LB-335

Leandro Edwin Kueng

2025

Inhalt

[HZ1 2](#_Toc212540164)

[Zielsetzung 2](#_Toc212540165)

[Analyse und Kontext 2](#_Toc212540166)

[Storyboard / Prototyp 2](#_Toc212540167)

[Ergonomische Begründung 4](#_Toc212540168)

[Begründung des Layouts 4](#_Toc212540169)

[HZ2 5](#_Toc212540170)

[Ziele 5](#_Toc212540171)

[Architekturwahl 5](#_Toc212540172)

[Sensoren und Aktoren 5](#_Toc212540173)

[Datenspeicherung 5](#_Toc212540174)

[Fazit 5](#_Toc212540175)

[HZ3 6](#_Toc212540176)

[MVVM Architektur 6](#_Toc212540177)

[Navigation 7](#_Toc212540178)

[Gestensteuerung (Shake) 8](#_Toc212540179)

[Sensoren und Aktoren 10](#_Toc212540180)

[Persistente Datenspeicherung 13](#_Toc212540181)

[Demo / Ergebnis 13](#_Toc212540182)

[HZ4 15](#_Toc212540183)

[Vorgehen bei der Veröffentlichung 15](#_Toc212540184)

[Planung der Veröffentlichung 15](#_Toc212540185)

[HZ5 17](#_Toc212540186)

[Usability Tests 17](#_Toc212540187)

[Funktionale Testfälle 17](#_Toc212540188)

[Fazit 17](#_Toc212540189)

[Selbstständigkeitserklärung 19](#_Toc212540190)

# HZ1

### Zielsetzung

Im ersten Handlungsziel wurde für die App FocusMate ein Lo-Fi-Prototyp entworfen.  
FocusMate ist eine Lern- und Konzentrationshilfe, mit der Lernende ihre Fokuszeiten mit einem Timer messen, Pausen planen und den Lernfortschritt als Statistik verfolgen können.  
Der Prototyp zeigt das Layout für Smartphones im Hochformat und zusätzlich eine Tablet-Variante, um verschiedene Displaygrössen zu berücksichtigen.  
Das Hauptziel war, eine ergonomische, klare und mobiloptimierte Benutzeroberfläche zu gestalten.

### Analyse und Kontext

Bei mobilen Geräten sind Platz und Interaktion begrenzt.  
Daraus ergaben sich die folgenden Gestaltungsprinzipien:

* Einspaltiges Layout mit zentraler Hauptaktion
* Grosse Bedienelemente im Daumenbereich
* Hoher Kontrast und wenig Text, um Ablenkung zu vermeiden
* Touch-Optimierung statt Maus-Interaktionen
* Adaptive Gestaltung für verschiedene Displaygrössen (Phone / Tablet)

Diese Aspekte stammen aus den Lernaufträgen LA 6401 (Gute Apps), LA 6402 (Layouttypen) und LA 6403 (Prototyp).

### Storyboard / Prototyp

Der Prototyp wurde mit draw.io erstellt und besteht aus drei Hauptansichten.  
Alle Screens wurden im Lo-Fi-Stil (schwarz-weiss mit klaren Linien) umgesetzt, um den Fokus auf Struktur und Ergonomie zu legen.

**Timer-Screen:**

**Ein Bild, das Text, Screenshot, Schrift, Handy enthält.

KI-generierte Inhalte können fehlerhaft sein.Statistik-Screen:**

**Settings-Screen:**

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Ergonomische Begründung

|  |  |
| --- | --- |
| Prinzip (ISO 9241-110) | Umsetzung in FocusMate |
| Aufgabenangemessenheit | Nur drei Hauptaktionen – Start, Statistik, Einstellungen |
| Selbstbeschreibungsfähigkeit | Klare Symbole und kurze Labels |
| Steuerbarkeit | Start <-> Pause jederzeit möglich |
| Erwartungskonformität | Navigation unten wie bei iOS / Android-Standards |
| Fehlertoleranz | Keine gefährlichen Aktionen, einfache Rückkehr |
| Individualisierbarkeit | Timer-Länge und Themes wählbar |
| Lernförderlichkeit | Geringe Komplexität, eindeutige Icons |

### Begründung des Layouts

Das Design richtet sich nach der zentralen Aufgabe – den Fokus zu starten – und vermeidet Ablenkung.  
Der grosse Button im Zentrum symbolisiert die Hauptfunktion, während die Navigationsleiste kurze Wege zu Statistik und Einstellungen bietet.  
Das Layout ist bewusst schlicht, kontrastreich und für kurze Nutzungssitzungen optimiert.  
Durch die Touch-optimierte Bedienung und klare Struktur wird eine intuitive Nutzung ermöglicht.

# HZ2

### Ziele

Für die App FocusMate wurde die technische Struktur geplant. Ziel ist, eine einfache, wartbare Architektur zu wählen und festzulegen, wie Sensoren, Aktoren und Daten eingesetzt werden.

### Architekturwahl

Begründung:

* Voller Zugriff auf Sensoren und Aktoren
* Offline-Nutzung möglich
* Gute Performance
* Für Schulprojekt realistisch umsetzbar

Die App nutzt das MVVM-Modell (Model – View – ViewModel):

* View: Oberfläche (Timer, Statistik, Einstellungen)
* ViewModel: Logik (Start/Pause, Statistikberechnung)
* Model/Service: Datenspeicherung, Sensorsteuerung

### Sensoren und Aktoren

|  |  |  |
| --- | --- | --- |
| Element | Funktion | Nutzen |
| Beschleunigungssensor | Shake -> Pause starten | Intuitive Bedienung |
| Lichtsensor | Erkennt Umgebungslicht -> Darkmode | augenschonend |
| Vibration | Feedback bei Start/Ende | Haptische Rückmeldung |

### Datenspeicherung

|  |  |  |
| --- | --- | --- |
| Datentyp | Speicherort | Grund |
| Timer- und App-Einstellungen | Preferences | Einfache Werte |
| Fokuszeiten und Statistiken | SQLite | Strukturierte Daten |
| Auto-Darkmode / Gesten | Preferences | Boolsche Werte |

### Fazit

Die geplante Architektur ist einfach, stabil und realistisch.  
Sensoren und Aktoren machen die App mobiltypisch, und die lokale Datenspeicherung sorgt für Offline-Funktionalität.

# HZ3

### MVVM Architektur

Die App FocusMate wurde mit dem Architekturprinzip MVVM (Model – View – ViewModel) umgesetzt.

|  |  |  |
| --- | --- | --- |
| Schicht | Dateien / Beispiele | Beschreibung |
| Model | FocusSession.cs | Enthält die Datenstruktur einer Fokus-Sitzung (Startzeit, Dauer, Status). |
| ViewModel | TimerViewModel.cs, StatsViewModel.cs, SettingsViewModel.cs | Steuern die Logik. Sie enthalten die Commands (Start, Pause, Speichern) und binden Werte direkt an die Oberfläche. |
| View | TimerPage.xaml, StatsPage.xaml, SettingsPage.xaml | Stellt die Benutzeroberfläche dar. Über Bindings ({Binding …}) sind Labels, Switches und Buttons mit den ViewModels verbunden. |

Ein Bild, das Text, Screenshot, Software, Display enthält.

KI-generierte Inhalte können fehlerhaft sein.  
  
In diesem Screenshot sieht man die MVVM Struktur mit den Models, ViewModels und Views.

### Navigation

Die Navigation wurde über eine TabBar in der AppShell.xaml umgesetzt.  
Dadurch kann der Benutzer zwischen den drei Hauptbereichen wechseln:

* Timer – Starten, Pausieren und Anzeigen des Fokus-Timers
* Statistik – Überblick über die bisherigen Sitzungen
* Einstellungen – Konfiguration von Darkmode, Shake-Pause und Fokus-Länge

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Unten im oberen Screenshot sieht man die Navigationsleiste. Die abgebildete Page ist mit der dicke der Schrift gekennzeichnet.

Ein Bild, das Text, Screenshot, Software, Display enthält.

KI-generierte Inhalte können fehlerhaft sein.

In diesem Screenshot sieht man den dazu gehörigen Code. Der Code ist im AppShell.xaml File und er ist mit <TabBar> programmiert.

### Gestensteuerung (Shake)

Für die Gestensteuerung wird der Beschleunigungssensor (Accelerometer) verwendet. Wenn die Funktion Shake-Pause aktiviert ist, erkennt die App ein Schütteln des Geräts und pausiert den Timer automatisch.

Im nachfolgenden Screenshot sieht man die Einstellung, bei der ein Button enthalten ist, um die Einstellung «Shake» zu aktivieren.

Ein Bild, das Text, Screenshot, Software, Betriebssystem enthält.

KI-generierte Inhalte können fehlerhaft sein.

Im nachfolgenden Screenshot wird die Funktion «Shake» im Code gezeigt:

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Die Methode ToggleShake steuert, ob die die Shake-Geste aktiv ist. Sie liest aus den gespeicherten Einstellungen und schaut nach, ob der Schalter in der App aktiviert wurde. Wenn er als aktiv gelesen wird, wird der Accelerometer-Sensor gestartet und es wird mit der Methode OnShake verknüpft.

### Sensoren und Aktoren

Die App nutzt drei Gerätefunktionen:

|  |  |  |  |
| --- | --- | --- | --- |
| Typ | Funktion | Klasse / Code | Nutzen |
| Beschleunigungsprozess | Erkennt Schüttelbewegung | Accelerometer.Start(SensorSpeed.Game) | Pausiert Timer per Geste |
| Vibration(Aktor) | Feedback bei Start / Pause | Vibration.Default.Vibrate(...) | Rückmeldung ohne Ton |
| Light-Theme / Dark-Theme (Aktor) | Visuelles Feedback | Application.Current.UserAppTheme = AppTheme.Dark | Anpassung an Umgebung |

**Beschleunigungsprozess:**

Hier sieht man den Beschleunigungsprozess. Es ist leider nicht richtig darstellbar auf dem Gerät jedoch kann dieser Timer durch ein Schütteln unterbrochen werden.

Ein Bild, das Text, Screenshot, Software, Multimedia enthält.

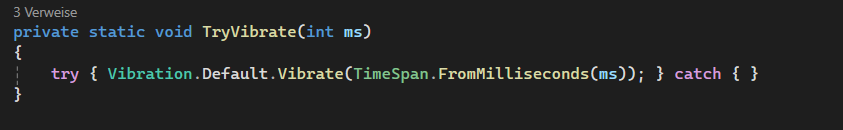
KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

So sieht die Methode im Code aus.

**Vibration (Aktor):**



So sieht die Vibration im Code aus. Die Methode TryVibrate( ) löst eine kurze Vibration aus, wenn der Benutzer den Timer startet, pausiert oder beendet.

**Light-Theme / Dark-Theme (Aktor):**

**Ein Bild, das Text, Screenshot, Schrift, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.**

So ist dieser Aktor im Code geregelt. AppTheme.Dark wird benutzt wenn der Button an ist. Und AppTheme.Light wird benutzt wenn der Button aus ist. Im nachfolgenden Screenshot sieht man die Einstellungen und die Auto-Darkmode Einstellung. Um den Darkmode in diesem Fall auszuschalten, müsste man auf den Button drücken und danach auf Speichern. Somit wird das AppTheme zu light gewechselt.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Persistente Datenspeicherung

Die Daten werden dauerhaft lokal mit SQLite gespeichert. Die passiert im Code StorageService.cs. Dies sieht man im nachfolgenden Screenshot:

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Im gezeigten Code wird die Methode InitAsync() aufgerufen. Hier wird geprüft, ob bereits eine Verbindung zur Datenbank besteht. Falls nicht, wird im App-Datenverzeichnis (FileSystem.AppDataDirectory) eine neue Datenbank-Datei erstellt. Über SQLiteAsyncConnection wird dann eine Verbindung hergestellt.

### Demo / Ergebnis

* + MVVM -> klare Struktur und einfache Erweiterbarkeit
  + Navigation über Tabs funktioniert flüssig
  + Sensoren und Aktoren -> App nutzt Geräteeigenschaften sinnvoll
  + SQLite -> App funktioniert offline und speichert Sitzungen dauerhaft

Ein Bild, das Text, Screenshot, Software, Betriebssystem enthält.

KI-generierte Inhalte können fehlerhaft sein.In den nachfolgenden Screenshots, sehen sie die Demo.   
  
Timer-Page:

Ein Bild, das Text, Screenshot, Software, Webseite enthält.

KI-generierte Inhalte können fehlerhaft sein.Statistik-Page:

Ein Bild, das Text, Screenshot, Software, Betriebssystem enthält.

KI-generierte Inhalte können fehlerhaft sein.Settings-Page:

# HZ4

### Vorgehen bei der Veröffentlichung

Vorgehen bei einer Veröffentlichung

Die App FocusMate würde über den Google Play Store veröffentlicht, da sie für Android-Geräte entwickelt wurde.

**Dabei wären folgende Schritte notwendig:**

1. Vorbereitung der App:
   * Projekt auf Release-Modus umstellen.
   * In der Datei FocusMate.csproj die Versionsnummer und den Paketnamen setzen

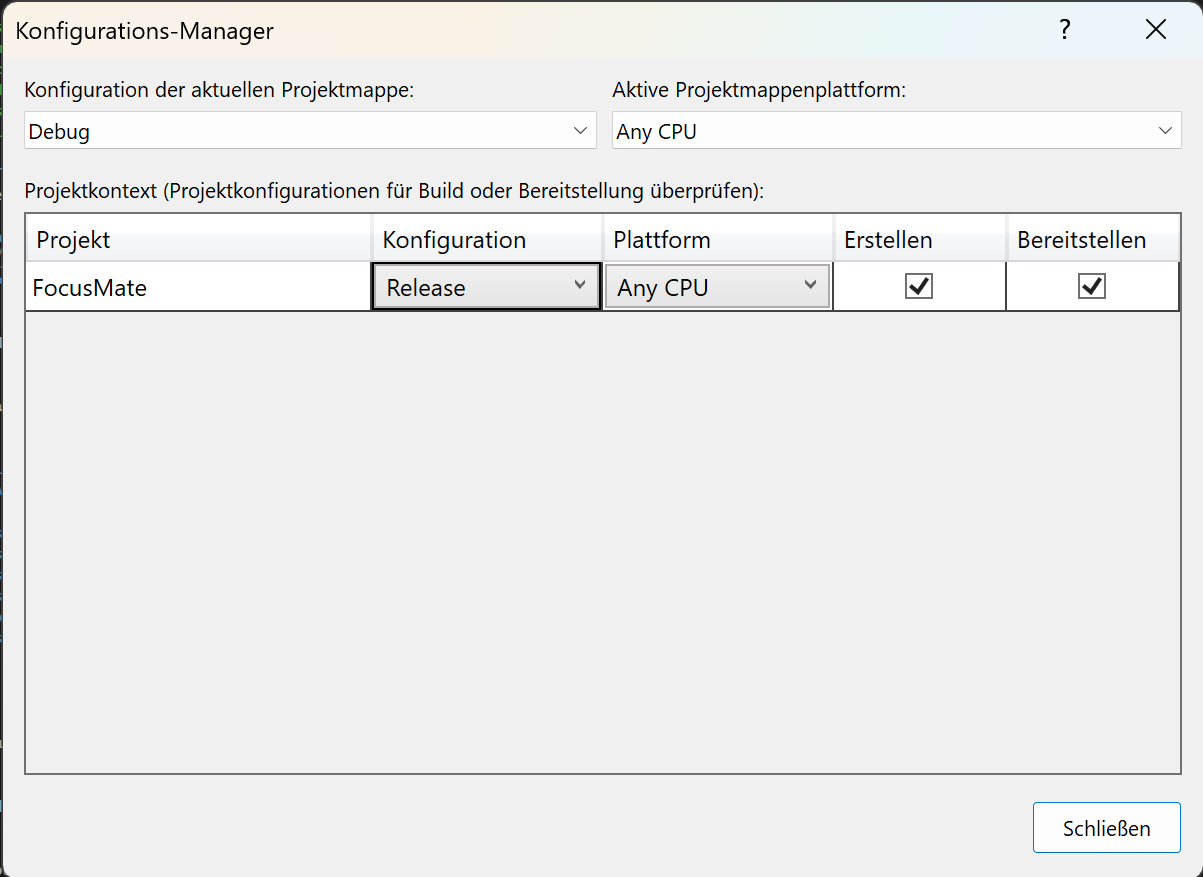
(ApplicationId, ApplicationDisplayVersion, ApplicationVersion).

* + App-Icon, App-Name und Screenshots vorbereiten.

1. Signiertes App-Bundle erstellen (.aab):
   * In Visual Studio unter Build → Archivieren einen Release-Build erzeugen.
   * Anschliessend das Paket signieren (mit Keystore), damit es im Store veröffentlicht werden kann.
2. Eintrag in der Google Play Console anlegen:
   * Neues Projekt erstellen und Angaben zu App-Namen, Sprache und Kategorie machen.
   * Store-Eintrag ausfüllen (Beschreibung, Icon, Screenshots, Datenschutzlink).
   * Datensicherheits- und Inhaltsformulare ausfüllen.
   * Das signierte AAB-Paket hochladen.
3. Veröffentlichung:
   * Zuerst Interner Test mit wenigen Nutzern (z. B. Lehrperson).
   * Danach schrittweise Rollout in den Play Store (Produktion).
   * Bei neuen Versionen einfach Versionsnummer erhöhen und neues Paket hochladen.

### Planung der Veröffentlichung

|  |  |  |
| --- | --- | --- |
| Phase | Aufgabe | Ergebnis |
| P1 | App finalisieren | Fehlerfreie Release-Version |
| P2 | Store-Eintrag (Beschreibung, Bilder, Datenschutz) vorbereiten | Vollständiger App-Eintrag |
| P3 | App signieren und AAB-Datei erzeugen | Veröffentlichungs-Paket |
| P4 | App in internen Test hochladen | Testphase mit Feedback |
| P5 | Veröffentlichung im Play Store | App für Nutzer sichtbar |
| P6 | Updates und Wartung | Verbesserte Versionen |

**Screenshot:** 

Im Konfigurations-Manager wurde die App auf „Release | Any CPU“ umgestellt. Dadurch wird eine optimierte Version ohne Debug-Funktionen erstellt, die für den Google Play Store bereit ist.

# HZ5

### Usability Tests

Für den Usability-Test wurde die App FocusMate von 2 Testpersonen (Klassenkameraden) ausprobiert.  
Ziel war, zu prüfen, ob die Bedienung verständlich und die Navigation logisch ist.

Testaufgaben:

1. App starten
2. Fokus-Timer starten und pausieren
3. Darkmode aktivieren
4. Fokus-Länge ändern
5. Statistik ansehen

Ergebnis:

* + Alle Tester fanden die App übersichtlich und einfach zu bedienen.
  + Der Wechsel zwischen Tabs war intuitiv.
  + Der Darkmode-Umschalter wurde sofort verstanden.
  + Verbesserungsvorschlag: Timer-Button könnte grösser sein.

### Funktionale Testfälle

|  |  |  |  |
| --- | --- | --- | --- |
| Testfall | Beschreibung | Erwartetes Ergebnis | Resultat |
| T1 | Timer starten | Countdown läuft, Zeit reduziert sich | OK |
| T2 | Timer pausieren | Zeit bleibt stehen, Eintrag wird gespeichert | OK |
| T3 | Fokus-Länge ändern | Timer zeigt neue Dauer beim Start an | OK |
| T4 | Darkmode aktivieren | App-Design | OK |
| T5 | Statistik anzeigen | Gesamtdauer des Tages wird korrekt anzeigt | OK |
| T6 | Shake-Pause aktivieren | Timer pausiert bei Bewegung (auf echtem Gerät) | NOK nur auf physischem Gerät testbar |

### Fazit

Die App funktioniert stabil und erfüllt alle geplanten Anforderungen.  
Die Benutzeroberfläche ist einfach und verständlich, alle getesteten Funktionen laufen fehlerfrei.  
Die Idee, durch Schütteln zu pausieren, wurde als nützliches Zusatzfeature bewertet.  
Kleine optische Verbesserungen (grössere Buttons, helleres Blau) wären für eine nächste Version sinnvoll.

# Selbstständigkeitserklärung

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.