Projekt Dokumentation - Projekttagebuch

Merlin Tschann:

Projektstart

28. April

- Flutter-Projekt erstellt
- Projektideen gesammelt und ausgewählt

Karten-Design

29. April

- Herz-Karten mit KI-Image-Generator erstellt

2. Mai

- Weitere Karten erstellt

Datenbank und Backend

6. Mai

- Supabase-Datenbank erstellt
- Erster Verbindungsversuch mit Flutter

3. Juni

- Datenbankstruktur überarbeitet (Spielernummer ergänzt)

11. Juni

- Weitere Datenbank-Updates

Spiellogik und Funktionen

8. Mai

- Dialekt-Namensgenerator mit klassischen Vorarlberger Wörtern (später entfernt)

10. Mai

- Erster Multiplayer-Versuch: Räume wurden angezeigt

21. Mai

- Beitritt über 5-stelligen Raumcode (als Primärschlüssel)

28. Mai

- Fehlerbehebungen bei den Räumen

2. Juni

- Karten-Auswertungsfunktion erstellt

5. Juni

- Beginn der Implementierung eines Streams für Spielzüge

6. Juni

- Trumpf-Auswahlfenster erstellt

13. Juni

- Regeln implementiert: Welche Karte darf gespielt werden

14. Juni

- Punktezählung begonnen
- Spielernamen unter Avataren angezeigt
- Benutzeroberfläche überarbeitet

15. Juni

- Punktezählung korrigiert
- Spieleranzeige: Wer ist am Zug (Version 1)

16. Juni

- Anzeige für gewählten Trumpf erstellt

Fehlerbehebungen und Optimierungen

4. Juni

- Allgemeine Fehlerbehebungen

6. Juni

- Weitere Bugfixes

15. Juni

- Zusätzliche Fehlerbehebungen
- Logging integriert
- Unit-Tests für das Frontend erstellt
- Präsentation erstellt und vorbereitet

Thomas Schallert:

Planung:

28. April

- Projektideen gesammelt und ausgewählt
- ERM und Klassendiagramme gezeichnet

Projekt initialisierung

08. Mai

- Alle Klassen der davor gezeichneten Diagramme initialisiert

Spiellogik und Funktionen

21. Mai

- Grunddesign für die Spiele erstellt
- Drag and Drop der Karten hinzugefügt

29. Mai

- Gemacht das man auf 4 Spieler warten muss

01. Juni

- Gemacht dass Karten unter den Spielern aufteilen

04. Juni

- Gemacht das immer neue Runden gestartet werden

06. Juni

- Gemacht das immer nur einer am Zug ist und spielen kann

07. Juni

- Anzeige der gespielten Karten der anderen Spielern

09. Juni

- Wenn 4 Karten gespielt wurden cleared es das drop Feld
- Eine neue Runde beginnt

13. Juni

- Zählen der Punkte hinzugefügt

15. Juni

- Update der Game UI
- Farbzwang hinzugefügt (mit Fehlern)
- Gemacht wer die Runde gewinnt + logic (das der wieder beginnt...)

17. Juni

- Doxygen Kommentare hinzugefügt

18. Juni

- Backend Unit-Test hinzugefügt
- Doxygen generiert

Swagger

28. Mai

- Swagger angefangen (Aber aufgehört)

14. Juni

- Swagger finalisiert mit paar bugs

Lastenheft

Wir entwickeln eine **Vorarlberg Jass App**.

Damit erfüllen wir die Projektanforderungen:

Frontend (POS)

- Flutter-App mit mindestens 3 Screens (z.B. Startseite, Spiel erstellen, Spielübersicht).
- Verwendung von GIT & GitHub (mit Kanban-Board).
- Einsatz von abstrakten Klassen und Vererbung:
    - **Abstrakte Klasse:** Modi
        - Eigenschaften: Spieleranzahl, Karten pro Spieler, Wer macht Trumpf
    - **Interface:** Spielkarte
        - Eigenschaften: Farbe (Herz, Kreuz, Pik, Karo), Wert (7, 8, 9, 10, Unter, Ober, König, Ass), Punktwert (z.B. 0, 0, 0, 0, 10, 2, 3, 4, 11 usw.)
- Unit-Tests, z.B. für Regeln

Backend (DBI)

- Supabase-Datenbank für Spieler-, Spiele- und Ergebnisdaten.
- REST-API (z.B. Python FastAPI) für die Kommunikation zwischen Frontend und Backend.
- Benutzerrollen: „admin“ und „user“.
- API-Endpunkte zur Abfrage von:
    - aktuellem Spielstand
    - Punkten
    - Rankings

Must-Have Features

- Fixe Jass-Regeln (Vorarlberger Jass)

- Modus Kreuzjassen
- Speicherung von Spielen und Spielergebnissen

Nice-to-Have Features

- Multiplayer-Modus (Lobby-System) (z.B. über Firebase)
- KI-Gegner
- Karten-Skins
- Mehr Spielmodi
- Schöne Animationen und Übergänge

Technik

- **Flutter** bietet viele vorgefertigte UI-Elemente, gute Performance für Spiele, gute Dokumentation (auch für Games) und unterstützt Multi-Platform-Entwicklung (auch Web).
- **FastAPI (Python)** ermöglicht uns eine schnelle und flexible Entwicklung der REST-API.
- **MariaDB** Speicherung aller Spiel- und Userdaten.
- **Firebase** vielleicht für multiplayer.

Klassendiagramm

Klassendiagramm_v1

ERM

ERM_v1

Pflichtenheft

Welche Softwarevoraussetzungen werden benötigt (mit Versionen)

Systemvoraussetzungen

Betriebssystem
Windows 10 oder höher
Getestet unter: Windows 10.0.26100.4351 (Windows 11 24H2 Build)

Flutter und Dart

Flutter SDK
Version: 3.29.3
Channel: `stable`
Git-Revision: `ea121f8859`
Datum: 11. April 2025

Dart SDK
Version: 3.7.2

Flutter DevTools
Version: 2.42.3

Abhängigkeiten aus pubspec.yaml

Diese Pakete werden im Projekt verwendet und müssen durch `flutter pub get` verfügbar sein:

- `flutter` (SDK)
- `logger: ^2.5.0`
- `supabase_flutter: ^2.0.0`
- `shared_preferences: ^2.5.3`
- `uuid: ^4.0.0`
- `http: ^0.13.6`
- `cupertino_icons: ^1.0.8`
- `flutter_lints: ^5.0.0` (nur für Entwicklung)

Backend

- connexion[swagger-ui] >= 2.6.0; python_version>="3.6"
- connexion[swagger-ui] <= 2.3.0; python_version=="3.5" or python_version=="3.4"
- connexion[swagger-ui] <= 2.14.2; python_version>"3.4"
- werkzeug == 0.16.1; python_version=="3.5" or python_version=="3.4"
- swagger-ui-bundle >= 0.0.2
- python_dateutil >= 2.6.0
  setuptools >= 21.0.0
  Flask == 2.1.1