC2: testQAWithoutText() <i>negativ</i>	
Scenario	Marker setzen keine Frage/ Antwort eingeben
Assertion	- kein neuer Marker wird hinzugefügt

UC2: testQAEmpyString() <i>rand</i>	
Scenario	Marker setzen leeren String als Frage/ Antwort eingegeben
Assertion	- kein neuer Marker wird hinzugefügt

UC2: testQAOneChar() <i>rand</i>	
Scenario	Marker setzen String mit genau einem Zeichen wird als Frage/ Antwort eingegeben
Assertion	- neuer Marker wird hinzugefügt

UC3: testPublishMultMarker() <i>positiv</i>	
Scenario	Projekt mit Markern veröffentlichen
Assertion	- Projekt nicht mehr in Liste im Editormenü - Spiel mit selbem Namen in Liste im Spielmenü

UC3: testPublishWithoutMarker() <i>negativ</i>	
Scenario	neues Projekt (ohne Marker) veröffentlichen
Assertion	- Projekt wird immernoch in Liste im Editormenü angezeigt - kein neues Spiel wird in Liste in Spielmenü angezeigt - Toast Benachrichtigung

UC3: testPublishOneMarker() <i>rand</i>	
Scenario	Projekt mit genau einem Marker veröffentlichen
Assertion	- Projekt wird nicht mehr in Liste im Editormenü angezeigt - Spiel mit selbem Namen wird in Liste im Spielmenü angezeigt

UC4: testStartGame() <i>positiv</i>	
Scenario	Spiel starten
Assertion	- nur erster Marker mit zugehöriger Frage wird an richtiger Position angezeigt

UC4: testCorrectAnswer() <i>positiv</i>	
Scenario	richtige Antwort eingeben
Assertion	- Toast Benachrichtigung mit Bestätigung wird angezeigt - letzter Marker und Frage wird nicht mehr angezeigt - nächster Marker mit zugehöriger Frage wird angezeigt

UC4: testCorrectLastAnswer() positiv	
Scenario	richtige Antwort auf letzte Frage eingeben
Assertion	- Toastbenachrichtigung über Sieg - Activity wechsel zu Spielmenü - bei erneutem Starten desselben Spiels wird nur der erste Marker angezeigt
UC4: testWrongAnswer() positiv	
Scenario	falsche Antwort eingeben
Assertion	- Toastbenachrichtigung über falsche Antwort - angezeigter Marker und Frage ändern sich nicht

UC4: testGiveUp() positiv	
Scenario	Spiel beenden (aufgeben)
Assertion	- Activity wechsel zu Spielmenü - bei erneutem Starten desselben Spiels wird nur der erste Marker angezeigt

UC4: testStopRestartGame() positiv	
Scenario	in offenem Spiel zu anderer App und zurück wechseln
Assertion	- aktueller Marker mit Frage wird angezeigt (gleicher Zustand wie vor Wechsel)

UC4: testDestroy() positiv	
Scenario	bei offenem Spiel App neu starten
Assertion	- bei Start des gleichen Spiels wird aktueller Marker der letzten Sitzung angezeigt

Nach erfolgreichem Testen der einzelnen Methoden ist es sicherlich sinnvoll größere Testszenarien zu entwickeln, die unter anderem komplette Spielabläufe simulieren. Auf Grund der bereits sehr umfangreichen Testbeschreibung und aus Zeitgründen habe ich diese hier weggelassen.

Ein Großteil der oben definierten Tests konnte nicht mit Espresso umgesetzt werden, da keine Möglichkeit gefunden wurde in den UI-Tests auf die Objekte im Map View zuzugreifen. Es kann an dieser Stelle also nicht überprüft werden, ob die Marker richtig gesetzt wurden und die Richtigen Informationen beinhalten. Alternativ lassen sich das Setzen, Speichern und Laden der Informationen für die Marker der MapView auch durch Integrationstest der Komponenten GeoQuizEngine und MapData mit Hilfe von Unittests überprüfen.

2.3 GeoQuizEngine

2.3.1 Zusammenfassung

Die Engine interpretiert und verifiziert die User-Eingaben und wandelt diese in Befehle zur Veränderungen an den Datensätzen innerhalb der MapData um. Außerdem wird von hier aus bei Bedarf die GeoQuizUI zur Aktualisierung der Anzeige aufgerufen.

2.3.2 Struktur und Technik

Diese Komponente ist momentan noch im selben Android Projekt Implementiert da das Problem mit dem Aktualisieren der extern geschriebenen Bibliotheken in Android Studio noch nicht gelöst werden konnte. Es wird allerdings angestrebt die Engine als unabhängiges Javaprojekt zu erstellen und als mit Unit Tests getestetes Packet einzubinden.

Die Interfaces der Dependencies müssen ebenfalls in diesem Projekt erzeugt werden, damit diese Komponente extern geschrieben und getestet werden kann.

2.3.3 Interfaces

```
/**
 * view to control connection
 * EDITOR and GAME indicate the context in which the method supposed to be
 * called
 */

public interface Commandable {

    /**
     * EDITOR/GAME
     * register current activity
     * sets state and invokes GUI to initialize game or editor
     * @param currentActivity
     * @param state          EDITOR = 1
     *                      GAME = 2
     */
    void setCurrentActivity(Showable currentActivity, int state);

    /**
     * EDITOR
     * set connection to devices GPS location provider
     * @param LocationProvider
     */
    void setLocationProvider(MyLocationProvider LocationProvider);

    /**
     * EDITOR
     * crate a new project
     * @param projectID project name
     * @throws NameAlreadyExistsException when this name is already taken by an
     * existing project or game
     */
    void createProject(String projectID) throws NameAlreadyExistsException;

    /**
     * EDITOR
     * deletes existing project or game
     * @param projectGameID project/game name
     */
    void deleteProjectGame(String projectGameID);

    /**
     * EDITOR
     * convert project into game
     * @param projectID project name
     * @throws EmptyProjectException if project has no marker
     */
}
```

```

    */
    void publishProject(String projectID) throws EmptyProjectException;

    /**
     * EDITOR/GAME
     * load project/game data
     * @param name project/game name
     */
    void loadData(String name);

    /**
     * EDITOR
     * save project attributes permanently
     */
    void save();

    /**
     * EDITOR
     * create new marker in currently open project on given coordinates
     * @param latitude
     * @param longitude
     */
    void placeMarkerManually(double latitude, double longitude);

    /**
     * EDITOR
     * create marker on GPS location in currently open project
     */
    void placeMarkerOnLocation() ;

    /**
     * EDITOR
     * set question and answer belonging to a specific marker in open project
     * @param markerID
     * @param question
     * @param answer
     */
    void putQuestionAnswer(int markerID, String question, String answer);

    /**
     * EDITOR
     * delete a marker with related coordinates, question and answer in open
     * project
     * @param markerID
     */
    void deleteMarker(int markerID);

    /**
     * GAME
     * check if given answer matches saved answer (case insensitive)
     * @param answer answer given by user
     * @param markerID marker where answer belongs to
     */
    void answer(String answer, int markerID);
}

/**
 * LocationListener objects will be notified once a MyLocationProvider object
 * obtained a location
 */
public interface LocationListener {

    /**
     * send coordinates to subscribed observer
     * @param coordinates
     */
    void locationAvailableListener(double[] coordinates);
}

```

2.3.4 Testkonzept

Diese Komponente soll androidunabhängig mit JUnit5 getestet werden. Dazu werden 2 Testklassen für Mockobjekte erstellt, die das Verhalten der Komponenten GeoQuizUI so wie MapData anhand entsprechender Interfaces simulieren. Mit Countern in den Methoden der Testklassen kann nachvollzogen werden welche Methode wie oft in einem Szenario aufgerufen wurde um korrektes Verhalten zu überprüfen.

In den Tabellen ist das Testszenario beschreiben sowie die Phasen SetUp, Exercise und Assertion. Die Methoden in Exercise beziehen sich auf das zu testende Objekt, die Methoden aus Assertion auf die Mockobjekte.

Auf das Testen einfacher Setter so wie das „Weiterreichen“ von Befehlen an MapData, an denen keine prüfende Logik der Engine beteiligt ist wird aufgrund der überschaubaren Fehleranfälligkeit verzichtet.

■ nicht durchgeführt

■ erfolgreich

■ fehlgeschlagen

setCurrentActivity()

testSetCurrentActivityEditor() positiv	
Scenario	korrektes Registrieren der Editor Activity und Aufruf an UI alle existierenden Marker anzuzeigen
SetUp	Showable Mockobjekt (SM) erzeugen; MapData Mockobjekt (MDM) erzeugen und Rückgabewert (RET) von getExistingMarkerAmount() definieren; GeoQuizEngine Objekt (GEO) erzeugen
Exercise	setCurrentAcitivity() wird mit entsprechenden Parametern für Editor aufgerufen
Assertion	Methode zum Anzeigen aller Marker auf das SM mit festgelegter Anzahl aufgerufen

testSetCurrentActivityGame() positiv	
Scenario	korrektes Registrieren der Game Activity und Aufruf an UI, den ersten Marker anzuzeigen
SetUp	SM erzeugen; MDM erzeugen; GEO erzeugen
Exercise	setCurrentAcitivity() wird mit entsprechenden Parametern für Game aufgerufen
Assertion	Methode zum Anzeigen eines Markers mit Parameter 0 auf das SM aufgerufen; keine Methode auf MDM aufgerufen

createProject()

testCreateProjectNewName() positiv	
Scenario	Erzeugen eines Projektes mit noch nicht vergebenem Namen
SetUp	MDM erzeugen und Liste für RET von getAvailableProjectNames() definieren; GEO erzeugen
Exercise	CreateProject() mit eindeutigem Namen als Parameter aufrufen
Assertion	getNames() aufgerufen; createProjekt() mit gleichem Namen aus Exercise als Parameter aufgerufen keine Exception geworfen

testCreateProjectTakenName() negativ	
Scenario	Erzeugen eines Projektes mit vergebenem Namen
SetUp	MDM erzeugen und Liste für RET von getNames() definieren; GEO erzeugen
Exercise	CreateProject() mit Namen aus Liste als Parameter aufrufen
Assertion	getNames() aufgerufen; NameAlreadyExistsException geworfen

publishProject()

testPublishProject2Markers() positiv	
Scenario	Veröffentlichen eines Projektes mit 2 Markern
SetUp	MDM erzeugen und RET getExistingMarkerAmount() mit 2 definieren; GEO erzeugen
Exercise	publishProject() mit Namen aufrufen
Assertion	loadData() mit Name einmal aufgerufen; getExistingMarkerAmount() einmal aufgerufen; publishProject() einmal aufgerufen

testPublishProject1Marker() rand	
Scenario	Veröffentlichen eines Projektes mit 1 Marker
SetUp	MDM erzeugen und RET getExistingMarkerAmount() mit 1 definieren; GEO erzeugen
Exercise	publishProject() mit Namen aufrufen
Assertion	loadData() mit Name einmal aufgerufen; getExistingMarkerAmount() einmal aufgerufen; publishProject() mit Name einmal aufgerufen

testPublishProjectNoMarker() <i>negativ</i>	
Scenario	Veröffentlichen eines Projektes mit keinem Marker
SetUp	MDM erzeugen und RET getExistingMarkerAmount() mit 0 definieren; GEO erzeugen
Exercise	publishProject() mit Namen aufrufen
Assertion	loadData() mit Name einmal aufgerufen; getExistingMarkerAmount() einmal aufgerufen; EmptyProjectException geworfen publishProject() nicht aufgerufen

answer()

testCorrectAnswer() <i>positiv</i>	
Scenario	Eingabe korrekter Antwort im Spiel (ignore case)
SetUp	SM erzeugen MDM erzeugen und RET getAnswer(0) mit „correct answer“ definieren; GEO erzeugen
Exercise	answer(0) mit „CORRECT ANSWER“ aufrufen
Assertion	getAnswer(0) auf MDM aufgerufen showMarker(1) auf SM aufgerufen

testWrongAnswer() <i>positiv</i>	
Scenario	Eingabe falscher Antwort im Spiel
SetUp	SM erzeugen MDM erzeugen und RET getAnswer() mit „correct answer“ definieren; GEO erzeugen
Exercise	answer(0) mit „wrong answer“ aufrufen
Assertion	getAnswer(0) auf MDM aufgerufen wrongAnswer() auf SM aufgerufen

2.4 MapData

2.4.1 Beschreibung

Diese Komponente organisiert alle Informationen die zusammen ein Projekt bzw. Spiel bilden. Sie regelt außerdem das Laden und Speichern der Daten.

2.4.2 Struktur und Technik

Projekte und Spiele unterscheiden sich darin, in welcher Liste der Name gespeichert ist. Wenn ein Projekt veröffentlicht wird, wird es in die Spiel-Liste verschoben und kann nicht mehr mit dem Editor geöffnet werden. Ein Projekt/Spiel umfasst ein oder mehrere Markerobjekte. Die Informationen zu einer Station eines Projekts/Spiels in folgenden Attributen speichert.

- id (integer) - legt außerdem die Reihenfolge des Freischaltens fest
- latitude (double)
- longitude (double)
- question (String)
- answer (String)

Soll ein Projekt/Spiel gespeichert werden, werden alle Attribute der Markerobjekte mithilfe von Shared Preferences lokal gespeichert. Der Key setzt sich dabei aus Projekt/Spielname-MarkerID-Attributname zusammen und ist damit eindeutig. Außerdem wird je ein String Set mit allen Namen der gespeicherten Projekte/Spiele unter dem Key geoQuiz_procejs bzw. geoQuiz_games ebenfalls lokal gespeichert.

Beim Erzeugen der Instanz von MapDataImpl werden die Namen der aller Projekte und Spiele in die entsprechende Liste geladen.

Zur Laufzeit der Anwendung kann immer nur ein Projekt oder Spiel mit allen zugehörigen Markerobjekten geladen werden.

2.4.2 Interfaces

Diese Komponente hat zwei Interfaces. Modifiable stellt die Verbindung zur Engine und Providable die Verbindung zur GUI.

```
/**
 * instructions for the business logic to work on the data
 * EDITOR and GAME indicate the context in which the method supposed to be
 * called
 */
public interface Modifiable {

    /**
     * EDITOR
     * crate a new empty project
     * @param projectName project ID
     */
    void createProject(String projectName);

    /**
     * EDITOR
     * delete existing project or game
     * @param projectGameName project/game ID
     */
    void deleteProjectGame(String projectGameName);

    /**
     * EDITOR
     * convert project into game
     * @param projectName project ID
     */
}
```

```

*/
void publishProject(String projectName) ;

/**
 * EDITOR/GAME
 * load project/game data from shared preferences
 * @param name name
 */
void loadData(String name);

/**
 * EDITOR
 * create marker in open project
 * @param latitude
 * @param longitude
 * @return marker ID
 */
int createMarker(double latitude, double longitude);

/**
 * EDITOR
 * set question and answer belonging to a specific marker
 * @param markerID
 * @param question
 * @param answer
 */
void setQuestionAnswer(int markerID, String question, String answer);

/**
 * EDITOR
 * delete a marker and related coordinates, question and answer
 * @param markerID
 */
void deleteMarker(int markerID);

/**
 * GAME
 * return answer related to a marker
 * @param markerID
 * @return answer
 */
String getAnswer(int markerID);

/**
 * EDITOR/GAME
 * write values of open project or game into local file
 */
void save();

/**
 * EDITOR/GAME
 * return a list of available projects
 * @return available projects
 */
ArrayList<String> getAvailableProjectNames();

/**
 * EDITOR/GAME
 * return a list of available games
 * @return available games
 */
ArrayList<String> getAvailableGameNames();

/**
 * EDITOR
 * get IDs of existing markers (ID of last Marker == amount)
 * @return amount
 */
int getExistingMarkerAmount();
}

```

```

/**
 * view to model connection
 */
public interface Providable {

    /**
     * EDITOR/GAME
     * returns coordinates related to a marker
     * @param markerID
     * @return double[] [0] = latitude
     *                  [1] = longitude
     *          null if marker with this id has been deleted to keep marker ID's
     *          matching list index
     * @throws IndexOutOfBoundsException when markerID is larger than marker
     *          amount
     */
    double[] getCoordinates(int markerID);

    /**
     * EDITOR/GAME
     * returns the question belonging to a specific Marker
     * @param markerID
     * @return question
     */
    String getQuestion(int markerID);

    /**
     * EDITOR
     * returns the answer belonging to a specific Marker
     * @param markerID
     * @return answer
     */
    String getAnswer(int markerID);

    /**
     * EDITOR
     * returns a list of available projects
     * @return available projects
     */
    ArrayList<String> getAvailableProjectNames();

    /**
     * GAME
     * returns a list of available games
     * @return available games
     */
    ArrayList<String> getAvailableGameNames();

    /**
     * EDITOR/GAME
     * returns the name of the loaded project/game
     * @return
     */
    String getCurrentProjectGameName();
}

```

2.4.3 Testkonzept

Umfangreiche Tests sind an dieser Stelle nicht nötig, da der Datentyp Marker wenig komplex ist und sich hinter den Gettern und Settern übersichtlicher Code verbirgt. Für das Speicherkonzept mittels der Shared Preferences gilt das Selbe. Fehler würden wahrscheinlich bei den Integrationstests auffallen.

Integrationstests sind allerdings durch Auslagern der Nutzung von Shared Preferences in einem eigenen Objekt möglich. So kann dieses bei den Tests durch ein Mockobjekt ersetzt werden und androidunabhängig getestet werden

2.5 LocationProvider

2.5.1 Beschreibung

Diese Komponente stellt den Kontakt zwischen der GeoQuizEngine und dem Location Manager von Android über welchen auf Anfrage der aktuelle Standort bereitgestellt werden soll. Um die GPS Koordinaten zu erhalten, registriert sich die Engine als Observer beim LocationProvider und wird benachrichtigt, sobald das Betriebssystem die Daten zur Verfügung stellt.

2.5.2 Interface

```
/**
 * engine - location provider connection
 */
public interface MyLocationProvider {

    /**
     * get current location from device, if successful notify observer otherwise
     * inform user
     */
    void getCurrentLocation();

    /**
     * register observer that want to be notified when location is available
     * @param observer
     */
    void registerObserver(LocationListener observer);
}
```

2.5.3 Testkonzept

vgl. 2.4.3