Dokumentation

**Personal Finance Manager**

**Hussein Hammade**

**Michael Hurschler**

**Arber Idrizaj**

Inhalt

[1. Einleitung 4](#_Toc199858960)

[💡 Relevanz des Themas 4](#_Toc199858961)

[💻 Ziel der Anwendung 4](#_Toc199858962)

[⚙️ Technische Umsetzung 4](#_Toc199858963)

[🧩 Besonderheiten 5](#_Toc199858964)

[2. Zielsetzung 5](#_Toc199858965)

[🎯 Hauptziel 5](#_Toc199858966)

[🧠 Schwerpunkte der Entwicklung 5](#_Toc199858967)

[🛠️ Erweiterbarkeit 5](#_Toc199858968)

[3. Funktionsübersicht 6](#_Toc199858969)

[4. Technischer Aufbau 7](#_Toc199858970)

[5. Projektstruktur 9](#_Toc199858971)

[6. Datenbankdesign 10](#_Toc199858991)

[7. Objektorientierung 11](#_Toc199858992)

[🔁 Vererbung, Polymorphismus & Abstraktion 11](#_Toc199858993)

[🧩 Einheitliche CRUD-Struktur mit Interfaces 11](#_Toc199858994)

[🛠️ Sauberer und wartbarer Code 11](#_Toc199858995)

[8. Tests 12](#_Toc199858996)

[9. Fazit und Ausblick 13](#_Toc199859004)

# 1. Einleitung

# 💡 Relevanz des Themas

In einer zunehmend digitalisierten Welt ist der Überblick über die eigenen Finanzen wichtiger denn je. Viele Menschen verlieren im Alltag schnell den Überblick über Ausgaben, Einnahmen und Sparziele – besonders ohne ein geeignetes Verwaltungssystem.

# 💻 Ziel der Anwendung

Der Personal Finance Manager (PFM) wurde entwickelt, um genau hier anzusetzen: Die Anwendung bietet eine einfache und strukturierte Möglichkeit, persönliche Finanzen am Computer zu verwalten. Nutzer:innen können ihre Ausgaben erfassen, Budgets setzen und mit Hilfe von Diagrammen ihre finanzielle Lage besser verstehen.

# ⚙️ Technische Umsetzung

Die Software wurde vollständig in C# mit WPF entwickelt und nutzt SQLite zur lokalen Datenspeicherung. Alle Daten bleiben auf dem Gerät des Nutzers, was Sicherheit und Unabhängigkeit garantiert.

# 🧩 Besonderheiten

Neben der klassischen Ausgabenverwaltung ermöglicht der PFM auch die grafische Auswertung, die Benutzeranmeldung, Kategorisierung von Transaktionen sowie eine einfache Bedienung – ideal für den privaten Einsatz.

# 2. Zielsetzung

# 🎯 Hauptziel

Ziel des Projekts war es, eine anwenderfreundliche, robuste und erweiterbare Software zu entwickeln, die sowohl grundlegende als auch fortgeschrittene Funktionen zur Finanzverwaltung umfasst.

# 🧠 Schwerpunkte der Entwicklung

Neben der technischen Umsetzung standen Aspekte wie Benutzerfreundlichkeit, Datensicherheit und eine saubere, wartbare Code-Struktur im Vordergrund.

# 🛠️ Erweiterbarkeit

Die Anwendung ist so aufgebaut, dass neue Funktionen wie z. B. Cloud-Synchronisation, mobile Anbindung oder automatische Kategorisierung von Transaktionen problemlos ergänzt werden könnten.

# 3. Funktionsübersicht

Die Anwendung umfasst folgende Hauptfunktionen:

* Benutzerregistrierung und Login mit Validierung
* Erfassung von Einnahmen und Ausgaben
* Budgetplanung für verschiedene Kategorien
* Anzeige des aktuellen Kontostandes (Saldo)
* Visuelle Darstellung von Statistiken und Ausgabenverteilung
* Bearbeiten und Löschen von Transaktionen
* Validierung und Fehlerbehandlung bei Benutzereingaben
* Speicherung aller Daten lokal mittels SQLite

# 4. Technischer Aufbau

Die Anwendung wurde mit dem Windows Presentation Foundation (WPF) Framework und der Programmiersprache C# entwickelt. Für die Gestaltung der Benutzeroberfläche kam XAML zum Einsatz, wodurch eine klare Trennung von Layout und Logik möglich ist. Die Benutzeroberfläche ist modern, übersichtlich und auf eine intuitive Bedienung ausgelegt.

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

KI-generierte Inhalte können fehlerhaft sein.

Die dahinterliegende Logik ist in den zugehörigen Codebehind-Dateien implementiert, welche direkt mit den Oberflächen verknüpft sind. Dadurch ergibt sich eine saubere Struktur, die eine einfache Erweiterung und Pflege der Anwendung ermöglicht. Die Projektstruktur wurde bewusst modular aufgebaut: Datenbankzugriffe und Datenmanipulationen sind in getrennten Klassen organisiert, was den Code übersichtlich hält und die Wiederverwendbarkeit erhöht.

# 5. Projektstruktur

# Das Projekt besteht aus mehreren strukturiert aufgebauten Dateien, die jeweils einen klar definierten Aufgabenbereich haben. Nachfolgend sind die zentralen Komponenten aufgeführt:

|  |  |
| --- | --- |
| Datei | Funktion |
| MainWindow.xaml | Startansicht der Anwendung mit zentraler Navigation |
| account.xaml | Anzeige der Benutzerinformationen und Kontodaten |
| register.xaml | Maske zur Benutzerregistrierung inkl. Validierung |
| addtransaction.xaml | Oberfläche zum Hinzufügen neuer Einnahmen oder Ausgaben |
| budget.xaml | Verwaltung und Übersicht der gesetzten Budgets |
| Userdb.cs | Backend-Logik für Benutzer: Registrierung, Login und Datenspeicherung |
| Balancedb.cs | Berechnung und Darstellung des aktuellen Kontostandes |
| transaciondb.cs | Verwaltung aller Transaktionen |

# 6. Datenbankdesign

Die Anwendung verwendet eine lokale SQLite-Datenbank, um alle relevanten Finanzdaten dauerhaft zu speichern. Diese Datenbank wird beim ersten Start der Anwendung automatisch erstellt, falls sie noch nicht existiert. Dadurch entfällt eine manuelle Konfiguration und die Anwendung ist sofort einsatzbereit.

* In der Datenbank befinden sich strukturierte Tabellen für:
* Benutzer (Login-Daten, Registrierung)
* Transaktionen (Einnahmen & Ausgaben)
* Budgets (Kategorien, Limits)

Für die Verwaltung der Datenbank werden SQL-Skripte verwendet, die beim Start ausgeführt werden. Diese Skripte prüfen, ob die Datenbankstruktur aktuell ist, und führen ggf. Initialisierung, Datenbank-Updates oder Beispieldaten-Insertions durch.

Die Verbindung zur Datenbank erfolgt mithilfe eines Singleton-Patterns, das sicherstellt, dass während der gesamten Laufzeit der Anwendung nur eine einzige Instanz der Datenbankverbindung aktiv ist. Dies erhöht die Effizienz und verhindert Verbindungsfehler durch Mehrfachzugriffe.

# 7. Objektorientierung

# 🔁 Vererbung, Polymorphismus & Abstraktion

Das Projekt folgt konsequent den Prinzipien der objektorientierten Programmierung. Zentrale Konzepte wie Vererbung, Polymorphismus und Abstraktion wurden in den Klassen gezielt eingesetzt, um Code zu strukturieren, Gemeinsamkeiten zu nutzen und Wiederverwendbarkeit zu fördern.

# 🧩 Einheitliche CRUD-Struktur mit Interfaces

Für alle Datenbankzugriffe (Create, Read, Update, Delete) kommen Interfaces zum Einsatz. Diese abstrahieren die Datenzugriffslogik und ermöglichen es, unabhängig vom konkreten Datenobjekt (z. B. Transaktion, Benutzer) eine einheitliche Struktur zu verwenden. Das erleichtert sowohl das Testen als auch die spätere Erweiterung der Anwendung.

# 🛠️ Sauberer und wartbarer Code

Dank der objektorientierten Struktur ist der Code modular aufgebaut. Neue Funktionen können einfach ergänzt oder bestehende Komponenten angepasst werden, ohne dass andere Teile der Software unbeabsichtigt beeinflusst werden. Das sorgt für eine langfristig wartbare und erweiterbare Codebasis.

# 8. Tests

# Ein zentraler Bestandteil der Softwareentwicklung ist die Sicherstellung der Funktionalität und Stabilität einer Anwendung – insbesondere bei zukünftigen Erweiterungen oder Änderungen im Code. Aus diesem Grund wurden im Projekt gezielt Unit-Tests für wichtige Klassen und Methoden implementiert.

# Die Unit-Tests konzentrieren sich auf jene Komponenten, die für die Kernlogik der Anwendung entscheidend sind. Dazu gehören unter anderem:

# die Validierung von Benutzereingaben (z. B. negative Beträge, ungültige Datumswerte)

# die Berechnung und Überwachung von Budgets

# die Datenbankverbindung sowie die korrekte Ausführung von SQL-Befehlen

# Getestet wurden sowohl Einzelfunktionen als auch die Reaktion auf fehlerhafte Eingaben, um die Anwendung möglichst robust gegenüber unerwartetem Verhalten zu machen.

# Die Verwendung von Unit-Tests bringt mehrere Vorteile mit sich: Einerseits wird die Zuverlässigkeit der Anwendung erhöht, andererseits lassen sich zukünftige Änderungen am Code besser absichern. Entwickler erhalten sofortiges Feedback darüber, ob bestehende Funktionen durch eine Codeänderung unbeabsichtigt beeinträchtigt wurden – ein wesentlicher Aspekt für eine nachhaltige Codequalität.

# 9. Fazit und Ausblick

Mit dem Personal Finance Manager wurde eine umfassende und moderne Anwendung geschaffen, die es Nutzerinnen und Nutzern ermöglicht, ihre persönlichen Finanzen einfach, übersichtlich und effizient zu verwalten. Die Software verbindet eine klare Benutzeroberfläche mit einer stabilen technischen Basis, wodurch ein zuverlässiges und gleichzeitig benutzerfreundliches System entstanden ist. Die Anwendung erfüllt alle grundlegenden Anforderungen an eine Finanzverwaltungssoftware und bietet darüber hinaus sinnvolle Erweiterungen wie Budgetfunktionen und grafische Auswertungen, die das Verständnis der eigenen finanziellen Situation deutlich verbessern. Durch den Einsatz objektorientierter Programmierung, sauber strukturierter Datenbankanbindung und modularem Aufbau ist die Anwendung nicht nur funktional, sondern auch langfristig wartbar und erweiterbar. Die klare Trennung von Oberfläche, Logik und Datenhaltung erleichtert zukünftige Anpassungen und neue Funktionen können gezielt integriert werden. Perspektivisch wäre es denkbar, das System durch Funktionen wie automatischen Datenexport, eine Cloud-Synchronisation zur geräteübergreifenden Nutzung oder sogar eine Anbindung an mobile Endgeräte zu erweitern. Damit bietet der Personal Finance Manager eine solide Grundlage für eine kontinuierliche Weiterentwicklung im Sinne einer noch umfassenderen digitalen Finanzverwaltung.