Realisierung

Delegator

|  |  |
| --- | --- |
| Klassifizierung | intern |
| Status | in Arbeit |
| Programmname | Delegator |
| Projektnummer | 1 |
| Projektleiter | Tabinas Kenan |
| Version | 0.1 |
| Datum | 26. März 2025 |
| Auftraggeber | Tabinas Kenan |
| Autor/Autoren | Tabinas Kenan |
| Verteiler |  |

Änderungsverzeichnis

| Version | Datum | Änderung | Autor |
| --- | --- | --- | --- |
| 0.1 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tabelle 1: Änderungsverzeichnis

Beschreibung

Die Studie beschreibt die angestrebte Lösung, in dem sie die groben Ziele definiert, möglichen Lösungsvarianten aufführt und diese dann bewertet. Sie bildet die Grundlage für die Entscheidung, ob ein Projekt freigegeben wird oder nicht. Sie ist die Voraussetzung für die Erarbeitung des Projektmanagementplans und des Projektauftrags.

# Designpattern in Flutter

Zuerst habe einfach angefangen, weil ich wenig Erfahrung mit Flutter hatte. Ich habe vieles ChatGPT / Claude machen lassen. Das zu einem gewissen Grad funktioniert. Der Chat Teil hat wunderbar funktioniert.

A screenshot of a phone

AI-generated content may be incorrect.A screenshot of a chat

AI-generated content may be incorrect.

## Design Pattern

Im Verlauf hat sich das leider nicht tragbar gezeigt. Beim erstellen von Tasks bin ich an die grenzen gekommen. Die Language Modells konnten mir nicht mehr helfen. Selbst mit Zurgiff auf das Repository, keine Chance. Der Code scheint gut auszusehen, sagten sie. Irgendwelche Lösung Ansätze die nicht weiter brachten.

Ich habe Claude nach Design Patterns gefragt folgende Vorschläge gab es:

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspekt** | **Feature-Based** | **Clean Architecture** | **MVVM** |
| Grundprinzip | Organisation nach Funktionalitäten | Strenge Schichtentrennung mit Abhängigkeitsregeln | Trennung von Ansicht und Ansichtslogik |
| Ordnerstruktur | Nach Features gruppiert, jedes Feature enthält alle relevanten Schichten | Konzentrische Kreise: Domain (innerste), Data, Presentation (äußerste) | Models, Views, ViewModels als Hauptgruppen |
| Stärken | Leichte Navigation innerhalb eines Features, bessere Isolation von Features, gut für parallele Teamarbeit | Sehr klare Abhängigkeitsrichtung, Domain-Schicht völlig unabhängig und Langfristige Wartbarkeit | Intuitive Struktur, einfacher zu erlernen und Gute Testbarkeit der ViewModels |
| Schwächen | Mögliche Codeduplizierung zwischen Features, kann bei gemeinsamen Komponenten unübersichtlich werden | Steilere Lernkurve, Mehr Boilerplate-Code und Überkompliziert für kleinere Apps | Weniger rigide Regeln als Clean Architecture, ViewModels können zu groß werden |
| Skalierbarkeit | Sehr gut | Exzellent | Gut |
| Komplexität | Mittel | Hoch | Mittel |
| Eignung für | Mittelgroße bis große Apps mit klar abgegrenzten Features | Komplexe, langlebige Enterprise-Anwendungen | Apps mit vielen UI-Zuständen |
| State Management | Flexibel (BLoC, Provider, Riverpod, usw.) | Meist Use-Cases und Repositories | Typischerweise Provider oder Riverpod |
| Testbarkeit | Gut | Sehr gut | Gut |
| Code-Isolation | Feature-Isolation | Schichten-Isolation | Zuständigkeits-Isolation |
| Lernkurve | Mittel | Steil | Flach bis mittel |
| Teamarbeit | Sehr gut (Feature-Teams) | Gut (Schichten-Teams) | Gut |
| Änderungen an Geschäftslogik | Auf Features beschränkt | Einfach durch isolierte Domain-Schicht | Auf ViewModels beschränkt |
| Änderungen an UI | Auf Features beschränkt | Auf Presentation-Schicht beschränkt | Nur Views betroffen |
| Wiederverwendbarkeit | Mäßig zwischen Features | Sehr gut (besonders Domain-Schicht) | Gut (besonders ViewModels) |
| Code-Organisation | Nach Business-Funktionen | Nach technischen Schichten | Nach Presentation-Pattern |
| Flutter-Spezifisch | Gut integrierbar | Erfordert mehr Anpassung | Natürlich passend |

## Nutzwertanalyse

Habe anschliessend eine Wertung gemacht. Basierend auf mein Projekt und folgender Wert:

|  |  |
| --- | --- |
| **Wert** | **Bedeutung** |
| 0 | Irrelvant |
| 1 | Schlecht |
| 2 | Mittel |
| 3 | Gut |

Tabelle 2: Wertung

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspekt** | **Feature- Based** | **Clean  Architecture** | **MVVM** |
| Grundprinzip | 0 | 0 | 0 |
| Ordnerstruktur | 0 | 0 | 0 |
| Stärken | 1 | 2 | 3 |
| Schwächen | 2 | 1 | 2 |
| Skalierbarkeit | 3 | 3 | 2 |
| Komplexität | 2 | 1 | 2 |
| Eignung für | 3 | 1 | 2 |
| State Management | 0 | 0 | 0 |
| Testbarkeit | 2 | 3 | 2 |
| Code-Isolation | 0 | 0 | 0 |
| Lernkurve | 2 | 1 | 2 |
| Teamarbeit | 0 | 0 | 0 |
| Änderungen an Geschäftslogik | 0 | 0 | 0 |
| Änderungen an UI | 0 | 0 | 0 |
| Wiederverwendbarkeit | 1 | 3 | 3 |
| Code-Organisation | 0 | 0 | 0 |
| Flutter-Spezifisch | 2 | 1 | 3 |
| **TOTAL** | **18** | **16** | **21** |

Tabelle 3: Pattern Nutzwertanalyse

## Fazit

Nach der Wertung hat klar MVVM gewonnen und so fahre ich fort. Beim ersten Anlauf ist mir Aufgefallen, das der Service, also die Schnittstelle zum Backend am Problematischstens ist. Daher habe ich mich folgenden Plan erstellt.

* Services
* Models
* Views
* Integrationstest

# Testinterface

Damit das weiter Entwicklen und Vorallem Testen einfacher wird habe ich mir fuer eien Struktur mit Interfaces Entschieden damit die Services einfacher zu Testen sind und ich einen Testfirst ansatz verfolgen kann.

Ich habe daher allen Code geloescht und von neu Angefangen:

A screen shot of a computer screen

AI-generated content may be incorrect.

Abbildung 1: Delete all things

# Testen und so

## Projects

### Project Get

A screen shot of a computer

AI-generated content may be incorrect.

### Project Delete & Get By Id

A black rectangular object with white text

AI-generated content may be incorrect.

### Project Delete

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

## Events

## Get Events

A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

## Create and Delete

A black rectangular object with white text

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

## Tasks

### Get

A black screen with white text

AI-generated content may be incorrect.

### Create and Delete

A black rectangular object with white text

AI-generated content may be incorrect.

A black rectangular object with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Messages

### Get

A black rectangular object with a white line

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

### Delete

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

## Invitation

### Org im not a part of

A black rectangular object with a white line

AI-generated content may be incorrect.

Abkürzungen und Glossar

| Abkürzung / Fachwort | Erläuterung |
| --- | --- |
| HERMES | Vorgehensmethodik für Projekte und Programme  HERMES 5 ist ein eCH Standard |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tabelle 4: Abkürzungen und Glossar

Inhaltsverzeichnis

1 Informationsbeschaffung 2

1.1 Konkurenz Analyse 2

2 Ziele 3

2.1 Projektziele 3

2.2 Projektvorgehensziele 5

3 Grobanforderungen 6

4 Grundsatzentscheidungen 7

4.1 Grundsatzentscheidungen Übersicht 7

5 Lösungsvarianten von B1 8

5.1 Variantenübersicht 8

5.2 Variante V1 «Weniger Admin mehr Musik» 8

5.2.1 Kurzbeschreibung 8

5.2.2 Zielegruppe 9

5.2.3 Konkurrenz Analyse 10

5.3 Variante V2 «Familien und Bands» 11

5.3.1 Kurzbeschreibung 11

5.3.2 Zielegruppe 12

5.3.3 Konkurrenz Analyse 13

5.4 Analyse und Bewertung der Varianten 14

5.4.1 Zielerreichung 14

5.5 Variantenwahl 14

6 Lösungsvariantenwahl von B2 15

6.1 Variantenübersicht 15

6.2 Variante V1 «WebOnly> 16

6.2.1 Kurzbeschreibung 16

6.2.2 Vorteile 16

6.2.3 Nachteile 16

6.3 Variante V2 «WebViewApp» 17

6.3.1 Kurzbeschreibung 17

6.3.2 Vorteile 17

6.3.3 Nachteile 17

6.4 Variante V3 «Native App» 18

6.4.1 Kurzbeschreibung 18

6.4.2 Vorteile 18

6.4.3 Nachteile 18

6.5 Variante V4 «Cross Plattform» 19

6.5.1 Kurzbeschreibung 19

6.5.2 Vorteile 19

6.5.3 Nachteile 19

6.6 Bewertungskriterien 20

6.6.1 Kriterien 20

6.6.2 Nicht Kriterien 20

6.7 Analyse und Bewertung der Varianten 21

6.7.1 Bewertung 21

6.7.2 Analyse 21

7 Lösungsvariantenwahl von B3 22

7.1 Ausgangslage 22

7.2 Variantenübersicht 23

7.3 Soll Situation 23

7.4 Variante V1 «On-Premise» 24

7.4.1 Hardware 24

7.4.2 Verantwortung 25

7.4.3 Vorschlag mit SolidRun für mögliches Update 26

7.5 Variante V2 «VPS» 27

7.5.1 Terminologie 27

7.5.2 DigitalOcean 28

Fazit: 28

7.5.3 Hetzner 29

Fazit: 29

7.5.4 Contabo 30

Fazit: 30

7.6 Variante V3 «CaaS» 31

Fazit: 31

7.6.1 Zeitersparrnis 32

7.7 Variantenentscheid 33

7.7.1 Wirtschaftliche Kriterien 33

7.7.2 Nutzerbasis für Breakeven Kurve 34

7.7.3 Nutzerbasis Breakeven Aufstellung 34

7.7.4 Variantenwahl 35

7.7.5 Varianten Wahl B3 35

8 Wirtschaftlichkeit 36

8.1 Basierend auf der Konkurrenz 36

8.2 Mögliches Preismodell 37

8.3 Marketing 38

8.3.1 Instagram 38

8.3.2 Betriebskosten 39

8.4 Zeiten 39

8.5 Kosten nach dem ersten Jahr 40

8.5.1 Wachstum Strategie 41

8.6 Kosten mit 10% Marketing 42

8.6.1 Wachstum Worstcase 43

8.6.2 Wachstum Bestcase 44

8.6.3 Reele Ausgaben 45

Abbildungsverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

Tabellenverzeichnis

Tabelle 1: Änderungsverzeichnis 1

Tabelle 2: Informationsbeschaffung Konkurenz 2

Tabelle 3: Ziele 4

Tabelle 4: Projektvorgehensziele 5

Tabelle 5: Grobanforderungen 6

Tabelle 6: Grundsatz Entscheidungen 7

Tabelle 7: Varientenübersicht von B1 8

Tabelle 8: B1 V1 Zielgruppe 9

Tabelle 9: B1 V1 Konkurenzanalyse 10

Tabelle 10: B1 V2 Zielgruppe 12

Tabelle 11: B1 V2 Konkurenz Analyse 13

Tabelle 12: Zielerreichung B1 14

Tabelle 13: Varientenübersicht von B2 15

Tabelle 14: B2 V1 Vorteile 16

Tabelle 15: B2 V1 Nachteile 16

Tabelle 16: B2 V2 Vorteile 17

Tabelle 17: B2 V2 Nachteile 17

Tabelle 18: B2 V3 Vorteile 18

Tabelle 19: B2 V3 Nachteile 18

Tabelle 20: B2 V4 Vorteile 19

Tabelle 21: B2 V4 Nachteile 19

Tabelle 22: B2 Kriterien 20

Tabelle 23: B2 Nicht Kriterien 20

Tabelle 24: B2 Bewertung 21

Tabelle 25: B3 Variantenübersicht 23

Tabelle 26: B3 V1 Hardware 24

Tabelle 27: B3 V1 Verantwortung 25

Tabelle 28: B3 V1 Inital Kosten 26

Tabelle 29: B3 V1 Fortlaufende Kosten 26

Tabelle 30: B3 V2 Terminologie 27

Tabelle 31: B3 V2 DigitalOcean 28

Tabelle 32: B3 V2 Hetzner 29

Tabelle 33: B3 V2 Contabo 30

Tabelle 34: B3 V3 Caas 31

Tabelle 35: B3 V1 Fortlaufende Zeit Kosten Best Case 32

Tabelle 36: B3 V1 Fortlaufende Zeit Kosten geschätzt 32

Tabelle 37: B3 V3 Nutzerbasis für CaaS nötig 34

Tabelle 38: B3 Variantenwahl Zeitvergleich 35

Tabelle 39: Wirtschaftlichkeit Preisvergleich 36

Tabelle 40: Wirtschaftlichkeit mögliches Preismodell 37

Tabelle 41: Wirtschaftlichkeit Umsatz pro User 37

Tabelle 42: Marketing Werbung auf Instagram 38

Tabelle 43: Marketing Instagram Ad Reach 38

Tabelle 44: Marketing Instagram Ad CPA 38

Tabelle 45: Marketing Werbekosten verglichen mit Einnahmen 39

Tabelle 46: Aufteilung bei 20% Pensum 39

Tabelle 47: Kosten bei der Diplomarbeit 40

Tabelle 48: Kostenaufstellung nach dem ersten Jahr 40

Tabelle 49: Vergleich mit nötigen Usern 40

Tabelle 50: Wachstum Strategie bei 1% 41

Tabelle 51: Kostenaufstellung nach dem ersten Jahr bei 10% 42

Tabelle 52: Instagram Ad Reach bei 10% 42

Tabelle 53: Instagram CPA bei 10% 42

Tabelle 54: Wachstum Strategie bei 10% Worstcase 43

Tabelle 55: Wachstum Strategie bei 10% Bestcase 44

Tabelle 56: Verlustpotential verschiedener Szenarien 45

Tabelle 57: Abkürzungen und Glossar 46