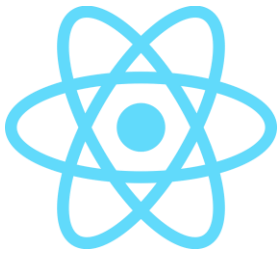
PROJEKT-DOKUMENTATION

Projekttitel: Budget Tracking

Projektverantwortlicher: Janik Preisig

Start Datum: 05.12.2025

End Datum: 12.12.2025



Budget Tracking

Inhaltsverzeichnis

1 Projektstart

1.1 Projektidee

Titel: "Budget Tracker" – Intuitives Budget-Tracking für den Alltag

Jeder kennt das Gefühl: Das Geld ist am Ende des Monats weg, aber niemand weiß, wohin es geflossen ist. Das Problem ist die mangelnde Transparenz über tägliche Ausgaben und Einnahmen.

Mein Projekt, der **" Budget Tracker "**, löst dieses Problem, indem es eine **einfache, schnelle und persistente** Möglichkeit bietet, alle Transaktionen zu erfassen. Die React-Frontend-Applikation nutzt eine REST-API, um **Einnahmen** und **Ausgaben** kategorisiert und mit einer Beschreibung zu speichern.

Die Kernfunktion: Mit wenigen Klicks kann der Benutzer den **aktuellen Kontostand** sehen und sofort erkennen, welche Kategorie (z.B. "Lebensmittel", "Freizeit") das Budget am stärksten belastet.

Nutzen: Anstatt langwieriger Tabellenkalkulationen bietet der "Finanz-Pilot" sofortige Übersicht und hilft dem Benutzer, die Kontrolle über seine Finanzen zurückzugewinnen und bessere Entscheidungen zu treffen. Das visuelle Design ist dabei auf klare Funktionalität optimiert.

2 Anforderungskatalog

2.1 Technische Anforderungen

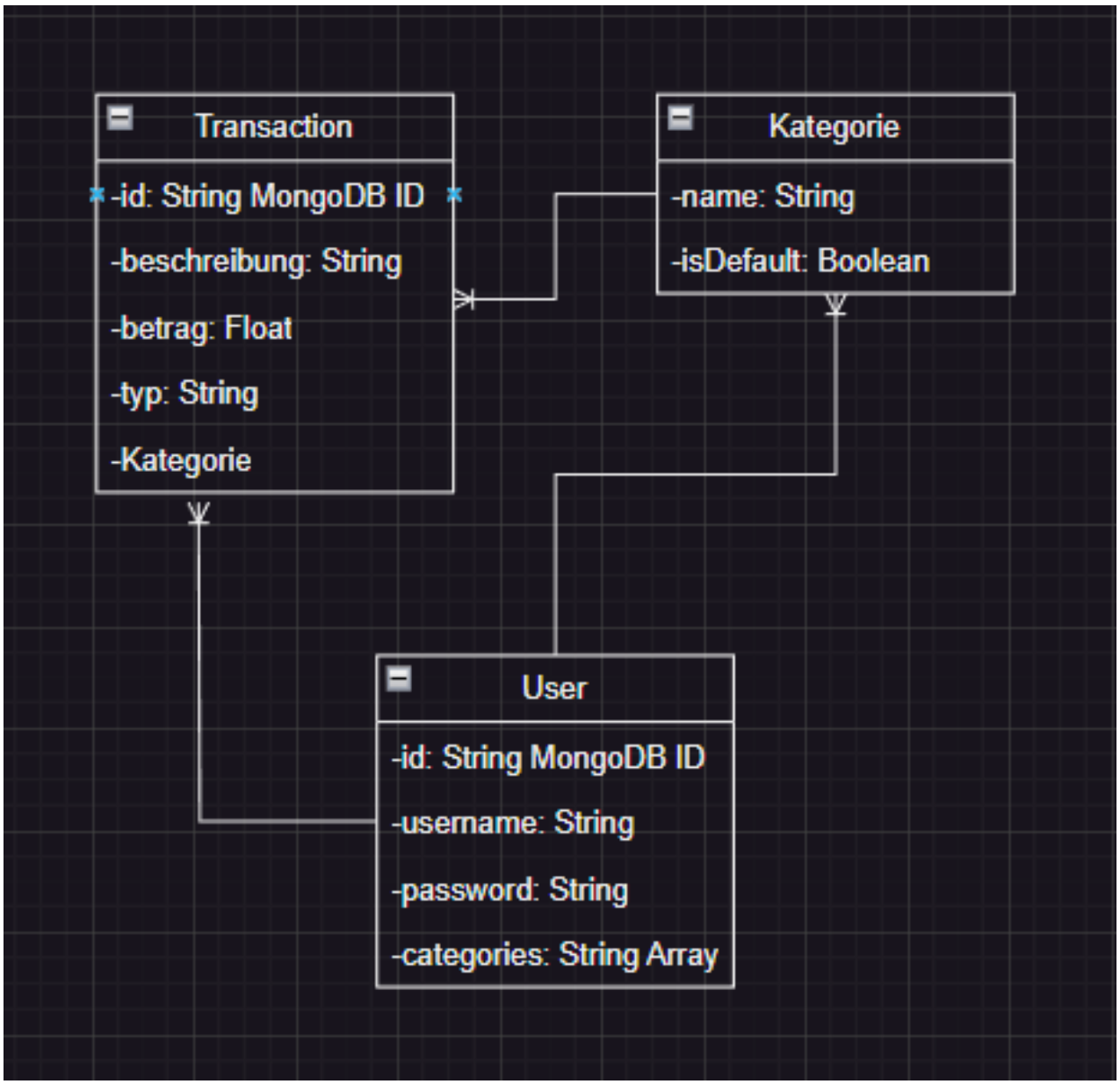
- **A-F2:** Eingabe neuer Transaktionen (Betrag, Kategorie, Beschreibung, Einnahme oder Ausgabe).
- **A-F3:** Verwaltung von Kategorien (Erstellen, Bearbeiten, Löschen).
- **A-F4:** Übersicht über alle Transaktionen (Filterbar nach Datum/Kategorie).
- **A-F5:** Dashboard mit Saldo und grafischer Darstellung der Einnahmen/Ausgaben pro Kategorie.

2.2 Anforderungstabelle

Einnahmen Ausgaben	Der User kann eine Einnahme dann wird Geld Hinzugefügt oder eine Ausgabe dann wird Geld abgezogen.
Aktueller Saldo	Alle Erfassten Transaktionen sollen zusammengerechnet werden.
Kategorie	**Fixkosten** wie Miete Versicherrungen und Abos **Variable Kosten** wie Essen, Pflege und Kleider **Freizeit & Luxus** Hobbys, Urlaub und Reisen, Kultur und Bildung **Mobilität & Verkehr** Benzin/Strom Reparaturen, Steuern, Maut und Parkhäuser **Sparen & Sonstiges** Ungeplante Einmalige Ausgaben oder Beträge auf die Seite legen Die Kategorien sind vom Benutzer definierbar und werden in einer strukturierten Übersicht gespeichert. Für die Deaktivierung oder Entfernung eines Eintrags steht der dedizierte Lösch-Button (rot gekennzeichnet) zur Verfügung.
Gehalt Hinzufügen	Es gibt ein Auswahl Knopf, der entweder Geld vom Konto abzieht (Ausgabe) oder Geld hinzufügt (Einnahme).
Beschreibung	Es gibt ein Textfeld, in die man eine Beschreibung zur Transaktion schreiben kann.
Löschen der Transaktion	Es gibt ein Roter Knopf auf dem «Löschen» steht wenn man den drückt wird die Transaktion gelöscht
Transaktion Speichern	Nachdem man die Beschreibung, der Betrag, Einnahme oder Ausgabe und die Kategorie gewählt hat kann man die Transaktion Speicher mit einem grünen Knopf der neben der Kategorie ist.

3 Klassendiagramm

3.1 UML



3.2 UML Erklärung

Das Diagramm zeigt drei Hauptentitäten (Transaction, Kategorie, User) und ihre Beziehungen. 1 User hat mehrere Kategorien und Transaktionen 1 Kategorie hat mehrere Transaktionen.

Transaktion:

- Die Transaktion hat eine ID die nicht im UI angezeigt wird, diese ist einzigartig.
- Die Beschreibung ist ein String und man kann beliebige Sachen hineinschreiben.
- Der Betrag ist ein Float er zeigt wie viel du ein oder auszahlst.
- Typ ist ein String und ist für, Einzahlen und Auszahlen.
- Kategorie ist ein String und zeigt an für was man das Geld einzahlt oder ausgibt.

Kategorie:

- name ist ein String und ist der Name der Kategorie
- isDefault ist wahr, wenn es eine System-Standardkategorie ist.

User:

- id ist eine String MongoDB ID Eindeutiger Primärschlüssel des Benutzers.
- username ist ein Eindeutiger Login-Name.
- password ist ein Eindeutiges Passwort für den Benutzer
- categories ist ein String Array also ein Liste von Kategorien.

4 Storyboard

4.1 Ablaufplan 1: Transaktion erfassen und Saldo prüfen (Kernfunktion)

Dieser Ablaufplan zeigt den direkten Weg des Benutzers von der Eingabe einer neuen Ausgabe bis zur sofortigen Aktualisierung der Gesamtübersicht.

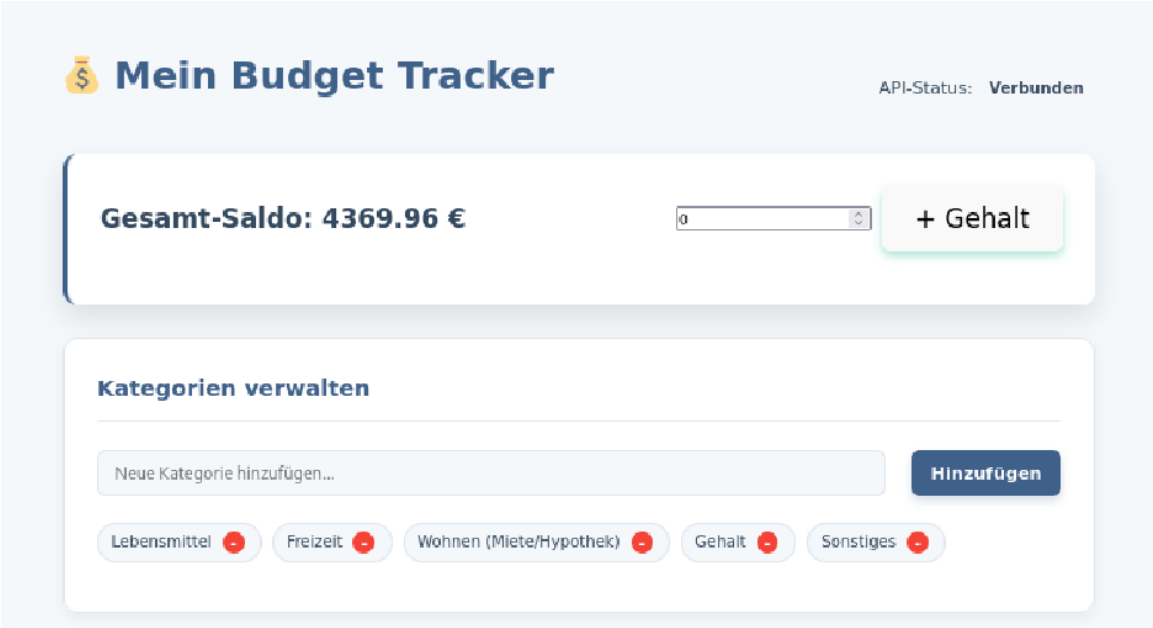
Schritt	Akteur	Aktion / Interaktion	Systemreaktion
1.	**Benutzer**	Öffnet die App / Navigiert zur Startseite.	Die App ruft Transaktionen von der API ab und zeigt den **aktuellen Gesamtsaldo** sowie die **Transaktionsliste** an.
2.	**Benutzer**	Füllt das Formular aus: Beschreibung, Betrag, Typ wählt eine **Kategorie** (Dropdown).	Das Formular validiert die Eingaben (z.B. Betrag > 0).
3.	**Benutzer**	Klickt auf den Button **"Transaktion Speichern"**.	Die App sendet die Daten als POST-Request an die REST-API.
4.	**System / API**	(Interner Prozess) Speichert die neue Transaktion persistent in der Datenbank.	API sendet eine Erfolgsbestätigung zurück.
5.	**System / Frontend**	Aktualisiert den lokalen Anwendungs-State (transactions).	Der neue **Gesamtsaldo** wird sofort neu berechnet und visuell im balance-box aktualisiert.
6.	**Benutzer**	Prüft die Liste der Transaktionen.	Die neue Transaktion wird **unten/oben** in der TransactionList angezeigt.

4.2 Ablaufplan 2: Kategorieverwaltung (Hinzufügen und Löschen)

Schritt	Akteur	Aktion / Interaktion	Systemreaktion
1.	**Benutzer**	Navigiert zur Sektion "Kategorien verwalten".	Die App zeigt die aktuelle Liste der Kategorien (z.B. in Tags mit Lösch-Buttons) an.
2a.	**Benutzer**	(Hinzufügen): Gibt eine neue Kategorie in das Textfeld ein und klickt "Hinzufügen".	Das System prüft auf Duplikate. Bei Erfolg wird der categories-State aktualisiert und die neue Kategorie erscheint in der Liste und im Dropdown der TransactionForm.
2b.	**Benutzer**	(Löschen): Klickt auf den roten Lösch-Button (×) neben einer Kategorie.	Systemprüfung: Die App prüft, ob die zu löschende Kategorie in der transactions-Liste noch verwendet wird.
3.	**System**	(Fall A: In Gebrauch)	Es erscheint eine Warnmeldung: "Kategorie kann nicht gelöscht werden, da sie noch verwendet wird." (Löschvorgang wird abgebrochen).
4.	**System**	(Fall B: Nicht in Gebrauch)	Es erscheint ein Bestätigungsdialog. Nach Bestätigung wird die Kategorie aus dem categories-State entfernt.
5.	**Benutzer**	Kehrt zur Transaktionserfassung zurück.	Die gelöschte Kategorie ist nicht mehr im Dropdown-Menü der TransactionForm sichtbar.

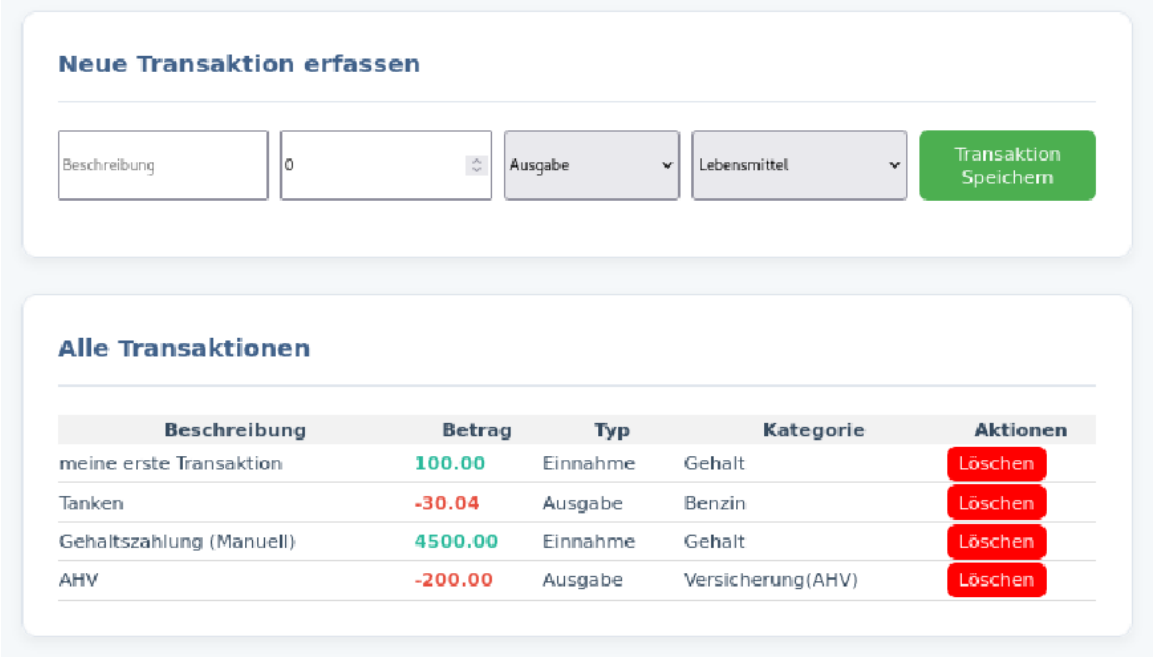
5 Screen-Mockups

5.1 Kategorien und Gehalt verwalten



Der Benutzer kann seinen aktuellen Gesamt-Saldo von 4369.96 € auf einen Blick überprüfen. Er hat die Möglichkeit, eine neue Einnahme wie das Gehalt schnell zu erfassen, indem er einen Betrag eingibt und den entsprechenden Button klickt. Im Bereich der Kategorienverwaltung kann er bestehende Kategorien wie "Lebensmittel" oder "Miete/Hypothek" über das rote Symbol entfernen. Außerdem kann er eine neue Kategorie erstellen, indem er den gewünschten Namen in das Textfeld eingibt. Durch das Betätigen des "Hinzufügen"-Buttons wird diese neue Kategorie sofort in die Liste übernommen. Somit dient das Dashboard der Pflege des Kontostands und der Anpassung der Budget-Struktur.

5.2 Transaktionen Erstellen und Löschen



Der Benutzer kann in diesem Bereich **neue Transaktionen** erfassen, indem er alle Details in die vorgesehenen Felder eingibt. Er muss zunächst eine **Beschreibung** festlegen und den genauen **Betrag** eingeben. Anschließend wählt der Nutzer über Dropdown-Menüs aus, ob es sich um eine **Einnahme** oder eine **Ausgabe** handelt. Zudem muss die Buchung einer passenden **Kategorie** zugeordnet werden, wie beispielsweise "Lebensmittel". Alle eingegebenen Informationen werden durch Klicken auf **"Transaktion Speichern"** der Historie hinzugefügt. Im unteren Bereich sieht der Benutzer eine **vollständige Liste** aller bisherigen Buchungen, inklusive Betrag, Typ und Kategorie. Über den roten **"Löschen"**-Button in der Spalte "Aktionen" kann jede einzelne Transaktion bei Bedarf wieder entfernt werden.

6 REST Schnittstellen

6.1 Dokumentation der REST-Schnittstellen

1. Aufbau der Datentypen und der Kommunikation

Um die Persistenz der Transaktionsdaten zu gewährleisten, kommuniziere ich mit dem Backend über eine schlanke REST-Schnittstelle. Die Basis-URL für alle Transaktionsoperationen ist http://localhost:8080/transactions/documents. Die Kommunikation basiert auf standardisierten HTTP-Methoden und dem Austausch von JSON-Objekten.

Ich muss bei der Kommunikation eine Besonderheit der Backend-Architektur berücksichtigen: Die eigentlichen Geschäftsdaten werden in einem *generischen Dokumenten-Objekt* verpackt.

Das eigentliche Geschäftsdatenobjekt (Transaction Content) umfasst alle für eine Transaktion notwendigen Felder: Die Beschreibung der Ausgabe oder Einnahme, den Betrag (als Gleitkommazahl), den Typ (entweder 'Einnahme' oder 'Ausgabe'), die zugehörige Kategorie und das Datum der Transaktion im ISO-Format.

Das API-Transportobjekt (GenericDocument) dient als Container für diese Geschäftsdaten. Dieses Objekt besteht aus zwei Schlüsselfeldern: der ID, die vom Server als eindeutiger Primärschlüssel generiert wird, und dem Content-Feld, das das komplette Transaction Content-Objekt enthält. Dieses Verpacken in das Content-Feld ist für meine POST-Anfragen zwingend erforderlich und muss bei der Verarbeitung der GET-Antworten wieder entpackt werden.

2. Dokumentierte Endpunkte und ihre Prozesse

Ich nutze drei Kern-Endpunkte, um die notwendigen CRUD-Operationen (Create, Read, Delete) für die Transaktionen durchzuführen.

Abrufen aller Transaktionen (Read-Operation)

Für das Laden und die Initialisierung des Frontend-States sende ich eine GET-Anfrage an den Endpunkt /transactions/documents. Mein Frontend-Code ist darauf ausgelegt, das Array von GenericDocument-Objekten, dass ich vom Server erhalte, zu entpacken. Ich

extrahiere die id und den Content, um die Transaktionsliste im React-State korrekt abzubilden.

Neue Transaktion erstellen (Create-Operation)

Um eine neue Einnahme oder Ausgabe persistent zu speichern, sende ich eine POST-Anfrage an den Endpunkt /transactions/documents. Hierbei ist das korrekte Verpacken der Daten essenziell: Ich nehme die vom Benutzer im Formular erfassten Transaction Content-Daten und bette sie in das Content-Feld des GenericDocument-Objekts ein. Der Server antwortet mir mit dem neu erstellten Dokument inklusive der generierten id. Diese id ist wichtig, da ich sie für spätere Lösch Operationen im Frontend-State benötige.

Transaktion löschen (Delete-Operation)

Zum Entfernen eines Eintrags sende ich eine DELETE-Anfrage an den Endpunkt /transactions/documents/{id}. Ich Hänge die eindeutige ID der zu löschende Transaktion als Pfadparameter an die URL an. Bei erfolgreicher Durchführung der Löschung antwortet der Server mit einem HTTP-Statuscode 200 (OK). Anschliessend entferne ich die Transaktion auch aus meinem lokalen React-State.

7 Testplan

7.1 Testplan

ID	Testfall	Schritte zur Durchführung	Erwartetes Ergebnis	Priorität
T-E2E-01	Erfassung Einnahme	1. App starten. 2. Transaktion eingeben: Beschreibung="Mieteinnahme", Betrag=1000, Typ="Einnahme". 3. Speichern.	Saldo zeigt **1000.00 €** in grüner Farbe. Transaktion erscheint in der Liste.	Hoch
T-E2E-02	Erfassung Ausgabe	1. Aufbauend auf T-E2E-01. 2. Transaktion eingeben: Beschreibung="Miete", Betrag=800, Typ="Ausgabe". 3. Speichern.	Saldo zeigt **200.00 €** (1000 - 800) in grüner Farbe. Transaktion erscheint in der Liste.	Hoch
T-E2E-03	Negativer Saldo	1. Aufbauend auf T-E2E-02. 2. Transaktion eingeben: Beschreibung="Laptop", Betrag=300, Typ="Ausgabe". 3. Speichern.	Saldo zeigt **-100.00 €** in roter Farbe.	Hoch
T-E2E-04	Löschfunktion	1. Aufbauend auf T-E2E-03. 2. Lösch-Button neben "Mieteinnahme" klicken. 3. Bestätigen.	"Mieteinnahme" verschwindet aus der Liste. Saldo aktualisiert sich auf **-1100.00 €** (-100 - 1000).	Hoch
T-E2E-05	Gehalt hinzufügen	1. Saldo auf 0 setzen (alle Transaktionen löschen). 2. Im Gehalts-Input 2500 eingeben. 3. Auf "+ Gehalt" klicken.	Saldo zeigt **2500.00 €** in grüner Farbe. Eine neue Einnahme mit Beschreibung="Gehaltszahlung (Manuell)" erscheint in der Liste.	Hoch
T-E2E-06	API-Fehleranzeige	1. **Spring Boot API stoppen**. 2. App neuladen.	Das Feld "API-Status:" zeigt **FEHLER**. Es erscheint die rote Fehlermeldung unterhalb der Liste.	Mittel

7.2 Testprotokoll

Kennzahl	Wert
Gesamtzahl Testfälle	6
Bestanden (Passed)	6
Fehlgeschlagen (Failed)	0
Blocker-Fehler (Schweregrad)	0
Pass-Rate	100%
Fazit	Die Kernfunktionalität der Anwendung ist **stabil** und erfüllt die Geschäftsanforderungen.

Die Kernfunktionalität von meinen Budget Tracker Funktioniert einwandfrei

Alle zentralen Geschäftslogiken – das Hinzufügen, Löschen und die korrekte, farbliche Darstellung des Gesamtsaldos (positiv/negativ) – funktionieren wie erwartet und sind stabil.

Somit ist die Anwendung in Bezug auf die Transaktionsverwaltung und die neue Gehaltsfunktion stabil und freigabereif.

8 User Stories

8.1 User Stories

Als Benutzer möchte ich meinen aktuellen Gesamt-Saldo sofort sehen können, um schnell meinen finanziellen Überblick zu erhalten und zu wissen, wie viel Geld verfügbar ist.

Als Benutzer möchte ich eine neue Einnahme (z.B. Gehalt) schnell über das Dashboard hinzufügen können, um meinen Saldo unkompliziert zu aktualisieren.

Als Benutzer möchte ich eine neue Ausgabentransaktion mit Beschreibung, Betrag und Typ eingeben können, um jede finanzielle Bewegung detailliert zu dokumentieren.

Als Benutzer möchte ich jede Transaktion einer bestehenden Kategorie (z.B. Lebensmittel oder Benzin) zuordnen können, um später Auswertungen nach Verwendungszweck zu erstellen.

Als Benutzer möchte ich neue, individuelle Kategorien (z.B. "Urlaub") erstellen und nicht benötigte Kategorien löschen können, um die Budgetstruktur an meine persönlichen Bedürfnisse anzupassen

Als Benutzer möchte ich alle meine Transaktionen in einer Liste einsehen können, um die Historie meiner Finanzen zu prüfen und fehlerhafte Buchungen bei Bedarf zu löschen.

9 Installationsanleitung

Die ZIP datei vom Github runterladen Extrahieren ins Verzeichnis vom Projekt die Api ist die vom Teams also da einfach docker compose up -d und dann läuft alles.

10 Hilfestellung

Vorlage Projektdokumentation von
https://www.timo24.de/lexikon/projektdokumentation-vorlage/ | Chat | Microsoft Teams

Coden hat mir Gemini und ChatGPT geholfen