Projekt. Aplikacja typu menedżer wydatków

Mateusz Kłakus 2IO_II WSTI

Instalacja projektu lokalnie

```
git clone https://github.com/Nokiusz/budget-tracker.git
cd budget-tracker
npm install
cd api
npm install
```

Zbudowanie aplikacji

```
cd budget-tracker
npm run build
```

Tak zbudowaną aplikację (folder `./build`) można przenieść do katalogu `assets` w strukturze plików Android Studio.

Uruchamianie API

```
cd api
npm run watch
```

Funkcje zaimplementowane w aplikacji

- API napisane w node.js używające bazy danych SQLite,
- Wyświetlenie listy wszystkich wydatków/przychodów,
- Dodanie wydatków/przychodów na listę,
- Edycję wydatku/przychodu na liście,
- Usunięcie wydatku/przychodu z listy,
- Wyświetlenie w formie wykresu wydatków z danego miesiąca i/lub kategorii,
- Nadanie wydatkom priorytetu,
- Sortowanie wydatków (np. po kategorii/walucie/typie),
- Dodawanie załącznika do wydatku – zdjęcia paragonu,
- Usunięcie załącznikaz wydatku,
- Wyświetlenie listy załączników,
- podgląd załącznika przypisanego do wydatku,
- Darkmode, ukrywanie wartości.

Dokumentacja REST API:

Endpointy

TRANZAKCJE

```
GET    "/api/transactions" => zwraca obiekt zawierający wszystkie tranzakcje
GET    "/api/transactions/:id" => zwraca pojedyńczą tranzakcję o danym id
GET    "/api/transactions/list" => zwraca listę tranzakcji z polami 'id'
podmienionymi na odpowiednie wartości z tabel słownikowych
GET    "/api/transactions/list/:id" => zwraca  tranzakcję o danym id z polami 'id'
podmienionymi na odpowiednie wartości z tabel słownikowych
POST   "/api/transactions" => dodaje tranzakcje
DELETE "/api/transactions/:id" => usuwa tranzakcje o danym id
PUT    "/api/transactions/:id" => aktualizuje tranzakcje o danym id
```

POST body:

```
{
  "description": "Test POST",
  "value": 3000,
  "categoryId": 2,
  "currencyId": 1,
  "typeId": 2,
  "priorityId": 1,
  "date": "29-04-2022"
}
```

TYPY

```
GET    "/api/types" => zwraca obiekt zawierający wszystkie typy
GET    "/api/types/:id" => zwraca pojedyńczy typ o danym id
```

KATEGORIE

```
GET    "/api/categories" => zwraca obiekt zawierający wszystkie kategorie
GET    "/api/categories/:id" => zwraca pojedyńczą kategorie o danym id
GET    "/api/categories/name/:name" => zwraca pojedyńczą kategorie o danej nazwie
POST   "/api/categories" => dodaje kategorie
DELETE "/api/categories/:id" => usuwa kategorie o danym id
PUT    "/api/categories/:id" => aktualizuje kategorie o danym id
```

POST body:

```
{
  "name": "test category"
}
```

WALUTY

```
GET    "/api/currencies" => zwraca obiekt zawierający wszystkie waluty
GET    "/api/currencies/:id" => zwraca pojedyńczą walutę o danym id
POST   "/api/currencies/" => dodaje walute
DELETE "/api/currencies/:id" => usuwa walute o danym id
PUT    "/api/currencies/:id" => aktualizuje walute o danym id
```

POST body:

```
{
  "name": "test",
  "symbol": "t",
  "acronym": "TST"
}
```

PRIORYTETY

```
GET    "/api/priorities" => zwraca obiekt zawierający wszystkie priorytety
GET    "/api/priorities/:id" => zwraca pojedyńczy priorytet o danym id
GET    "/api/priorities/name/:name" => zwraca pojedyńczy priorytet o danej nazwie
```

Załączniki

```
GET    "/api/attachments" => zwraca obiekt zawierający wszystkie załączniki
GET    "/api/attachments/:id" => zwraca pojedyńczy załącznik o danym id
POST    "/api/attachments" => dodaje załącznik
```

Dokumentacja Endpointów (Swagger)

```
"/docs" => Dokumentacja API oraz możliwość wykonania żądań z poziomu przeglądarki
```

Diagram API

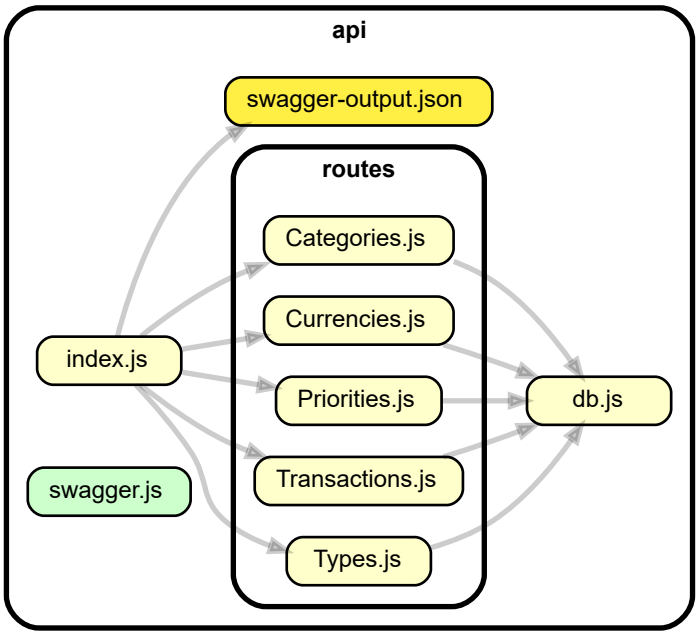
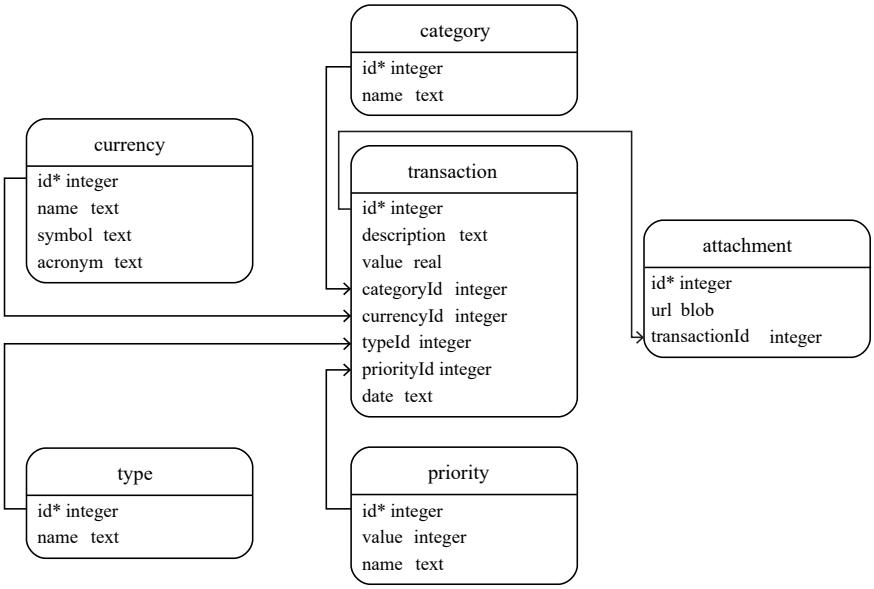

api
swagger-output.json
routes
Categories.js
Currencies.js
index.js
Priorities.js
db.js
swagger.js
Transactions.js
Types.js

Diagram Bazy danych



aplikacje-mobilne.db

Frontend

Warstwa frontendowa aplikacji została napisana za pomocą języka `JavaScript` wraz z biblioteką `React.js`.

Style aplikacji opierają się na systemie projektowania (Design system) `Ant Design`.

Aplikacja składa się z kilku "ekranów":

- Ekran głowny (Lista tranzakcji),
- Ekran dodawania nowej tranzakcji,
- Ekran edycji wybranej tranzakcji,
- Ekran wykresów,
- Ekran wyboru filtrów,
- Ekran ustawień.

Przełączanie pomiędzy ekranami jest realizowane poprzez menu umiejscowione na dole ekranu.

Stan aplikacji zarządzany jest poprzez wbudowane w Reacta `Context API` przechowywane w nim są pobierane dane oraz niektóre funkcje.

Diagram Frontend