Wyższa Szkoła Technologii Informatycznych w Katowicach

System Zarządzania Budżetem Domowym

Kierunek	Informatyka
Rok akademicki	2025/2026
Przedmiot	Projekt Systemu Informatycznego
Semestr	Piąty
Grupa	5ION1

Dane wykonującego	
Imię i nazwisko	Jakub Reczko
Nr albumu	09224
Dane prowadzącego	
	mgr Jacek Żywczok
Data oddania	
Data zaliczenia	

Spis treści

Spis treści **1**
Wprowadzenie do tematyki projektu **2**
Zamierzony cel projektu **3**
Cel główny 3
Cele szczegółowe 3
Wstępne założenia i uwarunkowania **4**
Założone ograniczenia i możliwość ewaluacji projektu **5**
Ograniczenia techniczne 5
Ograniczenia organizacyjne i czasowe 6
Chronologiczny plan pracy **7**
Wymagania funkcjonalne **8**
Obsługa transakcji 8
Obsługa kategorii 9
Podstawowa analiza finansowa 10
Filtrowanie danych 11
Przechowywanie danych **11**
Interfejs użytkownika 11
Wymagania niefunkcjonalne **12**
Wymagania sprzętowe 12
Wymagania systemowe 12
Wymagania organizacyjne **13**
Wymagania dotyczące danych **14**
Tabela: Category 14
Tabela: Transaction 14
Tabela: Settings 15
Relacje między tabelami 16
Uzasadnienie wyboru struktur danych 16
Diagram bazy danych 17
Metody pracy, narzędzia i techniki **18**
Technologie 18
Diagram Przypadków Użycia 18
Diagram Klas 18
Wireframes 18
Opis Struktur 21
Spis Metod 22
Wykaz Algorytmów 22
Bibliografia **23**
Spis Rysunków **24**
Spis Tabel **25**

Wprowadzenie do tematyki projektu

Zarządzanie budżetem domowym jest istotnym elementem codziennego życia.

Wielu użytkowników poszukuje prostych narzędzi, które pozwalają kontrolować wydatki, analizować wpływy oraz planować oszczędności w sposób intuicyjny i przejrzysty.

Dostępne na rynku aplikacje do planowania budżetu często oferują rozbudowane funkcje, które mogą przytłaczać użytkowników szukających prostoty. Wiele z nich wymaga również połączenia z Internetem, tworzenia kont lub udostępniania danych finansowych firmom trzecim, co budzi obawy o prywatność i bezpieczeństwo. Z tego powodu pojawia się potrzeba stworzenia lekkiego, lokalnego rozwiązania, które pozwoli użytkownikowi w pełni kontrolować własne dane finansowe.

Celem projektu jest stworzenie prostego programu komputerowego, który umożliwi:

- rejestrowanie transakcji finansowych,
- klasyfikowanie ich według kategorii,
- analizowanie przepływów finansowych w wybranych okresach,
- prezentację danych w formie zestawień i wykresów.

System ma na celu nie tylko ułatwienie zarządzania codziennymi wydatkami, lecz także kształtowanie dobrych nawyków finansowych przez prowadzenie domowego budżetu. Wyciąganie wniosków z operacji zwiększa świadomość na temat struktury jego finansów.

System będzie aplikacją desktopową lub webową, z prostym interfejsem graficznym, dostępną dla jednego użytkownika lokalnie.

Zamierzony cel projektu

Cel główny

Głównym celem projektu jest opracowanie prostego, intuicyjnego oprogramowania wspierającego użytkownika w efektywnym zarządzaniu budżetem domowym.

System ma umożliwić wizualizację przepływów finansowych, planowanie wydatków oraz analizę stanu finansowego w różnych okresach czasowych.

Aplikacja ma stanowić praktyczne narzędzie, które pozwoli użytkownikowi zrozumieć strukturę własnych przychodów i kosztów oraz wprowadzić bardziej świadome decyzje finansowe w codziennym życiu.

Cele szczegółowe

Aby zrealizować cel główny, projekt zakłada osiągnięcie następujących celów szczegółowych:

1. **Rejestracja transakcji finansowych** – stworzenie funkcji umożliwiającej wprowadzanie przychodów i wydatków z określeniem ich daty, kwoty, opisu oraz kategorii.
2. **Tworzenie i zarządzanie kategoriami budżetowymi** – użytkownik będzie mógł zdefiniować własne kategorie wydatków (np. jedzenie, mieszkanie, transport, rozrywka) oraz przypisywać do nich transakcje.
3. **Analiza stanu finansowego** – system będzie umożliwiał obliczanie bilansu w danym okresie (np. tygodniowym, miesięcznym, rocznym) oraz prezentację zestawień pokazujących strukturę wydatków i przychodów.
4. **Wizualizacja danych finansowych** – projekt przewiduje generowanie prostych wykresów słupkowych i kołowych, które ułatwią analizę nawyków finansowych i identyfikację obszarów, w których można wprowadzić oszczędności.
5. **Utrzymanie prostoty i czytelności interfejsu** – aplikacja ma być zrozumiała nawet dla osób bez doświadczenia technicznego, a jej obsługa nie powinna wymagać znajomości terminologii finansowej.
6. **Bezpieczeństwo i prywatność danych** – dane użytkownika będą przechowywane lokalnie, w bazie SQLite[1], co eliminuje ryzyko ich udostępnienia podmiotom zewnętrznym.

[1] https://sqlite.org/

Wstępne założenia i uwarunkowania

Projekt zakłada stworzenie prostego systemu do zarządzania budżetem domowym w warunkach akademickich, przy ograniczonym czasie realizacji i zasobach. Aplikacja będzie tworzona przez jedną osobę, w celu praktycznego zastosowania wiedzy zdobytej podczas studiów oraz doskonalenia umiejętności w zakresie programowania i projektowania systemów informatycznych.

1. Językiem programowania wykorzystanym do realizacji projektu będzie Next.js[2] (JavaScript/TypeScript) + Drizzle ORM[3] + HeroUI[4]
2. Aplikacja będzie działać lokalnie, z wykorzystaniem lekkiej bazy danych SQLite[5].
3. System zostanie zaprojektowany zgodnie z zasadami programowania obiektowego.
4. Zastosowany zostanie prosty model projektowy — **model kaskadowy**[6] (Waterfall).
5. Diagramy zostaną wykonane za pomocą programu Draw.io[7].
6. Dokumentacja zostanie wyeksportowana do formatu PDF.
7. Projekt będzie rozwijany etapowo, z raportowaniem postępów na każdych zajęciach.

[2] https://nextjs.org/
[3] https://orm.drizzle.team/
[4] https://www.heroui.com/
[5] https://sqlite.org/
[6] https://pl.wikipedia.org/wiki/Model_kaskadowy
[7] https://draw.io

Założone ograniczenia i możliwość ewaluacji projektu

Projekt realizowany w ramach zajęć akademickich posiada określone ograniczenia wynikające z warunków technicznych, organizacyjnych oraz czasowych. Głównym założeniem jest utrzymanie projektu w zakresie możliwym do wykonania przez jedną osobę, przy zachowaniu jednoczecnym zachowaniem terminów.
Podstawą jest czytelna struktura kodu, dostarczenie prostych funkcjonalności i kompletnej dokumentacji.

Ograniczenia techniczne

W ograniczeniach technicznych wykazałem jakie funkcjonalności nie zostaną zrealizowane. Oto kilka potencjalnych funkcjonalności, które można dodać do systemu, gdyby zainwestować więcej czasu:

Filtrowanie

- Możliwości filtrowania szczegółowego transakcji wprowadzonych do systemu
- Konfigurowanie szczegółowe filtrów dla list elementów wprowadzonych do systemu

Notyfikacje

- Integrowanie dostawców usługi wysyłek SMS i Email
- Notyfikacje sms, email o przekroczeniu budżetów czy innych istotnych zdarzeniach w systemie

Wizualizacja danych

- Zaawansowane wykresy liniowe obrazujące wydatki rozłożone w czasie
- Wykresy kołowe obrazujące proporcje kategorii w ramach budżetów

Użytkownicy

- Jednostanowiskowość systemu oznacza, że aplikacja będzie przeznaczona dla jednego użytkownika i będzie działać lokalnie na jego maszynie.
- Brak funkcji logowania zmniejsza bezpieczeństwo aplikacji i podnosi ryzyko wypłynięcia danych.

Bezpieczeństwo

- Integracja z zewnętrznymi API bankowymi oznacza, że wszelkie transakcje muszą zostać wprowadzone ręcznie.
- Zabezpieczenia kryptograficzne danych przechowywanych w systemie.
- Automatyczne generowanie kopii zapasowych.

Inne

- Możliwość eksportu danych do pliku np PDF i XLS.
- Interfejs użytkownika będzie uproszczony (brak zaawansowanych animacji, responsywności).

Ograniczenia organizacyjne i czasowe

- Prace nad projektem są ograniczone do kilku spotkań w semestrze (okres od października 2025 do stycznia 2026). Zakres funkcji musi być dostosowany do tego harmonogramu.
- Wszystkie elementy projektu (kod, dokumentacja, diagramy) muszą być wykonane samodzielnie, co ogranicza możliwość równoległej pracy nad większą ilością modułów.
- Projekt będzie realizowany z wykorzystaniem ogólnodostępnych narzędzi i bibliotek open source.
- Projekt ma charakter dydaktyczny, dlatego jego celem nie jest stworzenie komercyjnego produktu, lecz praktyczne zastosowanie wiedzy z zakresu inżynierii oprogramowania.

Chronologiczny plan pracy

Praca na projektem została zaplanowana według terminów oddania poszczególnych elementów dokumentacji i projektu. Praca została podzielona na etapy które w przyrostowy sposób doprowadzą do dokończenia projektu w sztywnym terminie.

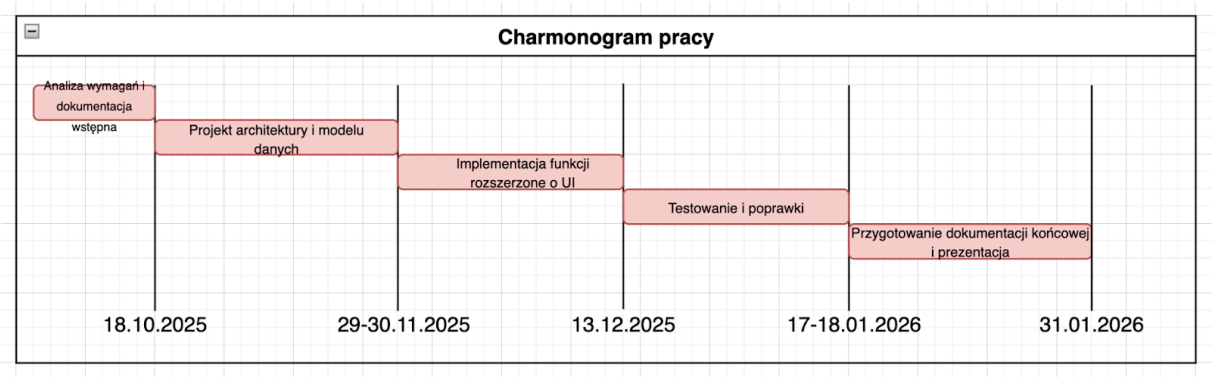


Tabela 1. przedstawiająca chronologię wykonywania projektu w stosunku do terminów ćwiczeń

Etap	Termin	Zakres prac
Analiza wymagań i dokumentacja wstępna	18.10.2025	Opracowanie założeń, planu pracy, opis projektu
Projekt architektury i modelu danych	29-30.11.2025	Diagramy klas, tabele bazy danych, relacje
Implementacja funkcji rozszerzone o UI	13.12.2025	Moduł dodawania przychodów i wydatków, Kategorie, filtry, widok sum miesięcznych
Testowanie i poprawki	17-18.01.2026	Testy funkcjonalne i niefunkcjonalne
Przygotowanie dokumentacji końcowej i prezentacja	31.01.2026	Podsumowanie, wnioski, podręcznik użytkownika

Wymagania funkcjonalne

System Zarządzania Budżetem Domowym powinien realizować następujące funkcjonalności

Obsługa transakcji

Tabela 2. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`addTransaction`
Opis	Dodawanie nowych transakcji (przychody i wydatki)
Dane wejściowe	Obiekt nowej transakcji
Dane wyjściowe	Obiekt utworzonej transakcji

Tabela 3. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`editTransaction`
Opis	Edytowanie istniejących transakcji.
Dane wejściowe	Obiekt istniejącej aktualizowanej transakcji
Dane wyjściowe	Obiekt istniejącej zaktualizowanej transakcji

Tabela 4. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`deleteTransaction`
Opis	Usuwanie transakcji z systemu.
Dane wejściowe	Identyfikator transakcji
Dane wyjściowe	*Brak*

Tabela 5. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`listTransactions`

Tabela 5. Przedstawiająca metodę funkcjonalną	
Opis	Wyświetlanie listy wszystkich transakcji wraz z podstawowymi informacjami (data, kwota, opis, kategoria, typ).
Dane wejściowe	brak
Dane wyjściowe	Tablica dodanych transakcji

Obsługa kategorii

Tabela 6. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`addCategory`
Opis	Dodawanie własnych kategorii budżetowych.
Dane wejściowe	Obiekt nowej kategorii
Dane wyjściowe	Obiekt utworzonej kategorii

Tabela 7. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`editCategory`
Opis	Edytowanie nazw istniejących kategorii.
Dane wejściowe	Obiekt istniejącej aktualizowanej transakcji
Dane wyjściowe	Obiekt istniejącej zaktualizowanej transakcji

Tabela 8. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`deleteCategory`
Opis	Usuwanie kategorii (jeśli nie narusza to integralności danych).
Dane wejściowe	Identyfikator kategorii
Dane wyjściowe	brak

Tabela 9. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`assignCategory`
Opis	Przypisywanie transakcji do wybranych kategorii. (data, kwota, opis, kategoria, typ).
Dane wejściowe	Identyfikator transakcji i Identyfikator kategori
Dane wyjściowe	Obiekt Przypisanej Transakcji

Podstawowa analiza finansowa

Tabela 10. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`sumExpenses, sumIncomes`
Opis	Obliczanie sum przychodów i wydatków w wybranym okresie.
Dane wejściowe	Data początkowa, Data końcowa
Dane wyjściowe	Zestawienie obliczeń

Tabela 11. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`getBalance`
Opis	Obliczanie bilansu (różnicy pomiędzy przychodami a wydatkami).
Dane wejściowe	Data początkowa, Data końcowa
Dane wyjściowe	Zestaw obliczeń

Tabela 12. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`getSummary`
Opis	Wyświetlanie zestawienia podsumowującego dla

Tabela 12. Przedstawiająca metodę funkcjonalną	
	wybranego zakresu dat.
Dane wejściowe	Data początkowa, Data końcowa
Dane wyjściowe	Zestaw obliczeń

Filtrowanie danych

Tabela 13. Przedstawiająca metodę funkcjonalną	
Nazwa funkcji	`filterTransactions`
Opis	Filtrowanie transakcji na podstawie danych wejściowych
Dane wejściowe	Data początkowa, Data końcowa, Kategoria, Typ transakcji
Dane wyjściowe	Filtrowana Lista Transakcji

Przechowywanie danych

- Zapisywanie wszystkich danych w lokalnej bazie SQLite.
- Automatyczne wczytywanie danych podczas uruchamiania aplikacji.
- Trwałe przechowywanie kategorii i transakcji na dysku użytkownika.

Interfejs użytkownika

- Prosty i intuicyjny interfejs umożliwiający szybkie przeglądanie i dodawanie danych.
- Formularze do wprowadzania transakcji i kategorii.
- Widok zbiorczy prezentujący stan bieżącego budżetu.

Wymagania niefunkcjonalne

Wymagania sprzętowe

System Zarządzania Budżetem Domowym jest aplikacją lekką i nie wymaga specjalistycznego sprzętu. Do poprawnego działania wystarczy standardowy komputer osobisty o następujących parametrach minimalnych:

- Procesor: dwurdzeniowy (np. Intel i3 lub odpowiednik AMD).
- Pamięć RAM: minimum 4 GB.
- Miejsce na dysku: min. 200 MB wolnej przestrzeni (na aplikację, bazę danych oraz pliki pomocnicze).
- Monitor o rozdzielczości co najmniej 1280x720 px.
- Klawiatura i urządzenie wskazujące (mysz lub touchpad).

Wariant alternatywny:

System może również działać na komputerach o wyższej wydajności, jednak nie wpływa to na działanie aplikacji, ponieważ obciążenie sprzętowe jest niewielkie.

Wymagania systemowe

Aplikacja po zbudowaniu będzie działać lokalnie i nie wymaga połączenia z Internetem.

Do jej poprawnego uruchomienia wymagane są:

System operacyjny:

- Windows 10 / 11, lub
- macOS, lub
- Linux (np. Ubuntu, Fedora).

Środowisko uruchomieniowe:

- Node.js (w wersji 22 LTS lub nowszej).
- Wsparcie dla TypeScript (kompilator tsc).

Biblioteki i narzędzia wymagane przez projekt:

- SQLite – lokalna baza danych.
- NPM – zarządzanie pakietami aplikacji.

Dodatkowe oprogramowanie potrzebne podczas tworzenia i testowania projektu (wymagane jest połączenie z internetem):

- Edytor kodu (np. Visual Studio Code).
- Przeglądarka internetowa (Chrome / Firefox / Edge) do obsługi interfejsu webowego.

Wymagania organizacyjne

System przeznaczony jest do użytku przez jednego użytkownika na pojedynczej maszynie.

Organizacja pracy z systemem oraz środowisko użycia zakłada:

Jednostanowiskowość – brak możliwości korzystania przez wielu użytkowników jednocześnie.

Brak integracji sieciowych – system działa bez połączenia z Internetem; wszelkie dane wprowadzane są ręcznie.

Warunki poprawnej pracy – aplikacja wymaga jedynie uruchomionej przeglądarki oraz działającego środowiska Node.js.

Wydajność – system nie jest przeznaczony do obsługi dużych zbiorów danych; jest zoptymalizowany pod kątem danych użytkownika indywidualnego.

Bezpieczeństwo danych – dane finansowe są przechowywane w lokalnej bazie SQLite, bez wysyłania ich na zewnętrzne serwery.

Organizacja pracy projektowej – tworzenie i rozwój aplikacji przebiega zgodnie z modelem kaskadowym; kolejne etapy są realizowane i dokumentowane zgodnie z harmonogramem zajęć.

Wymagania dotyczące danych

System korzysta z lokalnej bazy danych SQLite, przechowującej informacje o transakcjach, kategoriach oraz podstawowych ustawieniach aplikacji.

Tabela: Category

Przechowuje listę kategorii, do których użytkownik przypisuje transakcje.

Tabela 14. Przedstawiająca schemat tabeli Category

Pole	Typ danych	Opis	Uzasadnienie
id	`INTEGER PRIMARY KEY AUTOINCREMENT`	Unikalny identyfikator kategorii	Umożliwia jednoznaczne powiązanie z transakcjami
name	`TEXT NOT NULL`	Nazwa kategorii (np. Jedzenie, Transport)	Tekstowy opis danej grupy wydatków/przychodów
type	`TEXT CHECK(type IN ('income','expense')) NOT NULL`	Typ kategorii	Pomaga oddzielić kategorie przychodów od kategorii wydatków

Uwaga: Pole *type* upraszcza logikę filtrowania i generowania wykresów.

Tabela: Transaction

Przechowuje wszystkie zapisane przez użytkownika operacje finansowe.

Tabela 15. Przedstawiająca schemat tabeli Transaction

Pole	Typ danych	Opis	Uzasadnienie

Tabela 15. Przedstawiająca schemat tabeli Transaction			
id	`INTEGER PRIMARY KEY AUTOINCREMENT`	Unikalny identyfikator transakcji	Niezbędny do zarządzania rekordami
amount	`REAL NOT NULL`	Kwota transakcji (dodatnia lub ujemna)	REAL pozwala na zapis wartości dziesiętnych
date	`TEXT NOT NULL (format YYYY-MM-DD)`	Data wykonania transakcji	Przechowywana jako tekst dla prostoty (SQLite nie ma typu DATE)
description	`TEXT`	Krótki opis transakcji	Ułatwia identyfikację wpisów
category_id	`INTEGER NOT NULL`	Powiązanie z kategorią	Relacja do tabeli Category
type	`TEXT CHECK(type IN ('income','expense')) NOT NULL`	Typ transakcji	Podstawowa klasyfikacja wpisu
created_at	`TEXT NOT NULL`	Data dodania transakcji	Umożliwia sortowanie chronologiczne operacji

Relacja: `category_id` → `Category(id)` (klucz obcy)

Tabela: Settings

Przechowuje podstawowe ustawienia aplikacji.

Tabela 16. Przedstawiająca tabeli settings

Pole	Typ danych	Opis	Uzasadnienie
currency	TEXT DEFAULT 'PLN'	Domyślna waluta	Użytkownik może zmienić walutę wyświetlania

Relacje między tabelami

- **Category (1) → Transaction (N)**
  Każda transakcja należy do jednej kategorii.
  Jedna kategoria może mieć wiele transakcji.

- **Settings** nie posiada relacji — zawiera dane globalne.

Uzasadnienie wyboru struktur danych

- **Prostota** – projekt akademicki nie wymaga rozbudowanego modelu wielotabelowego.

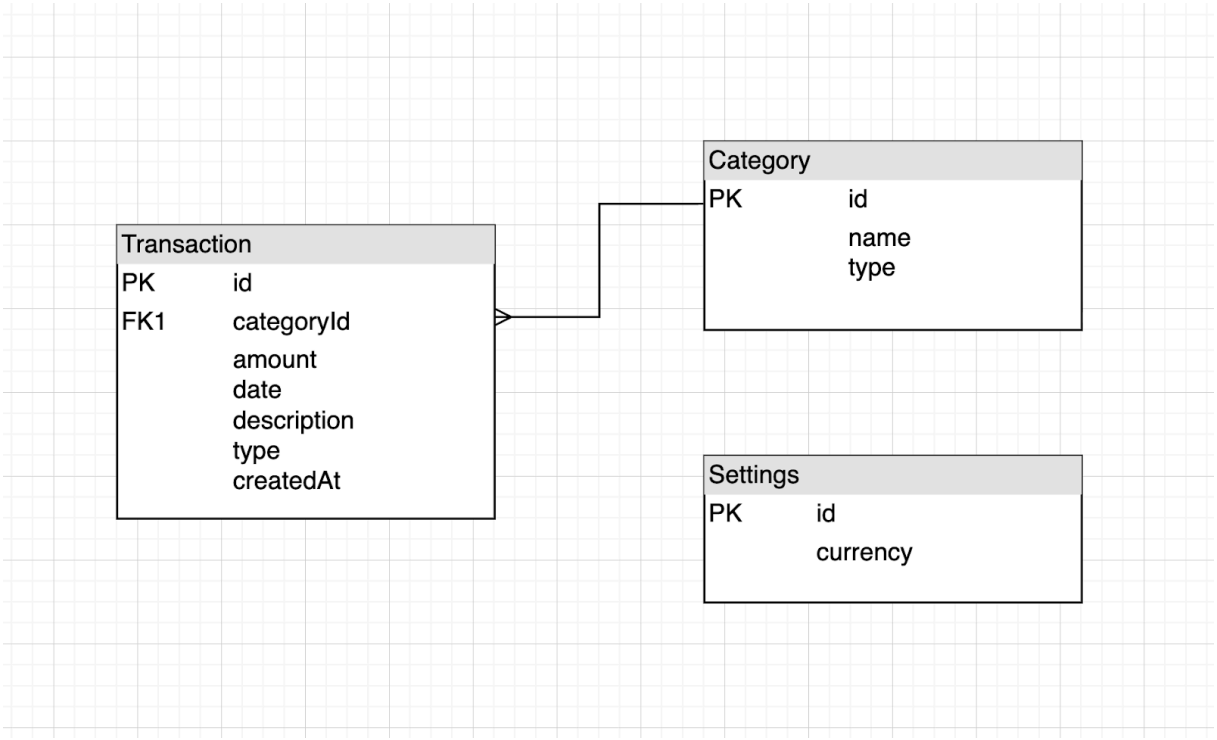
- **Łatwość implementacji** – SQLite w pełni wspiera operacje CRUD dla takich tabel.

- **Czytelność dokumentacji** – model jest łatwy do opisania i przedstawienia na diagramach.

- **Możliwości rozszerzeń** – w przyszłości można dodać tabele: Budżety, Użytkownicy, Limity, Tagowanie transakcji.

Diagram bazy danych

Schemat prezentuje strukturę bazy danych spełniającej podstawowe potrzeby systemu



Rysunek 1 - Diagram bazy danych Systemu zarządzania budżetem

Metody pracy, narzędzia i techniki

Opis głównych klas, metod, obiektów, struktur i algorytmów zastosowanych w projekcje (uwzględniając stosowanie gotowych narzędzi obcego autorstwa, w tym open source).

Technologie

Do uruchomienia kręgosłupa aplikacji skorzystałem z open source frameworka Next.js napisanego w języku JavaScript.

https://nextjs.org/docs/pages/api-reference/cli/create-next-app

Baza danych obsługiwana będzie za pomocą Drizzle ORM

https://orm.drizzle.team/docs/get-started-sqlite

Elementy interface'u dostarczone są przez bibliotekę Open Source HeroUI

https://www.heroui.com/docs/guide/introduction

Diagram Przypadków Użycia

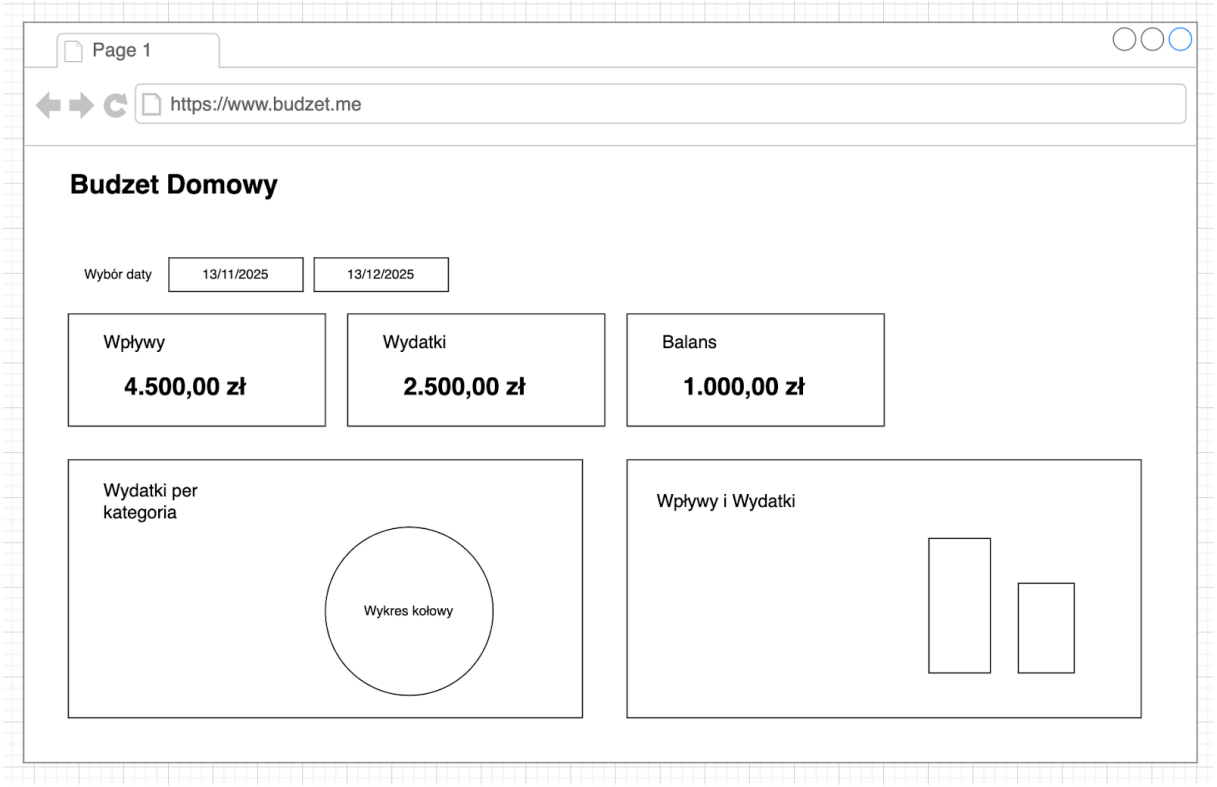
<do wykonania>

Diagram Klas

<do wykonania>

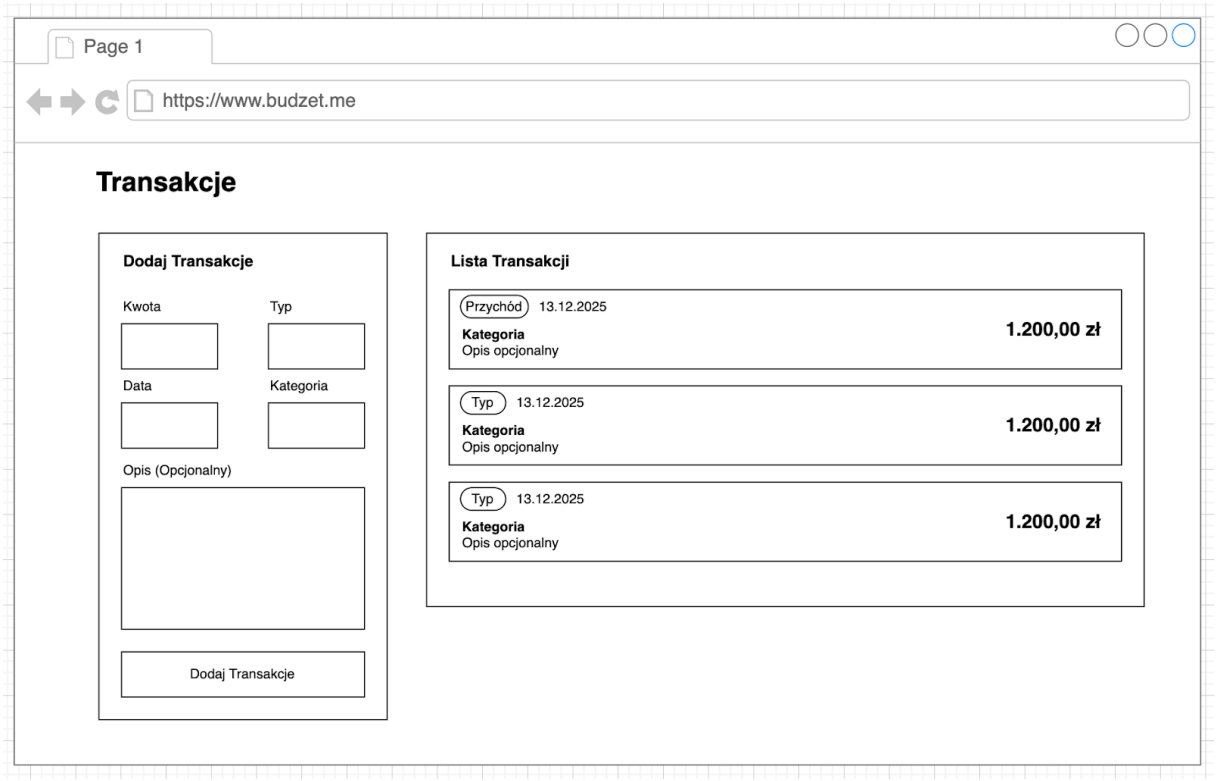
Wireframes

Strona główna prezentuje podsumowanie wpływów i wydatków dla konkretnego okresu



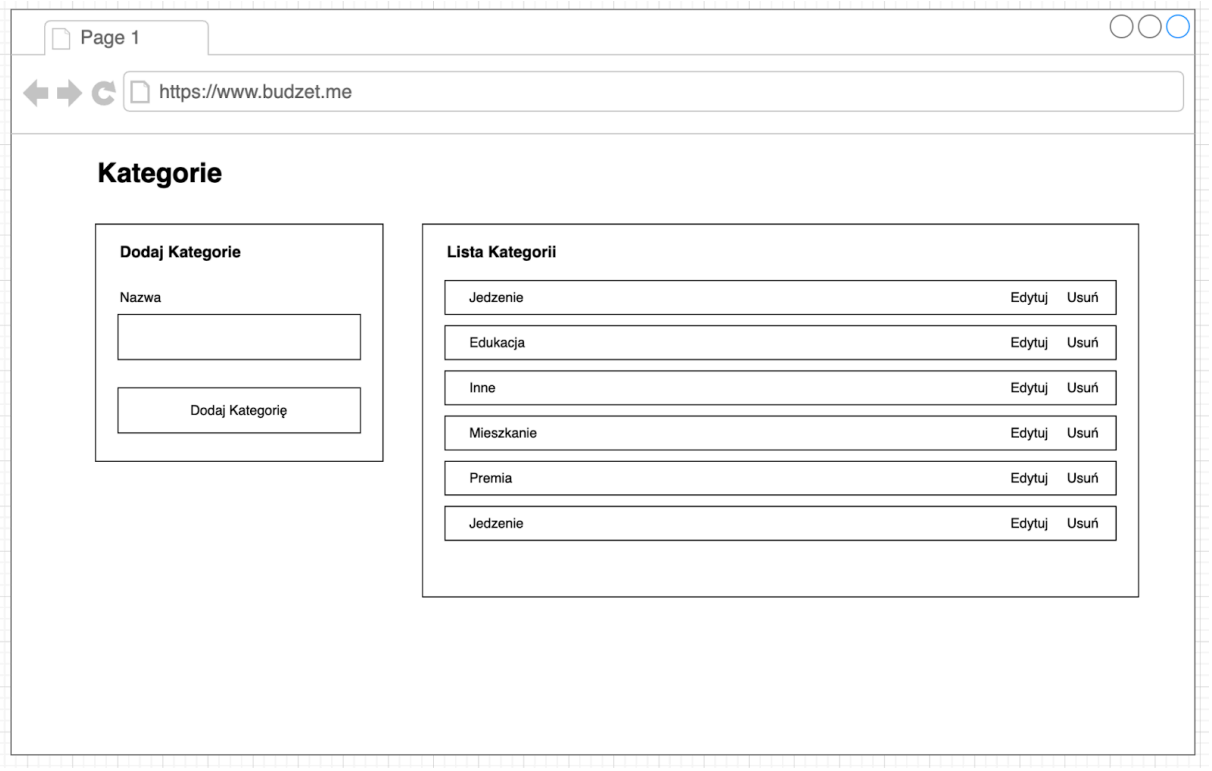
Rysunek 2 - Wireframe przedstawiający Stronę Główną

Strona Transakcje pozwala na dodawanie, listowanie, edycję i usunięcie transakcji



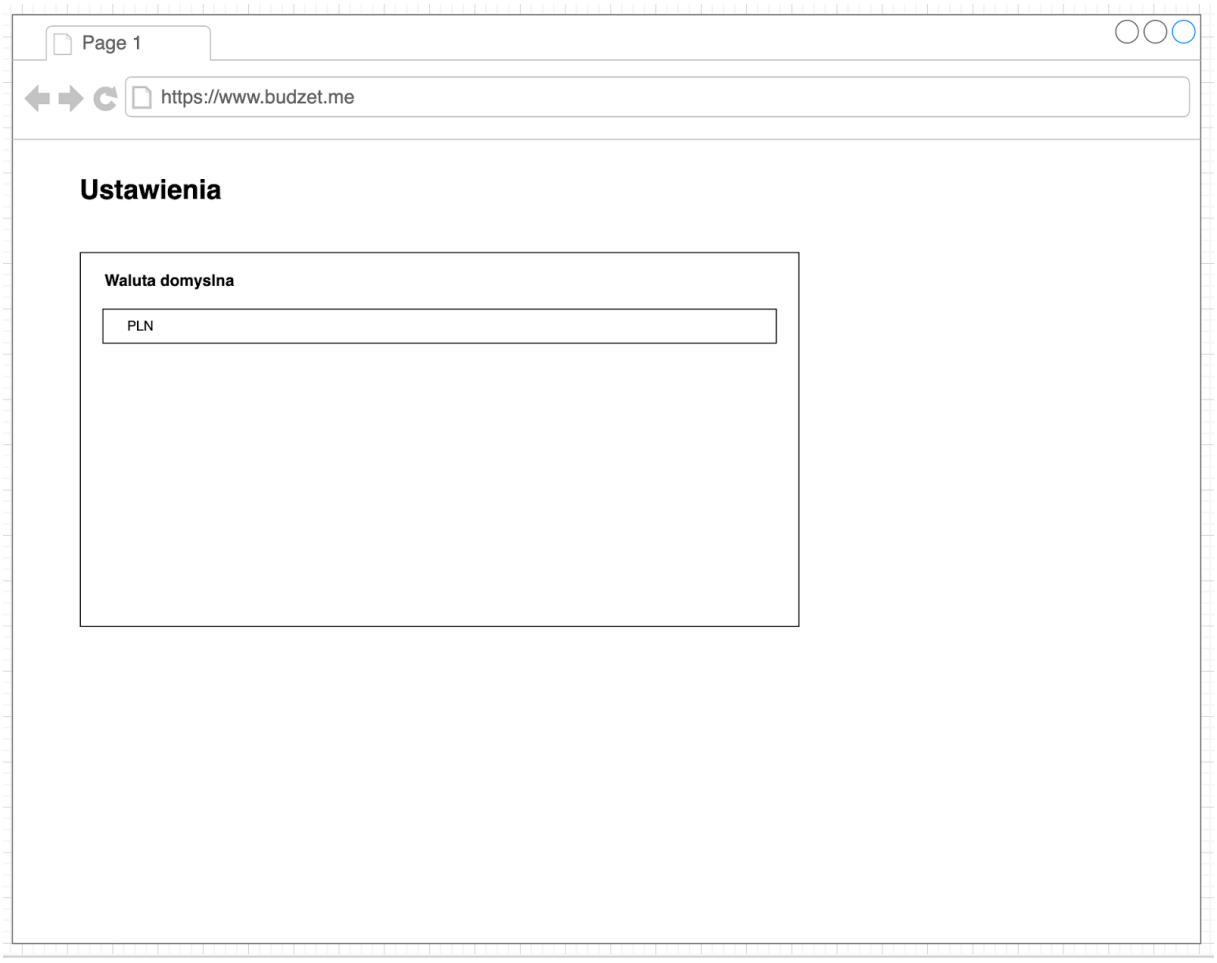
Rysunek 3 - Wireframe przedstawiający Stronę Transakcji

Strona Kategorie pozwala na dodawanie, listowanie, edycję i usunięcie kategorii



Rysunek 4 - Wireframe przedstawiający Stronę Kategorii

Strona Ustawienia pozwala na zmianę ustawień globalnych



Rysunek 5 - Wireframe przedstawiający Ustawienia

Opis Struktur

Tabela 18. - Przedstawia opis struktury strony głównej	
balance	

Tabela 19. - Przedstawia opis struktury transaction	
transaction	

Tabela 20. - Przedstawia opis struktury category	
category	

Tabela 21. - Przedstawia opis struktury settings	
settings	

Spis Metod

Tabela 21. - Przedstawia opis struktury settings	
settings	

Wykaz Algorytmów

<do wykonania>

Bibliografia

Spis Rysunków

Rysunek 1 - Diagram bazy danych Systemu zarządzania budżetem

Spis Tabel

Tabela 1 - przedstawiająca chronologię wykonania projektu w środowisku do terminu ćwiczeń

Tabela od 2. do 13. - Przedstawiająca metodę funkcjonalną

Tabela 14. - Przedstawiająca schemat tabeli category

Tabela 15. - Przedstawiająca schemat tabeli transaction

Tabela 17. - Przedstawiająca schemat tabeli settings

Tabela 18. - Przedstawia opis struktury strony głównej

Tabela 19. - Przedstawia opis struktury transaction

Tabela 20. - Przedstawia opis struktury category

Tabela 21. - Przedstawia opis struktury settings