### Jakub Przybysz

numer albumu: 45126 kierunek studiów: Informatyka

specjalność: Inżynieria systemów informacyjnych

forma studiów: studia niestacjonarne

### OPRACOWANIE APLIKACJI INTERNETOWEJ DO GRAFICZNEGO PRZEDSTAWIENIA PRZYCHODÓW I WYDATKÓW

**DEVELOPING AN INTERNET APPLICATION FOR THE GRAPHIC PRESENTATION OF REVENUE AND EXPENDITURE**

praca dyplomowa inżynierska

napisana pod kierunkiem:

**dr inż. Tomasza Wiercińskiego**

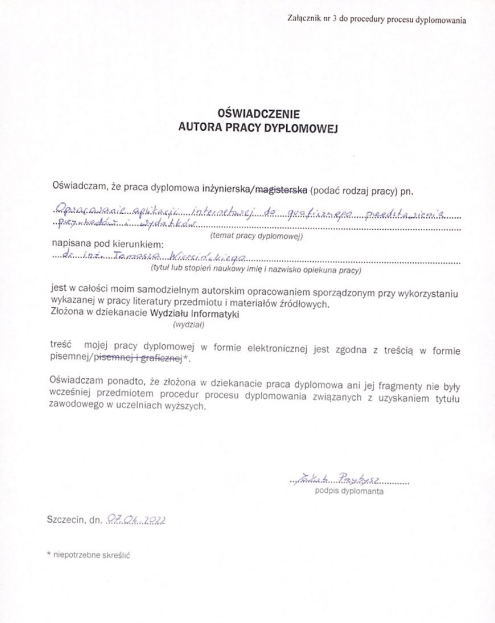
Katedra Inżynierii Oprogramowania i Cyberbezpieczeństwa

Data wydania tematu pracy: 08.06.2021

Data dopuszczenia pracy do egzaminu: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

(uzupełnia pisemnie Dziekanat)

Szczecin, 2022



## Streszczenie

Celem pracy jest zaprojektowanie oraz implementacja aplikacji internetowej do zarządzania przychodami oraz wydatkami za pomocą narzędzi do generowania wykresów. Realizacja projektu wynika z chęci stworzenia produktu, który wyróżniałby się na tle podobnych rozwiązań na rynku aplikacji umożliwiających dowolne zestawianie ze sobą danych. Największą wadą konkurencyjnych aplikacji internetowych jest brak dostępu do zapisanych danych, które mogłyby posłużyć do załadowania nowych diagramów, brakuje im też możliwości zestawienia informacji w innej bardziej czytelnej formie. Zaproponowane rozwiązanie pozwala użytkownikowi na swobodny dostęp do swoich rekordów, dając mu możliwość szybkiego wyszukiwania przychodów lub wydatków po nazwie, istnieje również funkcja swobodnego usuwania i edytowania danej pozycji z tabeli. Dodatkowo każdy z rekordów dodawany przez osobę korzystającą ze strony posiada dokładną datę, co daje jeszcze więcej możliwości w zestawianiu ze sobą informacji i analizowaniu ich na przestrzeni czasu. Aplikacja została napisana w języku Java z wykorzystaniem frameworka Spring, oraz JavaScript z biblioteką React, wykorzystana została również nierelacyjna baza danych MongoDB. W zakres pracy wchodzi analiza rynkowa konkurencyjnych stron internetowych, opis wykorzystanych technologii, stworzenie zarysu projektu oraz implementacja aplikacji wraz z testami.

**słowa kluczowe:** informatyka, programowanie, wykresy, aplikacja internetowa

## Abstract

The purpose of the study is to design and implement a web application for managing income and expenses with the help of tools for generating graphs. The project stems from the desire to create a product that would stand out from similar solutions on the application market, which would allow any combination of data. The biggest disadvantage of competitive web applications is the lack of access to the stored data, which could be used to load new diagrams, they also lack the ability to compile the information in another more readable form. The proposed solution allows users to freely access their records, giving them the ability to quickly search for revenues or expenses by name, there is also a function to freely delete and edit a given item from the table. In addition, each of the records added by the person using the site has an exact date, which gives even more possibilities in compiling information and analyzing it over time. The application will be mainly written in Java with Spring framework and JavaScript with React library, also non-relational database MongoDB will be used. The scope of work includes market analysis of competitive websites, description of technologies used, and creation of project outline and implementation of the application along with tests.

**keywords:** e.g.: computer science, programming, charts, internet application

# Spis treści

Wstęp 9

1. Analiza rynkowa 11

1.1 Aplikacje konkurencyjne 11

1.2 Obszary rozwoju 17

1.3 Podsumowanie 17

2. Wykorzystane technologie 19

2.1 GIT 19

2.2 IntelliJ 20

2.3 Java 21

2.3.1 Spring Framework 21

2.3.2 Spring Boot 22

2.4 MongoDB 23

2.5 Postman 24

2.6 JavaScript 26

2.6.1 React 27

2.6.3 React Bootstrap 28

2.6.4 Pozostałe biblioteki JavaScript 29

3. Zarys projektu 31

3.1 Wymagania 31

3.1.1 Wymagania funkcjonalne 31

3.1.2 Wymagania niefunkcjonalne 32

3.2 Przypadki użycia 32

3.2.1 Diagram przypadków użycia 33

3.3 Baza danych 42

3.3.1 Schemat bazy danych 42

3.4 Diagram klas 45

3.5 Projekt interfejsu 45

4. Implementacja i testowanie 55

4.1 Środowisko 55

4.2 Elementy projektu 55

4.3 Implementacja bazy danych 57

4.4 Interfejs użytkownika 59

4.4.1 Strona główna 59

4.4.2 Strona rejestracji użytkownika 61

4.4.3 Strona logowania użytkownika 62

4.4.4 Strona dodawania przychodów i wydatków 63

4.4.5 Strona z danymi użytkownika 64

4.4.6 Strona z wykresami przychodów i wydatków 66

4.4.7 Strona dodawania sparametryzowanego wykresu 67

4.5 Testy 69

4.5.1 Testy jednostkowe 69

4.5.2 Testy integracyjne 70

4.5.3 Testy manualne 71

4.6 Problemy implementacyjne 72

Podsumowanie 75

Bibliografia 77

Spis rysunków 79

Spis tabel 81

Spis kodów źródłowych 83

## **Wstęp**

Aplikacje internetowe w dzisiejszych czasach są nieodłączną częścią życia każdego człowieka, pełnią już nie tylko rolę źródeł informacji lecz zaspokajają coraz więcej potrzeb przeciętnego użytkownika sieci. Mocno zauważalne tempo wzrostu rozwoju technologicznego uzależniło w pewnym stopniu współczesnego człowieka od internetu, systemy wykorzystywane między innymi w komunikacji, edukacji oraz bankowości wymuszają podążanie za zmieniającym się światem. Znaczna większość stron internetowych powstała z myślą o wygodzie potencjalnego użytkownika, możliwość dostępu do najpotrzebniejszych informacji w jednym miejscu oraz urozmaicone systemy bezpieczeństwa pozwalają na załatwienie niezliczonej ilości spraw w tym samym momencie. Pomimo ogromnej ilości aplikacji internetowych skupiających się głównie na rozrywce istnieją również narzędzia wspierające użytkowanie skomplikowanych systemów w przemyślany oraz odpowiedzialny sposób.

Jedną z większych zalet stron internetowych jest swobodny dostęp do bankowości, który pozwala na robienie przelewów oraz sprawdzanie stanu własnego konta. W dzisiejszych czasach ilość nabywanych dóbr oraz rozpowszechniony konsumpcjonizm sprawia, że wyjątkowo trudnym staje się kontrolowanie własnych finansów. Fakt istnienia płatności internetowych oraz zbliżeniowych odbiera użytkownikowi poczucie kontroli nad własnymi środkami. Analiza przychodów oraz wydatków posługując się jedynie systemem bankowym może stać się utrapieniem w momencie próby znalezienia istotnej nieprawidłowości.

Zewnętrzne strony internetowe niepowiązane z bankami mogą służyć do kontroli oraz analizy własnego budżetu domowego, pozwalają zestawić ze sobą najważniejsze informacje przekazując użytkownikowi wynik, z którego ten może wyciągnąć wnioski oraz podjąć konkretne działania. Najczytelniejszą formą zestawiania ze sobą danych są wykresy, pozwalają one na wyśledzenie wyróżniających się wartości dając użytkownikowi faktyczną wiedzę na temat swojej historii finansowej. Wizualizacja przychodów i wydatków w odpowiednich formach może pomóc określić nieprawidłowości związane z nawykami, których przeciętny konsument nie dostrzegał.

Głównym powodem braku poczucia pełnej kontroli nad własnym stanem konta mogą być liczne transakcje niezauważalne w ciągu jednego dnia jednak ich suma na przestrzeni tygodni generuje ogromną kwotę będącą zaskoczeniem nawet dla właściciela konta. Istnieje szereg metod pozwalających na kontrole budżetu niwelujące błędy powstałe podczas nieprzemyślanego rozdysponowywania własnym majątkiem takie jak podział przychodów oraz wydatków na odpowiednie kategorie, regularne i konsekwentne rejestrowanie każdej wartości przychodzącej lub wychodzącej oraz odpowiednie analizowanie konkretnych danych na przestrzeni czasu zestawiając je w perspektywie minionych tygodni, miesięcy oraz lat.

**Cel i zakres pracy**

Celem pracy jest zaprojektowanie aplikacji internetowej do graficznego przedstawiania przychodów i wydatków (Charts.io), której zadaniem będzie spełnienie wszystkich wymagań dotyczących kontroli budżetu użytkownika wspomnianego powyżej dostarczając pełną wiedzę na temat finansów osoby korzystającej ze strony. Serwis pozwoli na przechowywanie danych finansowych użytkownika oraz zwracanie ich w odpowiedniej formie dostarczając szereg narzędzi do tworzenia sparametryzowanych grafów w celu łatwiejszej interpretacji dużej ilości danych.

W zakres pracy wchodzi szereg celów cząstkowych takich jak:

* Analiza rynkowa, obejmująca przedstawienie konkurencyjnych aplikacji posiadających podobne zastosowanie oraz zestawienie ich wraz ze swoimi rozwiązaniami technologicznymi,
* Dokładny opis technologii wykorzystywanych podczas projektowania oraz implementacji aplikacji internetowej,
* Zarys projektu,
* Implementacja aplikacji internetowej.

Efektem zrealizowania powyższych celów cząstkowych będzie:

* Przeanalizowanie stron internetowych o podobnych zastosowaniach www.onlinecharttool.com, www.charts.livegap.com, www.docs.google.com/spreadsheet, pod kątem ich funkcjonalności, rozwiązań oraz dostrzegalnych braków, znalezienie potencjalnych powodów ich pozycji na rynku, wyciągnięcie wniosków wraz z wyszukaniem możliwych obszarów rozwoju;
* Opis następujących technologii: Git (system kontroli wersji); IntelliJ (IDE zintegrowane środowisko programowania); język programowania Java; framework Spring; nierelacyjna baza danych MongoDB; Postman (platforma do projektowania i testowania REST API); język programowania JavaScript; React (biblioteka JavaScript); React Google Charts (nakładka React na serwis Google Charts do generowania wykresów); framework React-Bootstrap,
* Określenie zarysu projektu, który obejmuje wymagania funkcjonalne i niefunkcjonalne; przypadki użycia w formie diagramów oraz scenariuszy; schemat bazy danych; diagram klas; zaprojektowanie całego interfejsu strony,
* Zaimplementowanie aplikacji internetowej; opisanie środowiska programistycznego; wyszczególnienie i opis elementów projektu; zdefiniowanie wykorzystanych wzorców projektowych; implementacja bazy danych; testy manualne, jednostkowe oraz integracyjne; zaprezentowanie interfejsu strony użytkownika.

## **1. Analiza rynkowa**

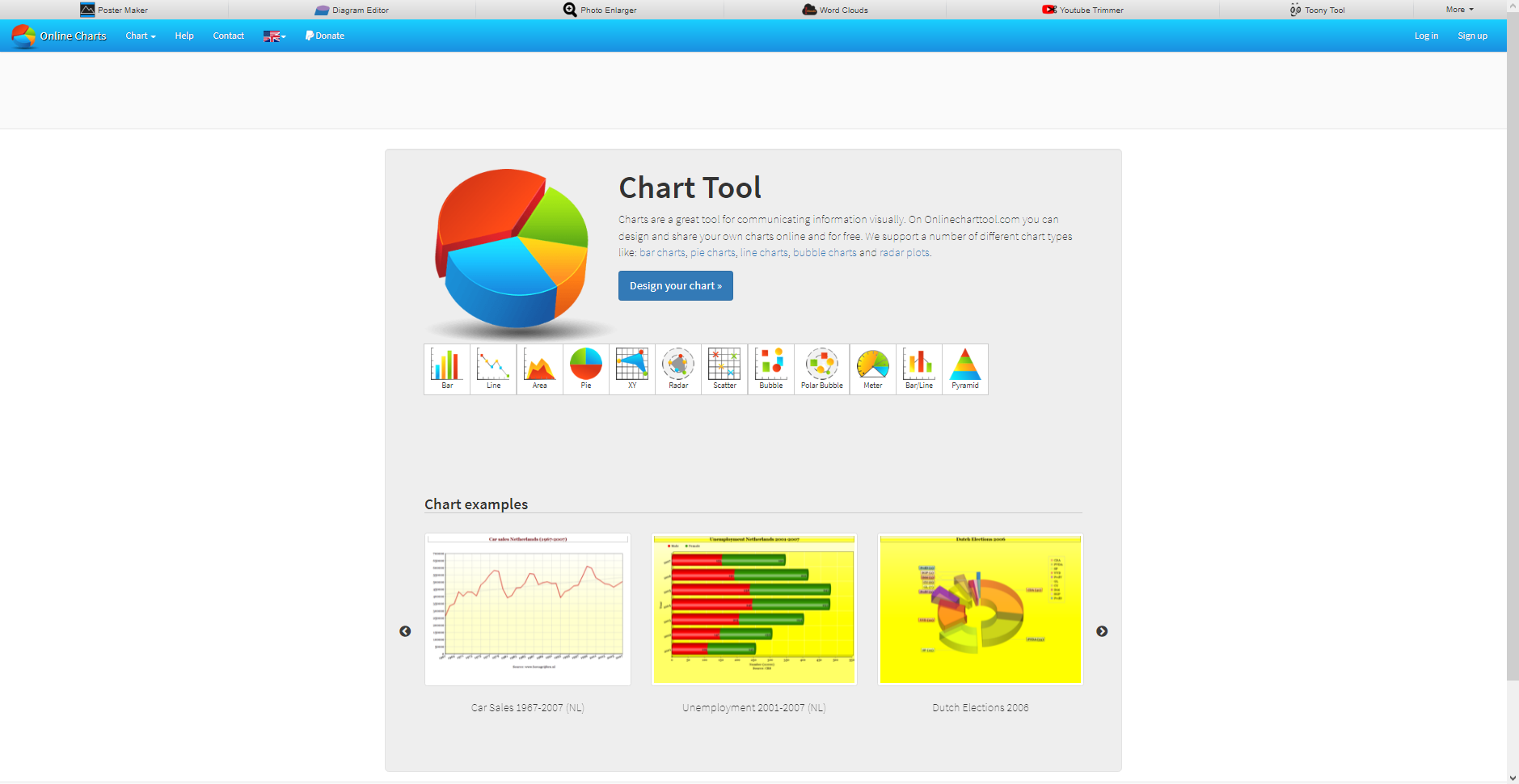
Według definicji analiza rynkowa to ogół czynności mających na celu zrozumienie pewnych procesów tłumaczących pozycje danego podmiotu na rynku[[1]](#footnote-1). W przypadku analizy stron internetowych kluczowymi elementami są spójność oprawy wizualnej, jej estetyka oraz responsywność[[2]](#footnote-2). Popularność aplikacji może wynikać z wielu czynników, nie rzadko główną rolę odgrywa marketing, który pomaga rozpromować daną witrynę, zazwyczaj jednak to funkcjonalność połączona z dopasowanymi rozwiązaniami do wymagań użytkownika przemawia na korzyść konkretnego serwisu. Zadaniem niniejszej analizy rynkowej jest zestawienie ze sobą najpopularniejszych aplikacji internetowych pozwalających na tworzenie oraz edytowanie wykresów na podstawie danych użytkownika. Analiza ma na celu przedstawienie zalet i wad poszczególnych aplikacji oraz charakterystykę wyróżniających się rozwiązań.

## **1.1 Aplikacje konkurencyjne**

Online Charts to strona internetowa pozwalająca użytkownikowi na zaprojektowanie oraz wygenerowanie dowolnych wykresów za darmo.

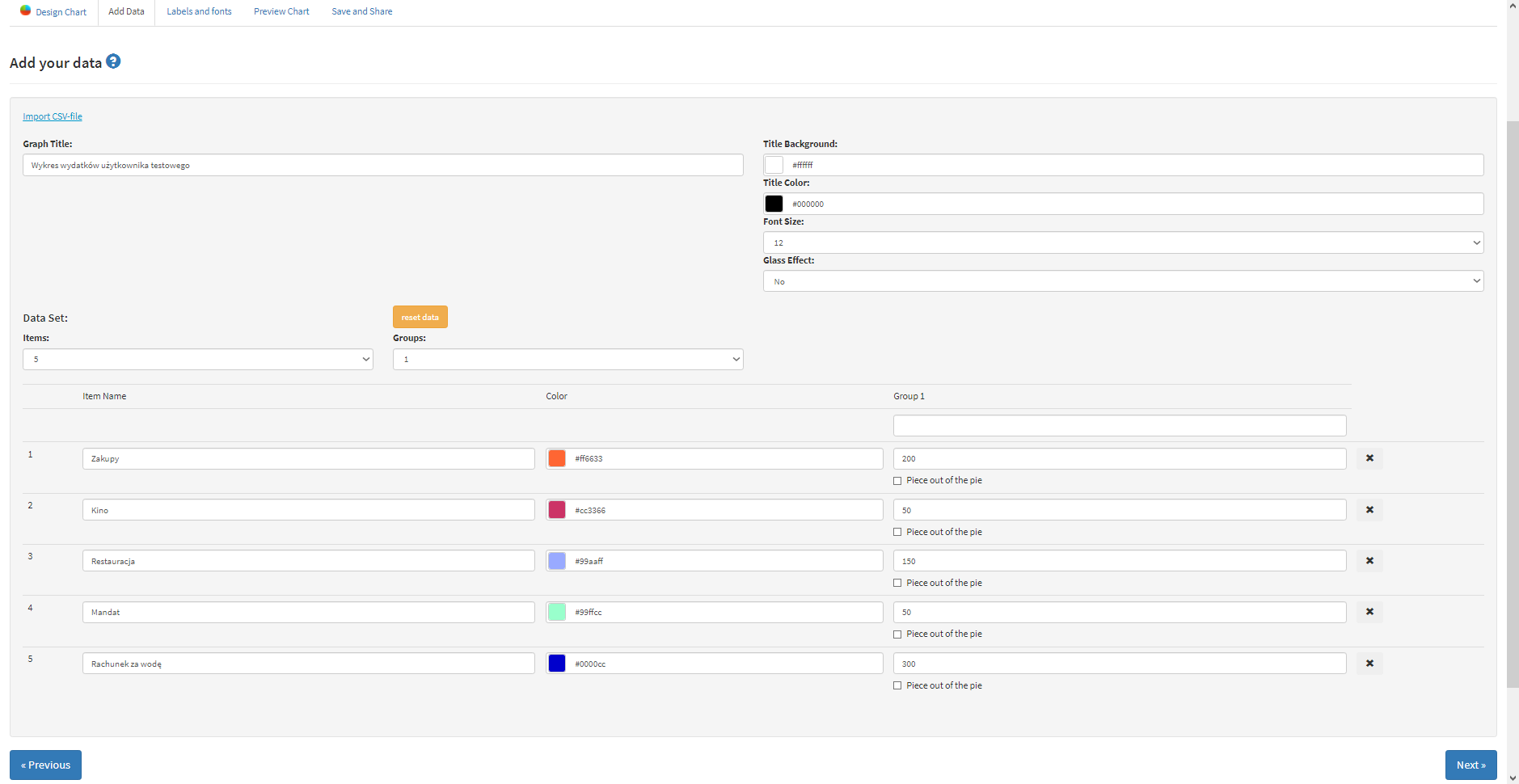
Największą zaletą strony jest czytelny interfejs, pozwalający osobie korzystającej ze strony na wybór interesujących go diagramów oraz opcji na jak najlepsze zestawienie ze sobą rekordów. Kolejną istotną zaletą jest brak konieczności logowania się, brak konta nie rzutuje w żaden sposób na podstawową funkcjonalność aplikacji. Sposób generowania grafu polega na przejściu pomiędzy zakładkami uzupełniając wymagane pola, gdzie można dostosować odpowiednio: kolor tła wykresu, jego tytuł, odpowiednie dane, a nawet styl oraz rozmiar czcionki. Posiadanie konta na stronie onlinecharttool.com pozwala na zapisanie oraz edytowanie wygenerowanego już wcześniej wykresu oraz eksportowanie go w takim formacie jak SVG, PNG, JPG, PGF czy też CSV. Możliwość pobrania diagramu z różnymi rozszerzeniami posiadają również użytkownicy niezalogowani. Pomimo wyjątkowo czytelnego interfejsu użytkownika sam proces zaprojektowania grafu jest bardzo czasochłonny, co ciągnie za sobą konsekwencje w momencie, kiedy użytkownikowi zależy na jak najszybszym zestawieniu ze sobą wielu informacji.

Największą wadą aplikacji internetowej jest brak możliwości gromadzenia użytych wcześniej danych często potrzebnych do utworzenia nowych wykresów, jedynym sposobem na usprawnienie procesu dodawania nowego grafu jest przekopiowanie informacji z zapisanych już wcześniej diagramów i wczytanie ich ponownie podczas tworzenia nowego wykresu, co wiąże się już z koniecznością posiadania konta. Poniżej zostały przedstawione rysunki strony głównej, panelu dodawania wykresu oraz efekt finalny, jakiego można spodziewać się korzystając z aplikacji.



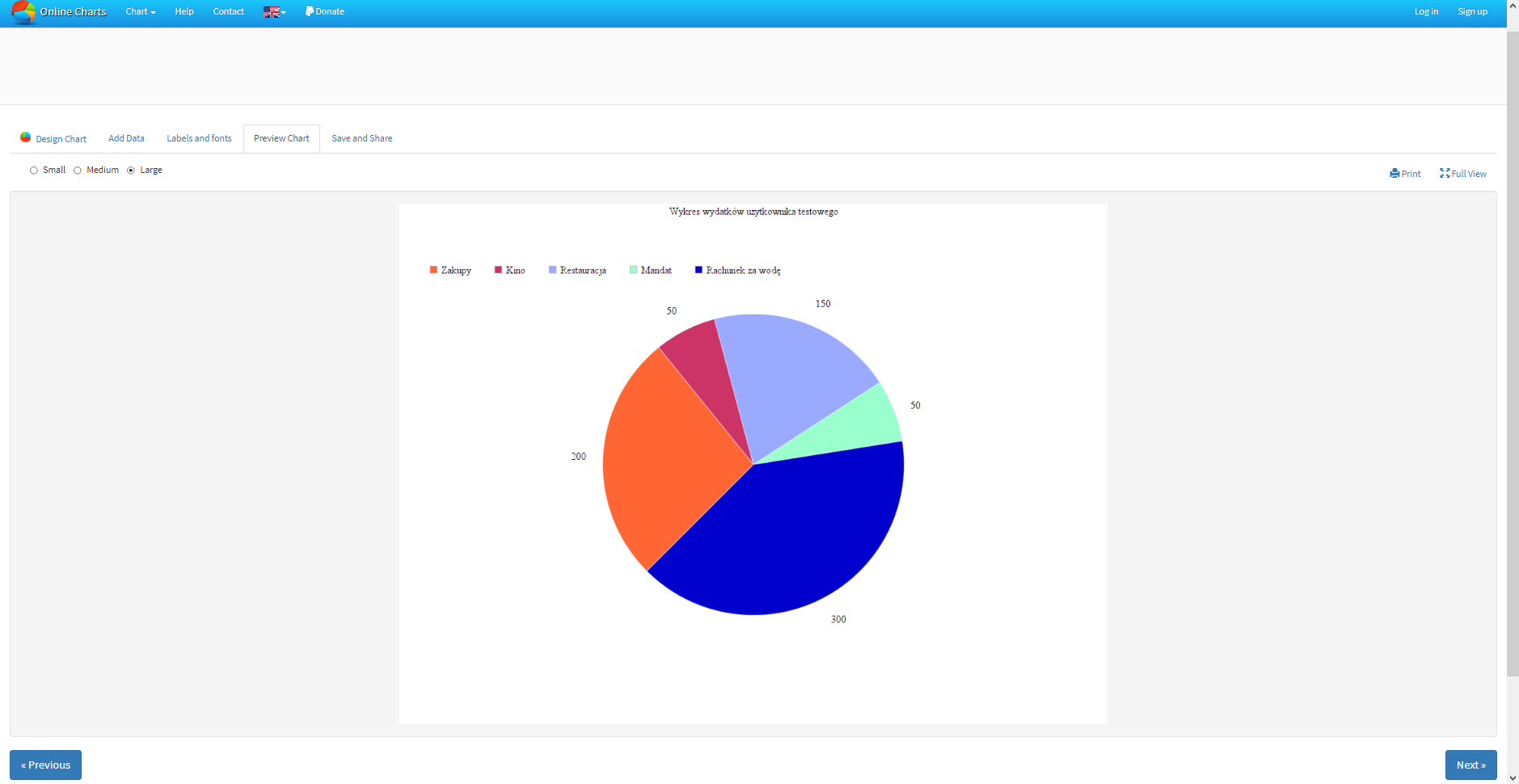
**Rysunek 1.1.** Strona główna Online Charts

Źródło: Na podstawie <https://www.onlinecharttool.com>



**Rysunek 1.2.** Panel dodawania wykresu strony Online Charts

Źródło: Na podstawie https://www.onlinecharttool.com/graph



**Rysunek 1.3.** Wygenerowany wykres strony Online Charts

Źródło: Na podstawie https://www.onlinecharttool.com/graph

Kolejną stroną jest Livegap Charts, czyli bezpośrednia konkurencja dla omawianej poprzednio aplikacji internetowej. Witryna cieszy się dużą popularnością w sieci, na stronie głównej można znaleźć informację o ponad jednym milionie użytkowników korzystających z aplikacji na całym świecie oraz dwóch milionach wygenerowanych wykresów. Stronę wyróżnia nowoczesny design oraz szeroki zakres możliwości w wyborze diagramów. Jednym z aspektów, który odróżnia charts.livegap.com od pozostałych serwisów na rynku o podobnych rozwiązaniach jest duży nacisk na social media. Twórcy strony zachęcają do udostępniania swoich wykresów na najpopularniejszych platformach takich jak Facebook, Twitter czy Reddit, co może być bezpośrednim czynnikiem wpływającym na tak duże zainteresowanie stroną. Widoczna jest również spójność oprawy wizualnej, oraz estetyka całej strony odbiegająca od tego, co prezentują inne konkurencyjne aplikacje internetowe.

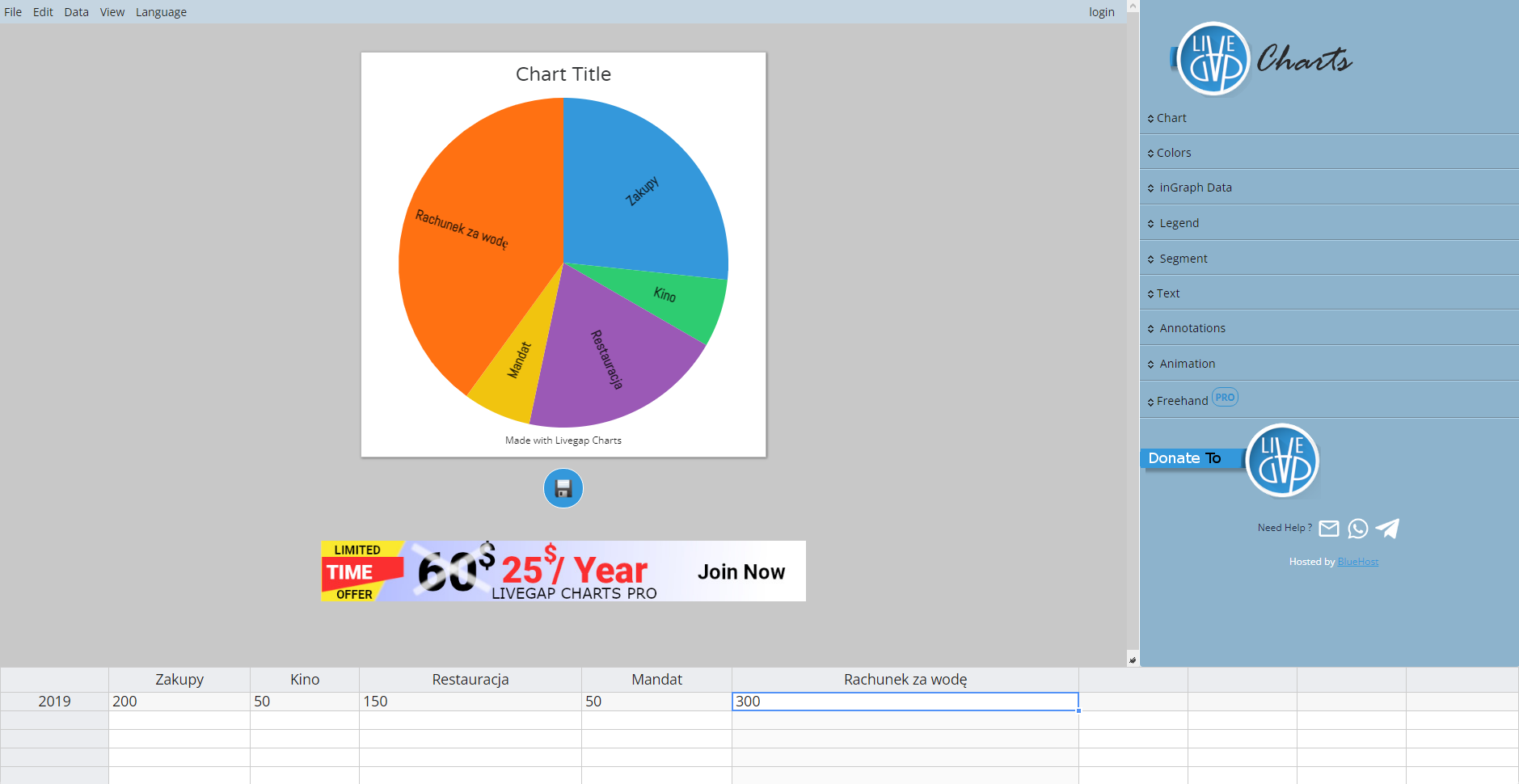
W przeciwieństwie do Online Charts generowanie nowych diagramów jest o wiele szybsze, użytkownik chcąc utworzyć nowy wykres wybiera jedynie rodzaj grafu, który automatycznie załadowuje się z przykładowymi danymi. Cały proces tworzenia nowego diagramu polega jedynie na dostosowaniu istniejących już informacji do własnych potrzeb zmieniając wartości pól, przedział czasowy, nazwę wykresu czy też czcionkę. Każdy użytkownik nawet ten niezalogowany ma możliwość pobrania grafu, jako obraz w formacie PNG lub wideo WEBM, istnieje również możliwość wyeksportowania go w formacie CSV.

Największą wadą całej strony jest brak jakichkolwiek korzyści płynących z posiadania konta, każda dodatkowa funkcjonalność, jaką oferuje serwis jest płatna, użytkownik jest zmuszony do zapłacenia za możliwość zapisania wykresu na stronie oraz wyeksportowania go w innym formacie. Dodatkowym minusem jest również znak wodny, którego pozbyć się można jedynie w momencie wykupienia abonamentu na stronie. Poniższe rysunki przedstawiają stronę główną aplikacji internetowej oraz panel tworzenia nowego wykresu.



**Rysunek 1.4.** Strona główna Livegap Charts

Źródło: Na podstawie https://charts.livegap.com



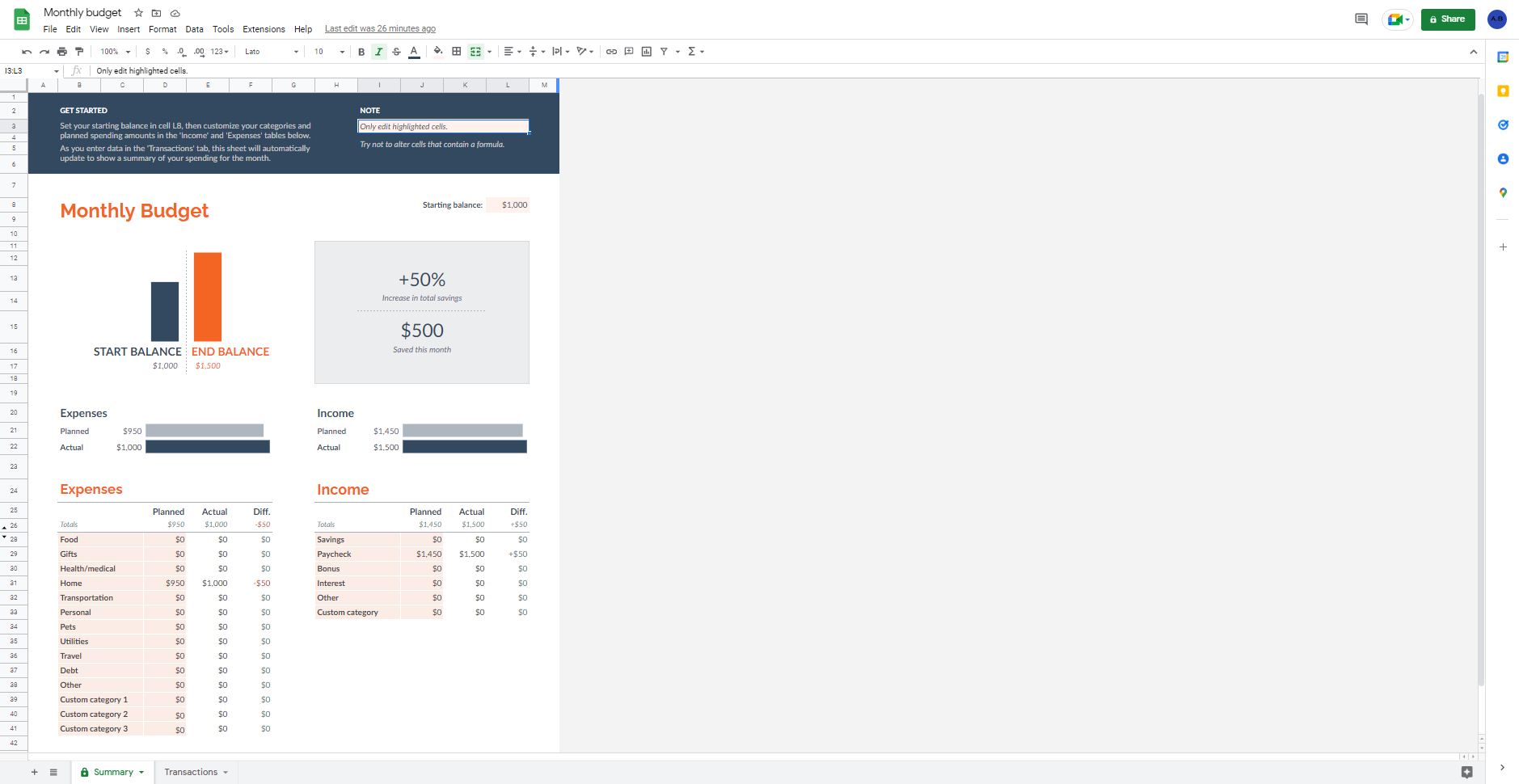
**Rysunek 1.5.** Wygenerowany wykres wraz panelem edycji danych strony Livegap Charts

Źródło: Na podstawie https://charts.livegap.com/app.php?lan=en&gallery=pie

Stroną cieszącą się największą popularnością w tym zestawieniu jest Google Sheets, aplikacja posiadająca ogromne możliwości w aspekcie gromadzenia dużej ilości danych i przedstawiania ich w czytelny sposób. Ta aplikacja internetowa może być świetnym narzędziem do porównywania ze sobą przychodów i wydatków użytkownika poprzez generowanie odpowiednich wykresów. Czytelny interfejs strony, jej przejrzystość połączona z niezwykle rozbudowaną funkcjonalnością sprawia, że Google Sheets nie ma bezpośredniej konkurencji na rynku.

Największą zaletą strony jest bez wątpienia jej uniwersalność w możliwości operowania na wszelkiego rodzaju danych, umożliwiając użytkownikowi ogromny wybór, co do formy zarządzania swoimi rekordami. W przeciwieństwie do Online Charts aplikacja internetowa nie posiada panelu umożliwiającego sprawne dodawanie informacji, alternatywą są szablony, których zadaniem jest wygenerowanie przygotowanych przez aplikacje sparametryzowanych form pozwalających użytkownikowi pracować na gotowych ustawieniach. Wybierając szablon „Monthly budget” w prosty sposób można zestawić ze sobą przychody i wydatki oraz wygenerować dla nich wykresy, wówczas zasada działania przypomina mocno funkcjonalność strony Livegap Charts gdzie użytkownik chcąc dodać własne dane wpierw musiał zastąpić przykładowe informacje na wygenerowanym wykresie. Google Sheets umożliwia eksportowanie swoich plików w formatach XLSX, ODS, PDF, HTML, CSV a nawet TSV, istotnym faktem jest, iż korzystanie z pełnej funkcjonalności witryny jest w pełni darmowe, wymaga jedynie posiadania konta Google uprawniającego również do korzystania z takiej aplikacji jak Google Docs.

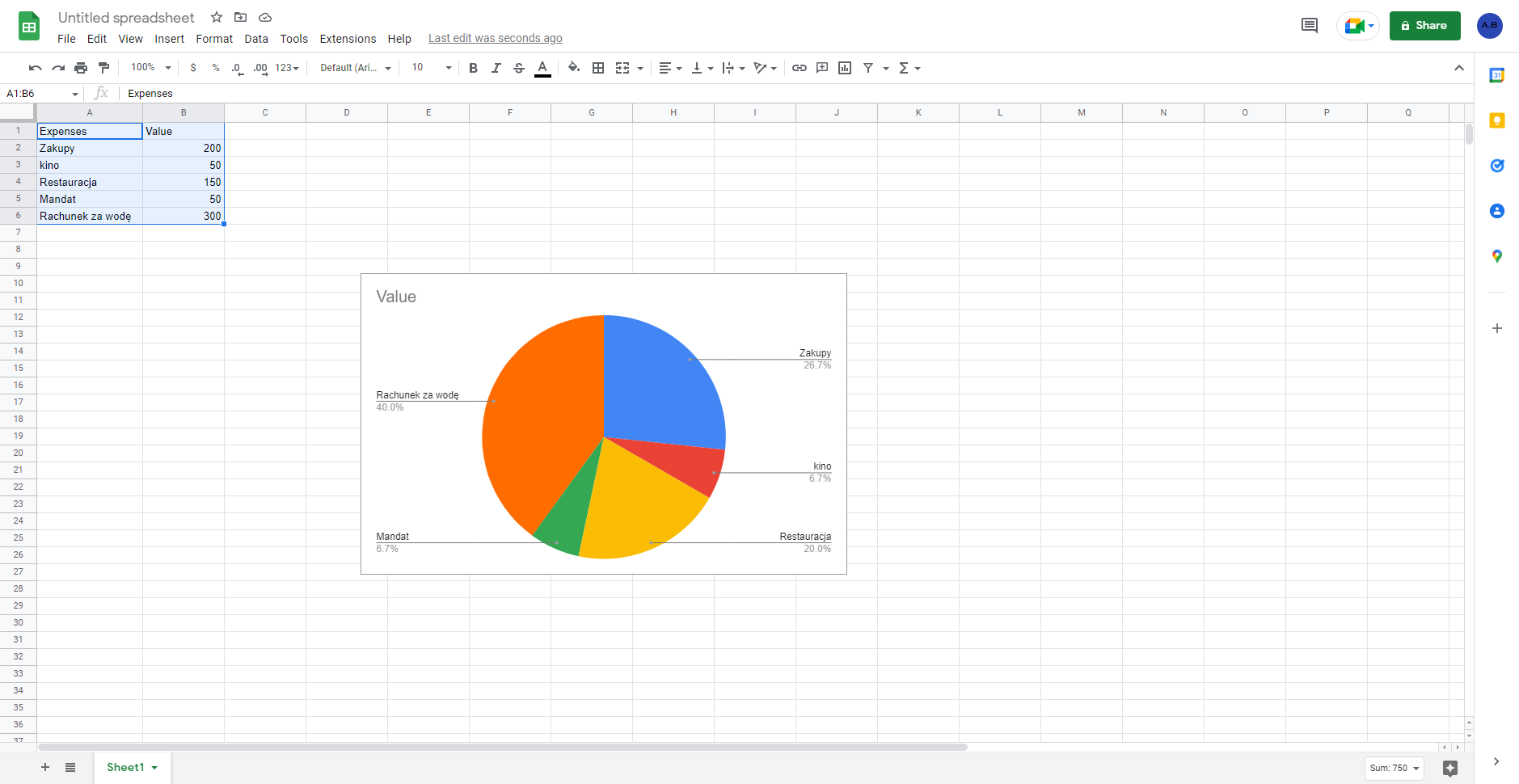
Największą wadą strony jest bez wątpienia fakt, że jest jedną z bardziej skomplikowanych aplikacji internetowych na rynku o podobnych zastosowaniach a opanowanie większości z oferowanych przez nią narzędzi zajmuje sporo czasu szczególnie nowemu użytkownikowi. Korzystanie z Google Sheets wymaga rozważnego zaplanowania formy, w jakiej mają być zestawiane ze sobą dane, zwłaszcza w momencie aktywnego korzystania z tych informacji. Nieumiejętne wpisywanie kolejnych rekordów tworząc niepotrzebne kolumny i wiersze staje się dużym problemem nie tylko podczas generowania wykresów, lecz również w odczytywaniu wciąż powiększającej się tabeli na przestrzeni kolejnych miesięcy. Poniższe rysunki obrazują możliwości aplikacji internetowej, przykładowy szablon, który oferuje nam strona oraz czysty arkusz z naniesionymi informacjami, dla których wygenerowano wykres.



**Rysunek 1.6.** Szablon Monthly budget strony Google Sheets

Źródło: Na podstawie

https://docs.google.com/spreadsheets/d/1wDDp8YIF3Y3zjgXxWy7EfSLRejF8\_oVs5lXo7P5zdoE/edit#gid=0



**Rysunek 1.7.** Wygenerowany wykres wraz z tabelą danych strony Google Sheets

Źródło: Na podstawie

https://docs.google.com/spreadsheets/d/12YO9rd29INlhYs6QdBrtOYecckKnd6FAGXeu9e1iQec/edit#gid=

## **1.2 Obszary rozwoju**

Każda z przeanalizowanych stron internetowych posiada cechy, które mogłyby stać się potencjalnymi obszarami rozwoju dla aplikacji Charts.io. Jednym z ciekawszych rozwiązań jest generowanie gotowych już wykresów z przykładowymi danymi, pozwalając użytkownikowi jedynie na zmianę lub ewentualne dodanie interesujących go pól, co bez wątpienia przyśpiesza całość procesu tworzenia diagramu. Najbardziej interesującą funkcjonalnością jest możliwość eksportowania wygenerowanych wykresów do takich formatów jak CSV czy też PNG pozwalając na wykorzystanie ich w innym celu. Ponadto aplikacje konkurencyjne pozwalają na edytowanie stylów swoich wykresów takich jak rozmiar czcionki, kolor tła czy też własny tytuł wykresu. Wymienione obszary posłużą do ewentualnego rozwoju niniejszej aplikacji internetowej.

## **1.3 Podsumowanie**

Zadaniem aplikacji Charts.io będzie zniwelowanie największych wad występujących na stronach konkurencyjnych oraz dostosowanie jej funkcjonalności do potrzeb jak największej ilości użytkowników. Poniższa tabela prezentuje główne wady aplikacji konkurencyjnych.

**Tabela 1.1.** Wady aplikacji konkurencyjnych

Źródło: opracowanie własne

|  |  |  |  |
| --- | --- | --- | --- |
| Wady konkurencyjnych aplikacji internetowych: | Online Charts | Livegap Charts | Google Sheets |
| Czasochłonne projektowanie grafu. | Tak | Nie | Nie |
| Brak możliwości skorzystania z użytych wcześniej danych. | Tak | Nie | Nie |
| Nakładany znak wodny przy wygenerowanym wykresie. | Nie | Tak | Nie |
| Brak dostępu do swoich danych bez wcześniejszego wykupienia konta premium. | Nie | Tak | Nie |
| Dodatkowe funkcjonalności strony są płatne. | Nie | Tak | Nie |
| Konieczność przyswojenia dużej ilości wiedzy na temat większości narzędzi oferowanych przez stronę. | Nie | Nie | Tak |
| Problem z wizualizacją dużej ilości informacji w domyślnej formie tabeli. | Nie | Nie | Tak |
| Utrudnione dodawanie kolejnych rekordów w przypadku dużej ilości danych. | Nie | Nie | Tak |
| Problem z wizualizacją dużej ilości informacji w domyślnej formie tabeli. | Nie | Nie | Tak |

W aplikacji Charts.io użytkownik nie będzie musiał przejmować się dostępem do swoich danych. Przejrzystość interfejsu pozwoli na swobodne poruszanie się po stronie i korzystanie ze wszystkich możliwości aplikacji za darmo. Na żaden wykres nie będzie nakładany znak wodny. Aplikacja pozwoli na swobodny dostęp użytkownikowi do wszystkich jego informacji dodanych podczas korzystania z serwisu pozwalając mu na edytowanie każdego jego elementu i zapisywanie do bazy. W ten sposób dane nawet sprzed wielu lat będą bezpieczne w zbiorze wszystkich rekordów osoby korzystającej ze strony. Istnieje wiele witryn pozwalających użytkownikowi na generowanie wykresów, mało jest jednak takich, które pozwalają na kolekcjonowanie danych i ponowne ich użycie w innych przypadkach. Charts.io będzie umożliwiać przechowywanie kluczowych danych do analizy przychodów i wydatków użytkownika pozwalając na przedstawienie ich graficznie automatycznie określając ich sumę, rozdzielając je na poszczególne kategorie, zestawiając je ze sobą oraz obrazując zmieniające się dane na przestrzeni dni, miesięcy czy nawet lat. Główną zaletą powyższego rozwiązania jest brak jakichkolwiek ograniczeń w kontekście tworzenia i zestawiania ze sobą informacji.

## **2. Wykorzystane technologie**

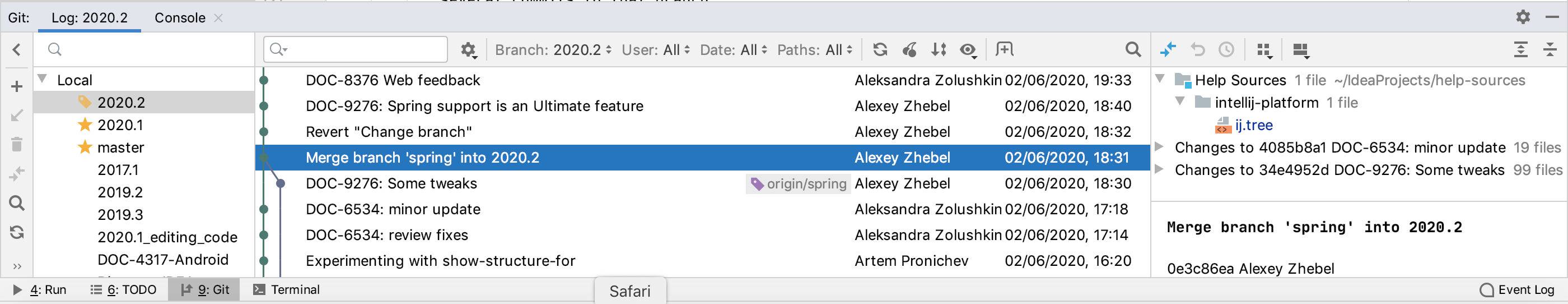
W niniejszym rozdziale zostanie przeprowadzona charakterystyka precyzyjnie dobranych technologii wraz z poparciem ich zastosowań przy tworzeniu strony internetowej Charts.io. Zaprezentowane zostaną nowoczesne rozwiązania, których celem jest dostarczenie starannie wykonanego produktu spełniającego wcześniej ustalone założenia funkcjonalne.

## **2.1 GIT**

GIT to rozproszony system kontroli wersji, który powstał w 2005 roku, jako narzędzie wspomagające rozwój jądra Linux. Oprogramowanie jest typu open source, działa na licencji GNU GPL w wersji 2[[3]](#footnote-3). Głównym zadaniem narzędzia GIT jest umożliwienie programistom zapisywanie wersji swojego kodu z dowolnego miejsca bez konieczności połączenia z siecią, pozwala również na korzystanie z wersji projektu na innej gałęzi (tzw. branch) bądź kodu znajdującego się w innym repozytorium. Jednym z głównych zadań oprogramowania jest ułatwienie pracy nad projektem wielu osobom równocześnie. GIT posiada szereg pożytecznych zastosowań, które mocno przydają się podczas tworzenia projektu również w przypadku pracy samodzielnej. Dzięki znajomości niewielkiej ilości komend każdy użytkownik oprogramowania jest wstanie zapisać swoje zmiany (tzw. commit), porównać je z poprzednią wersją, cofnąć je, wrócić do wersji projektu dostosowując go do swoich potrzeb a także stworzyć osobną gałąź rozwijając kod w zupełnie innym kierunku. Najbardziej przydatną funkcjonalnością podczas tworzenia aplikacji internetowej wydaję się możliwość przełączania się pomiędzy różnymi wersjami kodu, co pozwala na szybsze znalezienie i poprawę potencjalnych błędów.

Podczas tworzenia Charts.io zostanie wykorzystany GitHub, czyli serwis hostingowy przeznaczony do projektów programistycznych, który wykorzystuje rozproszony system kontroli wersji GIT[[4]](#footnote-4). Największą zaletą serwisu jest przechowywanie kodu źródłowego aplikacji użytkownika na własnym serwerze, co chroni przed utratą danych a jednocześnie umożliwia pobranie ich w dowolnym czasie na innym urządzeniu.

Do obsługi oprogramowania GIT zostanie użyte środowisko programistyczne IntelliJ posiadające szereg dodatkowych narzędzi pozwalających na szybszą i bardziej precyzyjną pracę z systemem kontroli wersji. Poniższy rysunek obrazuje przykładowe drzewo robocze w IntelliJ-u zawierające wszystkie gałęzie projektu oraz całą historie naniesionych zmian.



**Rysunek 2.1.** Drzewo robocze w aplikacji IntelliJ systemu kontroli wersji GIT

Źródło: Na podstawie

<https://www.jetbrains.com/help/idea/apply-changes-from-one-branch-to-another.html#rebase-branch>

## **2.2 IntelliJ**

Do tworzenia aplikacji Charts.io posłuży narzędzie IntelliJ IDEA, czyli zintegrowane środowisko programistyczne napisane w języku Java do tworzenia oraz utrzymania oprogramowania komputerowego. Narzędzie zostało wydane w 2001 roku przez firmę JetBrains zajmującą się tworzeniem podobnych produktów dla programistów piszących w różnych językach takich jak C++, C# czy też Python[[5]](#footnote-5). Głównym zadaniem IDE (Integrated development environment) jest utrzymanie wszystkich kluczowych technologii do implementacji oraz kompilacji projektu w jednym miejscu ułatwiając tym samym pracę nad aplikacją. Program oprócz podstawowej funkcjonalności implementacji oraz kompilacji kodu posiada również mnóstwo innych zalet takich jak wspomniany wcześniej już system kontroli wersji, integracje z Maven i Gradle (narzędzia odpowiadające za proces konwertujący kod źródłowy do postaci użytkowej lub wykonywalnej[[6]](#footnote-6)), niezwykle sprawnie działające automatyczne podpowiedzi a także szereg skrótów klawiszowych pozwalających na dynamiczne poruszanie się po kodzie. IntelliJ posiada również możliwość rozbudowy aplikacji za pomocą pluginów, zestaw narzędzi do refaktoryzacji, lokalną historię zmian, wbudowany edytor XML i wiele innych opcji sprawiających, że ten produkt od JetBrains uważany jest za jeden z najlepszych środowisk programistycznych przy tworzeniu aplikacji w języku Java oraz Kotlin.

Lista głównych powodów stojących za wyborem środowiska IntelliJ IDEA do tworzenia i rozwijania aplikacji Charts.io[[7]](#footnote-7):

* Integracja z narzędziem Maven, który posłuży do budowania projektu w Javie;
* Szereg opcji służących do inspekcji i potencjalnej poprawy kodu;
* Zintegrowany system kontroli wersji GIT;
* Lokalna historia zmian pozwalająca na szybki wgląd do poprzedniej wersji kodu;
* Automatyczne podpowiedzi podczas pisania kodu;
* Możliwości szybkiego poruszania się po elementach projektu;
* Ułatwiony dostęp do zarządzania wersjami języka programistycznego w ustawieniach IDE;
* Opcja korzystania z oprogramowania na urządzeniach z systemami Windows, MacOS oraz Linux;
* Możliwość szybkiego tworzenia dokumentacji do kodu (JavaDoc);
* Wygodne narzędzia do debugowania aplikacji;
* Inteligentna analiza błędów zwracająca informacje o kompilacji zakończonej porażką.

## **2.3 Java**

Językiem programowania, w którym napisania zostanie backendowa część aplikacji Charts.io jest Java. Pierwsza wersja języka Java pojawiła się w 1995 roku i aż do dzisiaj jest bardzo popularnym rozwiązaniem stosowanym i rozwijanym w setkach tysięcy projektów na całym świecie (według statystyk github.io). Powstała, jako wysokopoziomowy obiektowy język oparty na klasach zaprojektowana tak, aby zminimalizować ilość zależności implementacyjnych[[8]](#footnote-8). Java nadaje się do tworzenia wszelkiego rodzaju oprogramowania, każda aplikacja uruchamiana jest za pomocą maszyny wirtualnej, która tłumaczy kod do postaci zrozumiałej przez system operacyjny (bytecode)[[9]](#footnote-9). Największymi zaletami są niezależność od systemu operacyjnego, łatwość nauki wynikającej ze składni podobnej do języka C i C++ oraz stosunkowo krótki czas tworzenia aplikacji[[10]](#footnote-10). W przeciwieństwie do wymienionych powyżej języków Java posiada Garbage Collector, czyli swojego rodzaju program służący do usuwania z pamięci nieużywanych obiektów, co bez wątpienia wpływa na wygodę tworzenia rozbudowanych aplikacji a tym samym skraca jej czas.

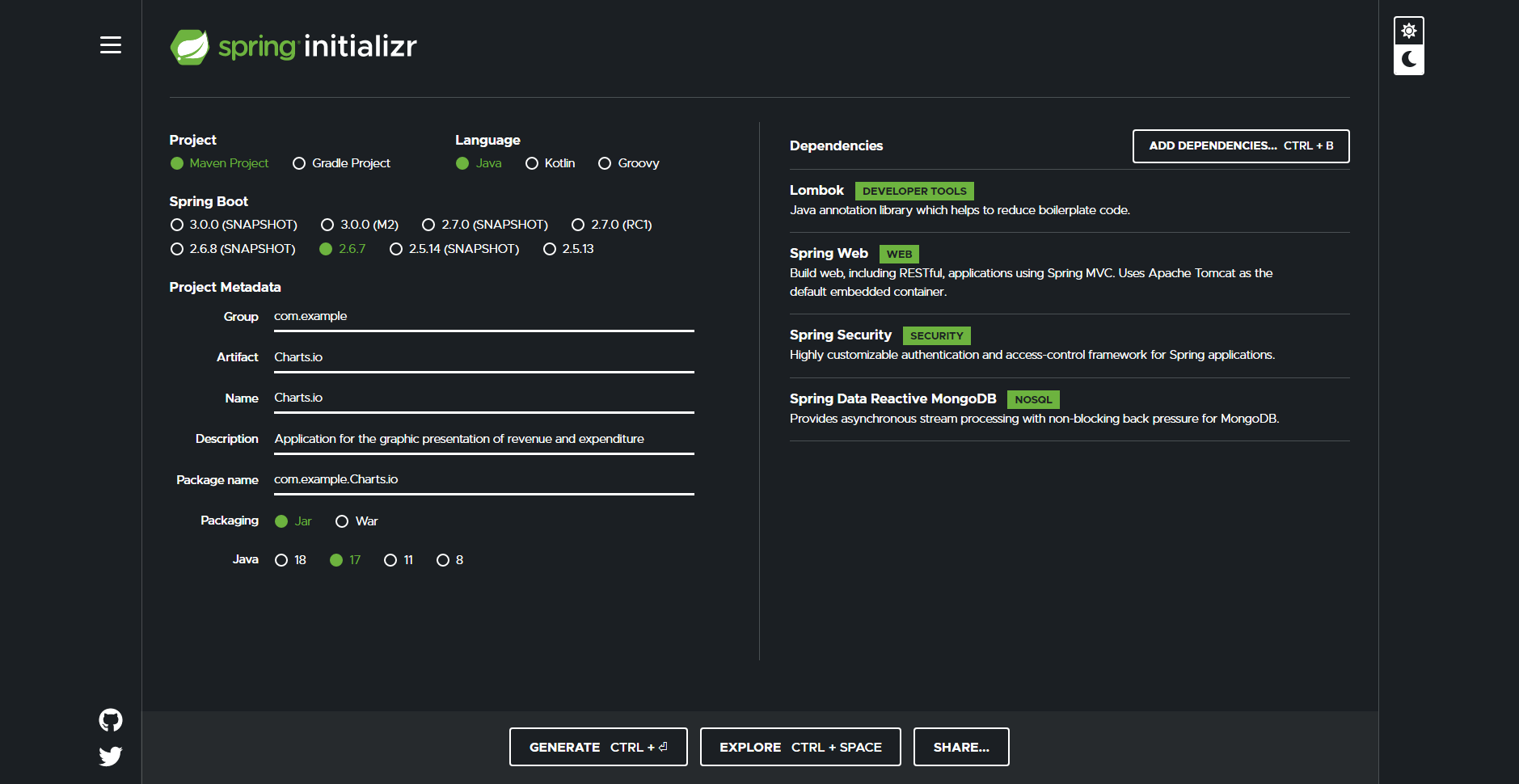
## **2.3.1 Spring Framework**

Framework lub inaczej platforma programistyczna to szkielet do budowy aplikacji, którego zadaniem jest dostarczenie niezbędnych bibliotek i komponentów definiujących strukturę i działanie danego kodu[[11]](#footnote-11). Struktura (ang. framework) stosowana jest podczas budowania aplikacji webowych jak i desktopowych a nawet mikrousług, może również posłużyć do wygenerowania całego ekosystemu.

Spring skupia się na prostocie, szybkości oraz produktywności, co czyni go najpopularniejszym frameworkiem dla języka Java. Głównym powodem, który stoi za wyborem framework-u Spring w aplikacji Charts.io jest szybkie i bezproblemowe tworzenie aplikacji internetowych, co wynika z braku większości kodu szablonowego i konfiguracji związanych z tworzeniem stron internetowych. Narzędzie dostarcza nowoczesny model programowania stron, który usprawnia tworzenie aplikacji HTML po stronie serwera, interfejsów API REST oraz dwukierunkowych systemów opartych na zdarzeniach (ang. event-based system).

## **2.3.2 Spring Boot**

Spring Boot jest swojego rodzaju rozszerzeniem frameworka Spring, który zakłada pozbycie się niepotrzebnej konfiguracji szablonowej wymaganej do skonfigurowania aplikacji. Zamiany zaoferowane przez oprogramowanie zapewniają szybszy i wydajniejszy ekosystem programistyczny. Cechami charakterystycznymi dla technologii Spring Boot są: wbudowany Tomcat (jeden z najpopularniejszych kontenerów webowych), zapewnienie uproszczonej konfiguracji projektu poprzez zależności startowe, automatyczne konfigurowanie biblioteki Spring i 3rd party, gotowe funkcje i adnotacje oraz absolutny brak wymagań dotyczących konfiguracji XML[[12]](#footnote-12). Głównym powodem, dla którego narzędzie zostanie wykorzystane podczas implementacji strony Charts.io jest prostota szybkiego tworzenia aplikacji webowych w języku Java. Konfiguracja projektu odbywa się za pomocą narzędzia Spring inititializr. Serwis umożliwia szybkie wygenerowanie nowego projektu za pomocą panelu na stronie start.spring.io, który pozwala na wybór narzędzia do budowania projektu (Maven lub Gradle), języka programowania oraz jego wersji, posiada również szeroki zakres zależności (ang. dependencies) w skład, których wchodzą narzędzia do obsługi bazy danych, kontroli bezpieczeństwa, sterowania siecią, znaleźć też można pożyteczne biblioteki jak Lombok, który pomaga zredukować standardowy kod oraz wiele innych. Poniżej znajduje się rysunek przedstawiający stronę spring initializr z przykładowo uzupełnionymi polami.



**Rysunek 2.2.** Strona tworzenia projektu Spring framework

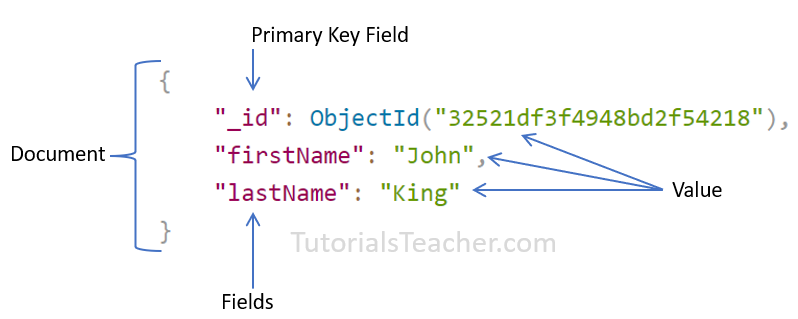
Źródło: Na podstawie https://start.spring.io

## **2.4 MongoDB**

MongoDB to nierelacyjna baza danych napisana w języku programowania C++, wydana i rozwijana przez MongoDB Inc. od 2009 roku[[13]](#footnote-13). Jako baza dokumentów wykorzystywana jest do budowania wysoce dostępnych i skalowalnych aplikacji internetowych[[14]](#footnote-14), elastyczne podejście do schematów sprawiło, że MongoDB zyskało ogromne zainteresowanie wśród programistów. Do głównych zalet należy możliwość korzystania z bazy we wszystkich głównych językach programowania, umożliwia natychmiastowe rozpoczęcie pracy nad tworzeniem aplikacji bez martwienia się o konfiguracje, co ma istotny wpływ na całość projektu.

W przeciwieństwie do relacyjnych baz danych wykorzystujących SQL (Structured Query Language), które przechowują dane w tabelach składających się z kolumn i wierszy połączonych ze sobą relacjami MongoDB magazynuje każdy rekord w bazie, jako dokument w formacie BSON (binarna reprezentacja danych) aplikacja może następnie pobrać te informacje w formacie JSON[[15]](#footnote-15). Bazę danych cechuje duża elastyczność pozwalająca na dynamiczne zmiany w strukturze dokumentów, dodatkowo w jednym dokumencie mogą być osadzone inne dokumenty, rolę kolumn odgrywają pola, które mogą być indeksowane zwiększając tym samym wydajność wyszukiwania. Dzięki architekturze skalowalnej MongoDB umożliwia współpracę wielu małych maszyn do obsługi ogromnych ilości danych poprzez tworzenie szybkich systemów[[16]](#footnote-16).

Powodem, dla których aplikacja Charts.io będzie korzystać z nierelacyjnej bazy danych MongoDB jest szereg zalet pozwalających na szybkie i sprawne operowanie danymi. Jedną z tych zalet jest format dokumentu JSON, który można pobrać bezpośrednio z bazy, pozwalający na szybką analizę rekordów dzięki swojej czytelnej formie. Największą zaletą narzędzia z perspektywy programisty jest kontrola nad schematem bazy danych, deweloperzy mają możliwość dostosować schemat bazy wraz z rozwojem aplikacji. Poniższy rysunek obrazuje przykładowy dokument bazy MongoDB w formacie JSON.



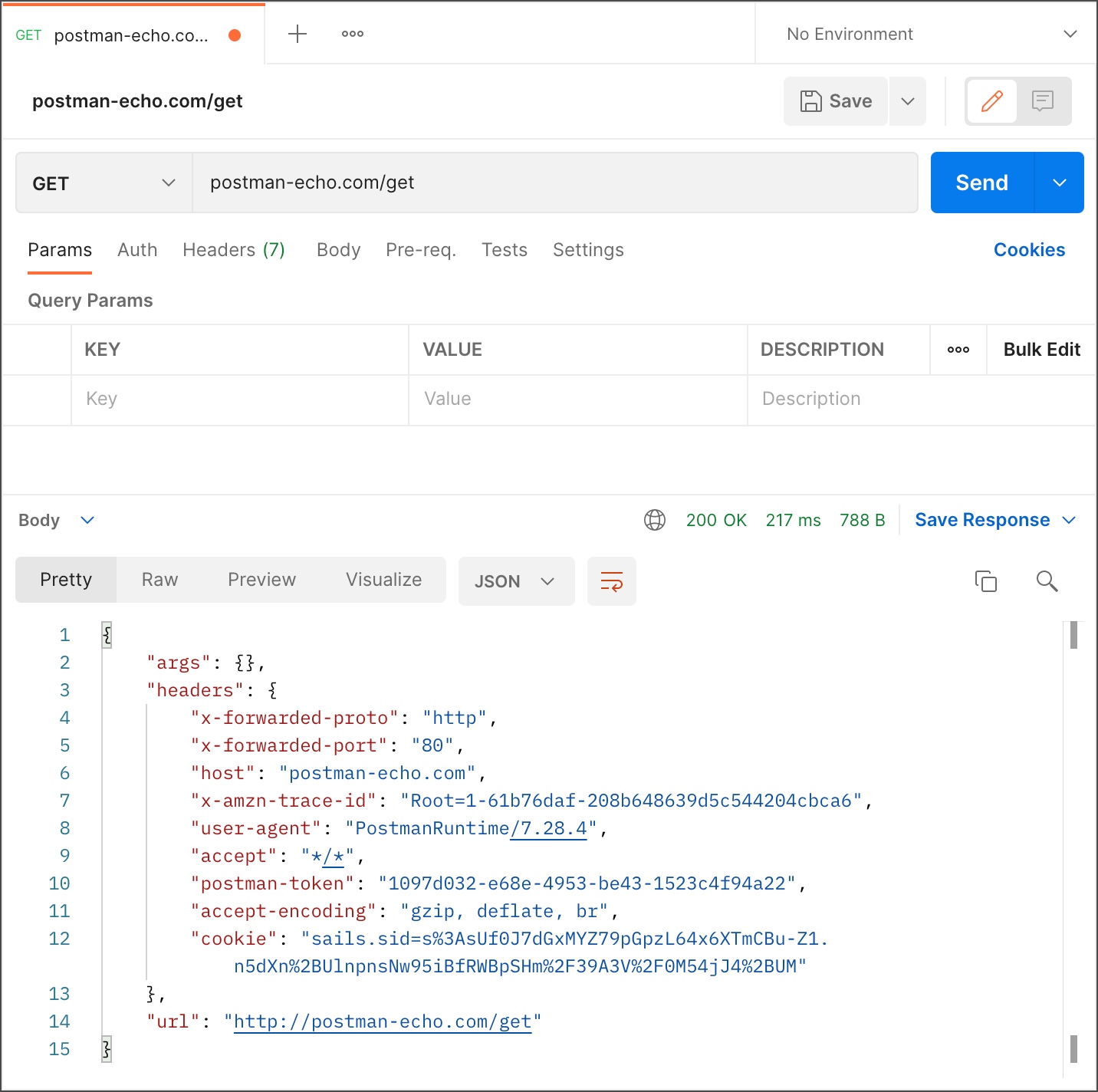
**Rysunek 2.3.** Dokument nierelacyjnej bazy danych MongoDB w formacie JSON

Źródło: Na podstawie <https://www.tutorialsteacher.com/mongodb/documents>

## **2.5 Postman**

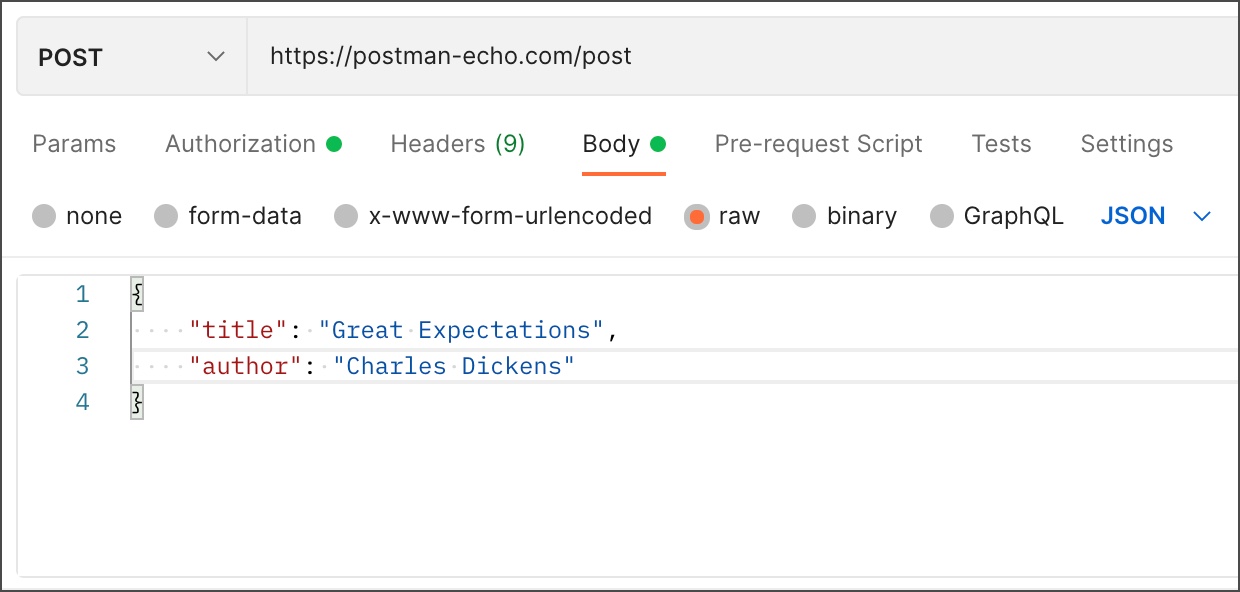
Postman to oprogramowanie, którym posługują się deweloperzy podczas projektowania, budowania, testowania i iterowania API (Application Programming Interface)[[17]](#footnote-17). API, czyli interfejs programowania aplikacji to zestaw reguł, które umożliwiają różnym komponentom oprogramowania komunikowanie się ze sobą za pomocą zestawu definicji i protokołów[[18]](#footnote-18). Głównym zastosowaniem aplikacji podczas tworzenia strony Charts.io będzie możliwość testowania żądań wysyłanych do serwera za pomocą graficznego interfejsu użytkownika weryfikując poprawność wcześniej napisanego kodu. Backendowa część aplikacji będzie odpowiedzialna za wygenerowanie interfejsu REST API (Reprezentacyjny transfer stanu) definiującego szereg funkcji takich jak GET, POST, PUT, DELETE itp., które pozwalają użytkownikom na uzyskanie odpowiedniego dostępu do wybranych danych serwera. Główną cechą REST API jest bezstanowość, która oznacza, że serwery nie zapisują danych klienta między żądaniami, owe zapytania przypominają mocno adresy URL jednak zwracają jedynie zwykłe dane[[19]](#footnote-19). Przeciętna przeglądarka nie jest wstanie poradzić sobie z przetestowaniem wszystkich metod pozwalających na korzystanie z REST API, do oceny poprawności wygenerowanych żądań pod kontem funkcjonalności posłuży Postman.

Oprogramowanie posiada szereg narzędzi umożliwiających dodawanie parametrów adresu, uzgodnienie szczegółów dotyczących autoryzacji, dodanie nagłówków (ang. headers) oraz danych wysyłanych przez klienta (Postman daje możliwość wyboru formatu, w którym mają zostać wysłane informacje), całość ułatwia projektowanie głównych funkcjonalności aplikacji internetowych oraz umożliwia kompleksową analizę wszystkich przypadków użycia. Rysunki znajdujący się poniżej prezentuje program Postman z przykładowymi zapytaniami GET i POST.



**Rysunek 2.4.** Aplikacja Postman z zapytania GET

Źródło: Na podstawie https://learning.postman.com/docs/getting-started/sending-the-first-request/

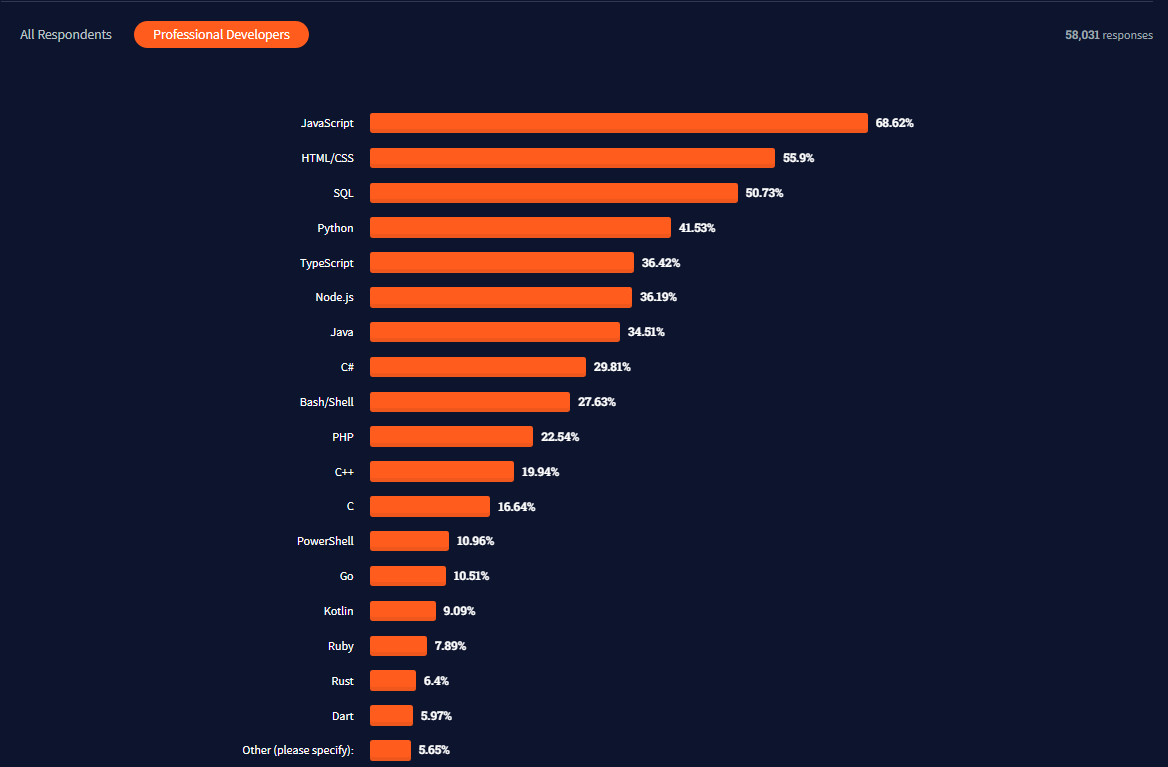


**Rysunek 2.5.** Aplikacja Postman z zapytania POST

Źródło: Na podstawie <https://learning.postman.com/docs/sending-requests/requests/>

## **2.6 JavaScript**

JavaScript (JS) to najpopularniejszy język programowania i podstawowa technologia World Wide Web, która jest wykorzystywana przez aż 97% stron internetowych po stronie klienta[[20]](#footnote-20). Język służy głównie do tworzenia dynamicznych i interaktywnych treści internetowych. Największą zaletą JS jest jej wszechstronność na wielu płaszczyznach, pozwala na budowanie skomplikowanych aplikacji i programów takich jak statyczne i dynamiczne strony internetowe, narzędzia do sterowania sprzętem a nawet serwery internetowe (ang. web server)[[21]](#footnote-21). JavaScript zostanie wykorzystana w aplikacji Charts.io po stronie wizualnej tworząc UI (user interface) i UX (user experience) specjalnie dostosowane dla potencjalnego użytkownika korzystającego ze strony. Jej największą zaletą są liczne frameworki i biblioteki takie jak Angular lub React pozwalające na zaprojektowanie dynamicznej i czytelnej strony posiadającej ciekawe animacje, często wykorzystujące komponenty responsywne, czyli takie, które dostosowują swoją szerokość i układ elementów do rozdzielczości urządzenia, na którym są wyświetlane. Poniższy rysunek przedstawia najczęściej używane języki programowania na podstawie głosów programistów z całego świata korzystających ze strony stackoverflow.com, ankieta oraz jej wyniki pochodzą z 2021 roku.

****

**Rysunek 2.6.** Ranking najpopularniejszych języków programowania z 2021 roku

Źródło: Na podstawie https://insights.stackoverflow.com/survey/2021

## **2.6.1 React**

React to jedna z najpopularniejszych otwartych i bezpłatnych bibliotek JavaScript służących do projektowania i tworzenia np. interfejsów użytkownika, statycznych i dynamicznych stron internetowych, aplikacji jednostronicowych oraz mobilnych. Jego głównym zadaniem jest zarządzanie i renderowanie stanu do modelu DOM (Document Object Model), czyli wieloplatformowego interfejsu reprezentującego dokument, jako strukturę drzewa[[22]](#footnote-22). Tworzenie aplikacji z wykorzystaniem React-a wymaga użycia dodatkowych bibliotek i funkcji po stronie klienta[[23]](#footnote-23). Największymi zaletami biblioteki są szybkość tworzenia i wdrażania aplikacji oraz wysoka wydajność. Powodem, dla którego React posłuży do tworzenia aplikacji Charts.io jest elastyczność, która umożliwia łatwe utrzymanie kodu dzięki swojej strukturze w przeciwieństwie do innych frameworków frontendowych[[24]](#footnote-24) pozwalając tym samym odnaleźć się w projekcie szczególnie podczas rozwijania lub dodawania nowych funkcjonalności, co wpływa bezpośrednio na czas potrzebny do stworzenia całego projektu.

**2.6.2 React Google Charts**

Główną funkcjonalnością aplikacji Charts.io będzie możliwość generowania wykresów dla przychodów i wydatków użytkownika, narzędziem odpowiedzialnym za tą część wizualną strony będzie React Google Charts, czyli darmowa biblioteka JavaScript umożliwiająca wizualizacje informacji poprzez prezentowanie ich w sposób graficzny. Do największych zalet technologii należy przede wszystkim duża ilość komponentów pozwalająca na przedstawianie danych w różnych zestawieniach ułatwiając tym samym analizę i interpretacje rekordów. Kilkadziesiąt szablonów umożliwia zobrazowanie danych na podstawie kategorii, wartości lub czasu, który jest do niej przypisany, niektóre wykresy pozwalają nawet na interakcje z użytkownikiem pozwalając mu na wyszukanie rekordu po nazwie lub określenie zakresu interesujących go wartości. Czytelna i mocno rozbudowana dokumentacja sprawia, że dodawanie nowego wykresu do projektu jest wyjątkowo łatwe i szybkie.

## **2.6.3 React Bootstrap**

React Bootstrap to biblioteka JavaScript wykorzystywana w projektach React-owych, posiada dużą ilość komponentów pozwalających na zbudowanie pełnego widoku strony internetowej. Zawiera takie elementy jak przyciski, tabele, wyszukiwarki tekstu, checkbox-y oraz wiele innych, wszystko, aby ułatwić korzystanie z pełnej funkcjonalności aplikacji tworząc czytelny interfejs użytkownika. Ogromną zaletą jest bardzo prosta implementacja biblioteki, która wymaga jedynie zaimportowania odpowiednich komponentów. Wszystkie elementy React Bootstrap są responsywne, co sprawia, że strona jest czytelna na każdym urządzeniu. Korzystanie z komponentów biblioteki jest wyjątkowo szybkie, strona react-bootstrap.github.io pozwala na zaimportowanie kodu oszczędzając tym samym czas deweloperom na pisanie oraz dostosowanie komponentów od zera. Pomimo dużej popularności wynikającej z łatwości dodawania i operowania elementami, React Bootstrap wciąż wykorzystywany jest do tworzenie nowych stron internetowych wyróżniających się swoim wyglądem dzięki licznym komponentom oraz nieograniczonej wariancji w wyborze stylów i animacji.

## **2.6.4 Pozostałe biblioteki JavaScript**

W tabeli zamieszczono wszystkie najważniejsze bibliotek JS, które zostaną wykorzystane przy tworzeniu aplikacji Charts.io wraz z ich zastosowaniem.

**Tabela 2.1.** Pozostałe biblioteki JavaScript

Źródło: opracowanie własne

|  |  |  |
| --- | --- | --- |
| L.p. | Nazwa biblioteki JavaScript | Zastosowanie |
| 1. | React | Zarządzanie i renderowanie stanu wieloplatformowego interfejsu reprezentującego dokument, jako strukturę drzewa. |
| 2. | React Google Charts | Udostępnianie komponentów generujących wykresy dla danych użytkownika, pozwalająca na dostosowanie ich wartości, tytułu oraz kompozycji. |
| 3. | React Bootstrap | Dostęp do dużej ilości elementów pozwalających na stworzenie podstawowego interfejsu użytkownika, jako graficzna reprezentacja logiki działającego kodu. |
| 4. | React Fontawesome | Biblioteka posiadająca dużą ilość ikon, których zadaniem będzie ułatwienie użytkownikowi korzystanie ze strony poprzez umieszczone symbole obrazujące zastosowanie danego komponentu. |
| 5. | React Redux | Redux służy, jako magazyn danych dla dowolnej warstwy interfejsu użytkownika zapewniając proste mechanizmy subskrypcji, których zadaniem jest radzenie sobie z dużą ilością stanu aplikacji zwłaszcza, kiedy potrzebne są w wielu miejscach aplikacji[[25]](#footnote-25). |

## **3. Zarys projektu**

Niniejszy rozdział poświęcony jest zarysowi projektu aplikacji internetowej Charts.io. Zarys składa się z wymagań funkcjonalnych i niefunkcjonalnych, przypadków użycia w formie diagramu oraz scenariusza, schematu bazy danych, diagramu klas oraz projektu interfejsu użytkownika. Do graficznego przedstawienia przypadków użycia, schematu bazy danych oraz projektu interfejsu zostało wykorzystane wieloplatformowe oprogramowanie do rysowania wykresów draw.io oraz Hackolade pozwalająca na tworzenie wszelkiego rodzaju dokładnych i rozbudowanych diagramów.

## **3.1 Wymagania**

W celu zaprojektowania i stworzenia w pełni spójnego oprogramowania konieczne jest wcześniejsze określenie wszystkich wymagań, które musi spełniać aplikacja. Wymagania można podzielić na funkcjonalne, których zadaniem jest określenie funkcjonalności efektu końcowego projektu oraz niefunkcjonalne, czyli takie, które stanowią, o jakości oraz poprawności działania aplikacji.

## **3.1.1 Wymagania funkcjonalne**

* Rejestracja.
* Logowanie.
* Wyświetlanie tabeli wszystkich przychodów.
* Wyświetlanie tabeli wszystkich wydatków.
* Sortowanie wszystkich przychodów w tabeli według daty.
* Sortowanie wszystkich wydatków w tabeli według daty.
* Wyszukiwanie przychodów w tabeli po nazwie.
* Wyszukiwanie wydatków w tabeli po nazwie.
* Dodanie przychodu.
* Dodanie wydatku.
* Edytowanie przychodu.
* Edytowanie wydatku.
* Usuwanie przychodu.
* Usuwanie wydatku.
* Wyświetlanie wszystkich wykresów wygenerowanych automatycznie dla przychodów i wydatków użytkownika.
* Wyświetlanie wykresów wygenerowanych automatycznie zgodnych z wybraną kategorią (przychody, wydatki, przychody i wydatki, wykresy kołowe, wykresy kolumnowe).
* Generowanie nowego sparametryzowanego wykresu.

## **3.1.2 Wymagania niefunkcjonalne**

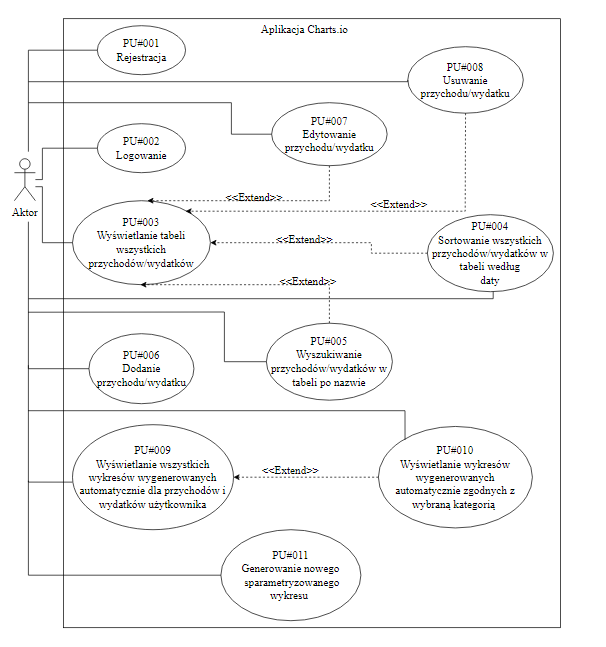
* Wsparcie dla przeglądarki Chrome nie starszej niż wersja 67 i Mozilla od wersji 60.
* Obsługa urządzeń z różnymi rozmiarami ekranów.
* Pełna dostępność do strony.
* Przejrzystość interfejsu użytkownika.
* Ciemny motyw aplikacji.
* Oznaczenia funkcjonalnych elementów strony ikonami.
* Łatwość użycia.

## **3.2 Przypadki użycia**

Przypadkiem użycia (ang. use case) nazywamy zbiór scenariuszy spełniających wcześniej określone wymagania, ich głównym zadaniem jest przedstawienie podstawowych funkcjonalności aplikacji oraz własności oprogramowania. Dla każdego przypadku użycia można określić przebieg główny składający się z czynności prowadzących do sukcesu będącym oczekiwanym warunkiem końcowym oraz przebieg alternatywny zaistniały w momencie niepoprawnego korzystania z funkcjonalności aplikacji najczęściej prowadzący do przerwania przypadku użycia, powrotu do poprzedniego etapu korzystania z oprogramowania lub zakończenia połowicznym sukcesem, który może nie spełniać wszystkich oczekiwań końcowych.

## **3.2.1 Diagram przypadków użycia**

Diagram przypadków użycia obrazuje interakcje między systemem a aktorem przedstawiając graficznie przebieg korzystania z odrębnych elementów aplikacji. Poniższy rysunek przedstawia zależności między aktorem a poszczególnymi przypadkami użycia.



**Rysunek 3.1.** Diagram przypadków użycia

Źródło: opracowanie własne

**3.2.2 Scenariusze przypadków użycia**

Poniższe tabele przedstawiają poszczególne scenariusze przypadków użycia, które składają się z nazwy PU, jego opisu, warunków początkowych i końcowych oraz przebiegów głównych jak i alternatywnych. Zadaniem scenariuszy przypadków użycia jest przeanalizowanie dokładnie każdej możliwej ścieżki korzystania z opracowywanej aplikacji oraz próba przewidzenia potencjalnych problemów implementacyjnych.

**Tabela 3.1.** PU#001 Rejestracja

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#001 |  |
| Nazwa | Rejestracja |
| Aktor | Użytkownik |
| Opis | Użytkownik chce się zarejestrować do aplikacji internetowej Charts.io. |
| Warunki początkowe | * Użytkownik znajduje się w aplikacji internetowej Charts.io. |
| Przebieg główny | 1. Użytkownik wybiera zakładkę rejestracji. 2. Użytkownik podaje unikalny login, email oraz hasło. 3. Użytkownik akceptuje podane informacje. 4. Użytkownik zostaje przekierowany do zakładki logowania. |
| Przebieg alternatywny | 3a. Jeśli podany login lub hasło są już zajęte wyświetlona zostanie informacja o błędnej rejestracji.  3b. Jeśli podane hasło jest krótsze niż 6 znaków wyświetlona zostanie informacja o zbyt krótkim haśle. |
| Warunki końcowe | * Nowe konto zostaje utworzone. * Użytkownik zostaje przekierowany do strony logowania. |

**Tabela 3.2.** PU#002 Logowanie

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#002 |  |
| Nazwa | Logowanie |
| Aktor | Użytkownik |
| Opis | Użytkownik chce zalogować się na swoje konto. |
| Warunki początkowe | * Użytkownik jest zarejestrowany. * Użytkownik znajduje się w zakładce logowania. |
| Przebieg główny | 1. Użytkownik podaje login i hasło. 2. Użytkownik zostaje przekierowany na główną stronę aplikacji. |
| Przebieg alternatywny | 1a. Jeśli podany login lub hasło są błędne wyświetlona zostaje informacja o błędnej próbie zalogowania się. |
| Warunki końcowe | * Użytkownik zostanie przekierowany na stronę główną aplikacji. |

**Tabela 3.3**. PU#003 Wyświetlanie tabeli wszystkich przychodów/wydatków

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#003 |  |
| Nazwa | Wyświetlanie tabeli wszystkich przychodów/wydatków. |
| Aktor | Użytkownik |
| Opis | Użytkownik chce wyświetlić tabele wszystkich swoich przychodów bądź wydatków. |
| Warunki początkowe | * Użytkownik znajduje się w aplikacji internetowej Charts.io, jako zalogowany. |
| Przebieg główny | 1. Użytkownik wybiera zakładkę wyświetlającą tabele przychodów/wydatków. 2. Użytkownik zostaje przekierowany do zakładki z tabelą wszystkich swoich przychodów/wydatków. |
| Przebieg alternatywny |  |
| Warunki końcowe | * Użytkownik zostaje przekierowany do wybranej zakładki zawierającej tabele wszystkich przychodów lub wydatków. |

**Tabela 3.4.** PU#004 Sortowanie wszystkich przychodów/wydatków w tabeli według daty

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#004 |  |
| Nazwa | Sortowanie wszystkich przychodów/wydatków w tabeli według daty. |
| Aktor | Użytkownik |
| Opis | Użytkownik chce posortować swoje przychody/wydatki po dacie. |
| Warunki początkowe | * Użytkownik znajduje się w zakładce wyświetlającej tabele wszystkich przychodów/wydatków. |
| Przebieg główny | 1. Użytkownik naciska symbol trójkąta znajdujący się przy kolumnie z datami. 2. Użytkownikowi wyświetlana jest tabela przychodów/wydatków posortowana chronologicznie. 3. Użytkownik ponownie naciska symbol trójkąta znajdujący się przy kolumnie z datami. 4. Użytkownikowi wyświetlana jest tabela przychodów/wydatków posortowana w odwrotnej kolejności. |
| Przebieg alternatywny | 1a. Jeśli użytkownik nie posiada dodanych żadnych przychodów lub wydatków wartości nie zostaną posortowane (tabela jest pusta). |
| Warunki końcowe | * Użytkownikowi zostaje wyświetlona tabela z odpowiednio posortowanymi wartościami. |

**Tabela 3.5**. PU#005 Wyszukiwanie przychodów/wydatków w tabeli po nazwie

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#005 |  |
| Nazwa | Wyszukiwanie przychodów/wydatków po nazwie. |
| Aktor | Użytkownik |
| Opis | Użytkownik chce wyszukać swoje przychody/wydatki na podstawie nazwy. |
| Warunki początkowe | * Użytkownik znajduje się w zakładce wyświetlającej tabele wszystkich przychodów/wydatków. |
| Przebieg główny | 1. W polu znajdującym się nad tabelą użytkownik wpisuje początek lub pełną nazwę przychodu/wydatku, którą chce wyszukać. 2. Użytkownik zatwierdza podaną nazwę klikając symbol lupy w celu zatwierdzenia wyszukiwania. 3. Użytkownikowi wyświetla się tabela przychodów/wydatków na podstawie nazwy podanej w pole wyszukiwania. |
| Przebieg alternatywny | 3a. Jeśli użytkownik nie posiada dodanych żadnych przychodów lub wydatków wartości nie zostaną wyszukane (tabela jest pusta).  3b. Jeśli przychód bądź wydatek o podanej nazwie nie istnieje zwrócona zostanie informacja o niepowodzeniu wyszukiwania.  3c. Jeśli podana nazwa przychodu bądź wydatku została wpisana błędnie zwrócona zostanie informacja o niepowodzeniu wyszukiwania. |
| Warunki końcowe | * Użytkownikowi zostaje wyświetlona tabela przychodów/wydatków zgodnych z podaną nazwą. |

**Tabela 3.6.** PU#006 Dodanie przychodu/wydatku

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#006 |  |
| Nazwa | Dodanie przychodu/wydatku. |
| Aktor | Użytkownik |
| Opis | Użytkownik chce dodać przychód/wydatek. |
| Warunki początkowe | * Użytkownik znajduje się w aplikacji internetowej Charts.io, jako zalogowany. |
| Przebieg główny | 1. Użytkownik przechodzi do zakładki dodawania przychodu/wydatku. 2. Użytkownik wpisuje nazwę przychodu/wydatku. 3. Użytkownik wypełnia pole określające wartość przychodu/wydatku. 4. Użytkownik dodaje datę przychodu/wydatku. 5. Użytkownik akceptuje dodanie przychodu/wydatku. |
| Przebieg alternatywny | 5a. Jeżeli nazwa przychodu/wydatku pozostanie pusta zwrócona zostanie informacja o wymaganym uzupełnieniu pola.  5b. Jeżeli wartość przychodu/wydatku pozostanie pusta zwrócona zostanie informacja o wymaganym uzupełnieniu pola.  5c. Jeżeli pole wartości przychodu/wydatku zostanie uzupełnione tekstem lub liczbami ujemnymi zwrócona zostanie informacja o błędnej wartości.  5d. Jeśli data przychodu/wydatku pozostanie pusta zwrócona zostanie informacja o wymaganym uzupełnieniu pola. |
| Warunki końcowe | * Użytkownikowi zostaje wyświetlona informacja o poprawnym dodaniu przychodu/wydatku. * Nowy rekord przychodu/wydatku jest dodany do bazy. |

**Tabela 3.7.** PU#007 Edytowanie przychodu/wydatku

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#007 |  |
| Nazwa | Edytowanie przychodu/wydatku. |
| Aktor | Użytkownik |
| Opis | Użytkownik chce edytować przychód/wydatek. |
| Warunki początkowe | * Użytkownik znajduje się w zakładce wyświetlającej tabele wszystkich przychodów/wydatków. * Użytkownik posiada dodany przychód/wydatek. |
| Przebieg główny | 1. Użytkownik wybiera ikonę edycji przychodu/wydatku znajdującą się przy rekordzie w tablicy. 2. Wyświetlone zostaje pole edycji wybranego przychodu/wydatku. 3. Użytkownik zmienia nazwę, wartość lub datę przychodu/wydatku. 4. Użytkownik akceptuje edycje danego rekordu. 5. Użytkownik zostaje przekierowany do zakładki wyświetlającej tabele wszystkich przychodów/wydatków. |
| Przebieg alternatywny | 5a. Jeśli nazwa przychodu/wydatku zostanie usunięta wyświetli się informacja o wymaganym uzupełnieniu pola.  5b. Jeśli wartość przychodu/wydatku zostanie usunięta wyświetli się informacja o wymaganym uzupełnieniu pola.  5c. Jeśli pole wartości przychodu/wydatku zostanie zmienione na tekst lub liczbę ujemną wyświetli się informacja o błędnej wartości.  5d. Jeśli data przychodu/wydatku pozostanie pusta zwrócona zostanie informacja o wymaganym uzupełnieniu pola. |
| Warunki końcowe | * Użytkownikowi zostaje wyświetlona informacja o poprawnej edycji danych. * Użytkownik zostaje przekierowany do zakładki wyświetlającej tabele wszystkich przychodów/wydatków. * Edytowany rekord jest zmieniony w bazie. |

**Tabela 3.8.** PU#008 Usuwanie przychodu/wydatku

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#008 |  |
| Nazwa | Usuwanie przychodu/wydatku. |
| Aktor | Użytkownik |
| Opis | Użytkownik chce usunąć przychód/wydatek. |
| Warunki początkowe | * Użytkownik znajduje się w zakładce wyświetlającej tabele wszystkich przychodów/wydatków. * Użytkownik posiada dodany przychód/wydatek. |
| Przebieg główny | 1. Użytkownik wybiera ikonę usunięcia przychodu/wydatku znajdującą się przy rekordzie w tablicy. 2. Wyświetlona zostaje informacja o pozytywnym usunięciu rekordu z tabeli. |
| Przebieg alternatywny |  |
| Warunki końcowe | * Wybrany rekord zostaje usunięty z tabeli przychodów/wydatków. * Rekord jest usunięty z bazy. |

**Tabela 3.9**. PU#009 Wyświetlanie wszystkich wykresów wygenerowanych automatycznie dla przychodów i wydatków

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#009 |  |
| Nazwa | Wyświetlanie wszystkich wykresów wygenerowanych automatycznie dla przychodów i wydatków. |
| Aktor | Użytkownik |
| Opis | Użytkownik chce wyświetlić wykresy wygenerowane automatycznie dla swoich przychodów i wydatków. |
| Warunki początkowe | * Użytkownik znajduje się w aplikacji internetowej Charts.io, jako zalogowany. |
| Przebieg główny | 1. Użytkownik przechodzi do zakładki z wygenerowanymi wykresami. 2. Wyświetlona zostaje lista wygenerowanych wykresów dla dodanych przychodów i wydatków. |
| Przebieg alternatywny | 2a. Jeśli użytkownik nie posiada dodanych żadnych przychodów lub wydatków wykresy nie zostaną wyświetlone. |
| Warunki końcowe | * Użytkownikowi zostają wyświetlone wykresy na podstawie dodanych przychodów i wydatków. |

**Tabela 3.10.** PU#010 Wyświetlanie wykresów wygenerowanych automatycznie zgodnych z wybraną kategorią

Źródło: opracowanie własne

|  |  |
| --- | --- |
| PU#010 |  |
| Nazwa | Wyświetlanie wykresów wygenerowanych automatycznie zgodnych z wybraną kategorią. |
| Aktor | Użytkownik |
| Opis | Użytkownik chce wyświetlić wykresy wygenerowane automatycznie z danej kategorii. |
| Warunki początkowe | * Użytkownik znajduje się w zakładce wyświetlającej wygenerowane automatycznie wykresy dla przychodów i wydatków. |
| Przebieg główny | 1. Użytkownik rozwija listę kategorii wyświetlanych wykresów. 2. Użytkownik wybiera jedną z interesujących go kategorii. 3. Wyświetlane zostają wykresy zgodne z wybraną kategorią. |
| Przebieg alternatywny | 3a. Jeśli użytkownik nie posiada dodanych żadnych przychodów lub wydatków wykresy nie zostaną wyświetlone.  3b. Jeśli użytkownik nie posiada odpowiednich danych spełniających konkretną kategorie wykresy z owej kategorii nie zostaną wyświetlone. |
| Warunki końcowe | * Użytkownikowi zostają wyświetlone wykresy na podstawie dodanych przychodów i wydatków zgodne z wybraną kategorią wyświetlania. |

**Tabela 3.11.** PU#011 Generowanie nowego sparametryzowanego wykresu

Źródło: opracowanie własne

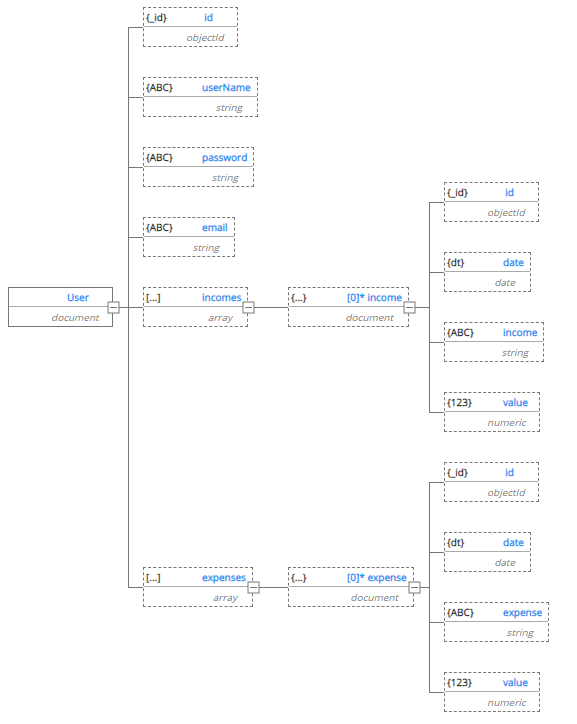
|  |  |
| --- | --- |
| PU#011 |  |
| Nazwa | Generowanie nowego sparametryzowanego wykresu. |
| Aktor | Użytkownik |
| Opis | Użytkownik chce stworzyć własny wykres na podstawie przychodów i wydatków. |
| Warunki początkowe | * Użytkownik znajduje się w aplikacji internetowej Charts.io, jako zalogowany. |
| Przebieg główny | 1. Użytkownik przechodzi do zakładki tworzenia nowego wykresu. 2. (Opcjonalne) Użytkownik uzupełnia nazwę wykresu. 3. Użytkownik ustawia datę początkową. 4. Użytkownik ustawia datę końcową. 5. Użytkownik wybiera rodzaj wykresu. 6. Użytkownik wybiera rodzaj wyświetlanych wartości na wykresie. 7. Użytkownik zatwierdza podane informacje. 8. Wyświetlany zostaje wykres dla wybranych ustawień. |
| Przebieg alternatywny | 7a. Jeśli nie zostanie podana data początkowa wyświetli się informacja o wymaganym uzupełnieniu pola.  7b. Jeśli nie zostanie podana data końcowa wyświetli się informacja o wymaganym uzupełnieniu pola.  7c. Jeśli nie zostanie wybrany rodzaj wykresu wyświetli się informacja o wymaganym uzupełnieniu pola.  7d. Jeśli data początkowa okaże się późniejsza od daty końcowej wyświetlona zostanie informacja o braku danych.  7e. Jeśli nie zostanie wybrany rodzaj wartości wyświetli się informacja o wymaganym uzupełnieniu pola.  7f. Jeśli użytkownik nie posiada odpowiednich danych spełniających wybrane ustawienia wyświetlona zostanie informacja o braku danych. |
| Warunki końcowe | * Użytkownikowi zostaje wyświetlony wykres wygenerowany na podstawie wybranych ustawień i kategorii |

## **3.3 Baza danych**

Aplikacja Charts.io będzie korzystała z nierelacyjnej bazy danych MongoDB, poniżej przedstawione został schematy bazy danych, których zadaniem jest zobrazowanie występujących w niej pól oraz zależności wynikających z obsługi wielu dokumentów.

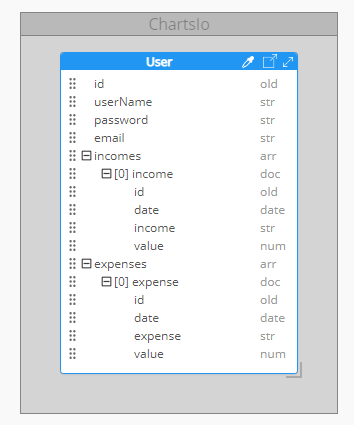
## **3.3.1 Schemat bazy danych**

Poniższe rysunki przedstawiają schemat bazy MongoDB dla aplikacji internetowej Charts.io wykonane za pomocą oprogramowania Hackolade, służącego do graficznego opracowywania schematów i modeli aplikacji posługujących się relacyjnymi oraz nierelacyjnymi bazami danych. Pierwszy z nich przypomina mocno format pliku JSON, drugi natomiast jest bardziej rozbity, przedstawia interakcje wszystkich dokumentów, przypomina typowy schemat dla relacyjnych baz danych. Dokument składa się z następujących pól: ‘id’ generowane automatycznie, ‘userName’ będący nazwą użytkownika, ‘password’, czyli hasło, pole ‘email’, które jest adresem email użytkownika oraz dwóch list ‘incomes’ i ‘expenses’ (przychody i wydatki), które stanowią własne dokumenty. Pola w dokumentach przychodów i wydatków to ‘id’, które ułatwia sortowanie oraz dostęp do określonego rekordu, ‘date’ będący datą przychodu lub wydatku, ‘income’ lub ‘expense’ w zależności od dokumentu, (aby odróżnić je podczas implementacji), które służą, jako nazwa przychodu lub wydatku w bazie oraz ostatnie pole ‘value’, które stanowi o wartości.



**Rysunek 3.2.** Schemat nierelacyjnej bazy danych MongoDB dla aplikacji Charts.io nr 1

Źródło: opracowanie własne

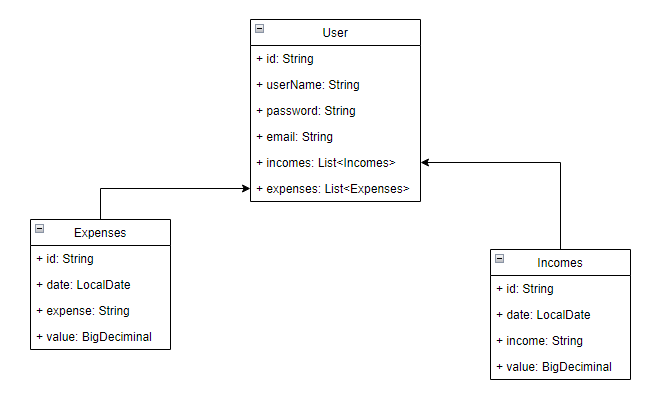


**Rysunek 3.3.** Schemat nierelacyjnej bazy danych MongoDB dla aplikacji Charts.io nr 2

Źródło: opracowanie własne

## **3.4 Diagram klas**

Poniższa grafika przedstawia diagram klas służący do zarządzania bazą danych oraz kontrolą rekordów w kodzie odpowiedzialną za podstawowe metody obsługi logiki biznesowej aplikacji. Diagram został wykonany za pomocą oprogramowania draw.io, który swoimi precyzyjnymi narzędziami pozwala na generowanie rozbudowanych i szczegółowych diagramów.



**Rysunek 3.4.** Diagram klas dla aplikacji Charts.io

Źródło: opracowanie własne

## **3.5 Projekt interfejsu**

Zadaniem projektu interfejsu użytkownika jest zaplanowanie spójnego wyglądu aplikacji pozwalającego na szybkie i swobodne poruszanie się po stronie wynikające z przemyślanego rozmieszczenia wszystkich elementów oraz narzędzi oprogramowania. Projekt powinien skupiać się wokół wszystkich przypadków użycia pozwalających użytkownikowi na dostęp do pełnej funkcjonalności aplikacji. Głównym zadaniem projektu będzie zaplanowanie strony głównej, ekranów logowania oraz rejestracji, zakładek zawierających tabele przychodów i wydatków, pola dodawania i edycji rekordów, wizualizacje wygenerowanych wykresów a także panelu tworzenia sparametryzowanego grafu. Poniższa tabela zawiera opis wymaganych elementów oraz interakcji, jakie powinna posiadać każda z zakładek Charts.io.

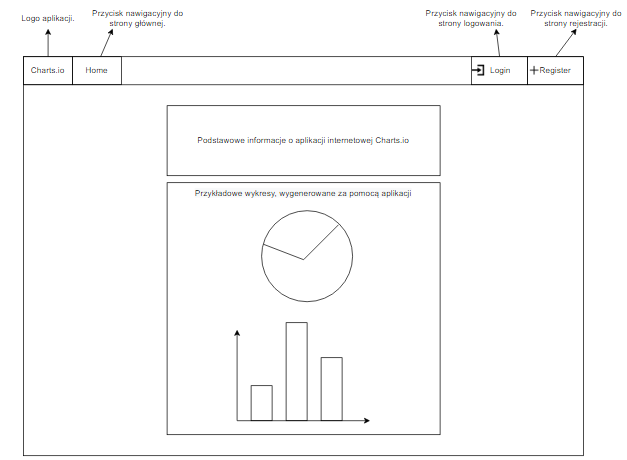
**Tabela 3.12.** Charakterystyka podstawowych widoków aplikacji Charts.io

Źródło: opracowanie własne

|  |  |  |
| --- | --- | --- |
| L.p. | Nazwa | Opis |
| 1. | Strona główna | Strona, którą użytkownik widzi, jako pierwszą rozpoczynając korzystanie z aplikacji, posiada podstawowe informacje zachęcające do korzystania z oferowanych funkcjonalności oraz przedstawia możliwości serwisu w sposób graficzny. |
| 2. | Strona rejestracji | Umożliwia nowemu użytkownikowi założyć konto i rozpocząć korzystanie z serwisu, posiada pola login, hasło, email, które należy wypełnić. Pod wypełnionymi polami znajduje się przycisk rejestracji. |
| 3. | Strona logowania | Pozwala na dostęp do całkowitej funkcjonalności strony, oraz informacji dodanych podczas poprzedniego logowania, składa się z pól loginu oraz hasła, które trzeba podać. Pod wypełnionymi polami znajduje się przycisk logowania. |
| 4. | Pole dodawania i edycji przychodu/wydatku | Wyświetla panel, do którego użytkownik podaje informacje odnośnie swoich przychodów/wydatków, konieczne jest dodanie nazwy rekordu, wartości oraz wybrać datę, którą można wpisać bądź zaznaczyć posługując się kalendarzem. |
| 5. | Zakładka z tabelą przychodów/wydatków | Tabela podzielona na kolumny zawierająca odpowiednio nazwę przychodu/wydatku, wartości, daty oraz akcji. Kolumna akcji składa się z przycisków edycji oraz usunięcia rekordu. Nad tabelą znajduje się wyszukiwarka danych. |
| 6. | Wizualizacja wygenerowanych wykresów | Zakładka, która wyświetla wszystkie wygenerowane przez aplikacje wykresy jeden pod drugim. W prawym górnym rogu znajduj się rozwijana lista kategorii filtrujących po wszystkich wykresach. |
| 7. | Panel tworzenia sparametryzowanych wykresów | Strona wyświetlająca panel, który należy wypełnić odpowiednimi danymi, pod panelem znajduje się przycisk akceptacji podanych informacji. |

Poniższe rysunki przedstawiają zaprojektowany interfejs użytkownika wraz z informacjami o nawigacji między każdą zakładką strony oraz opisem poszczególnych elementów, z których składać się będzie aplikacji Charts.io.

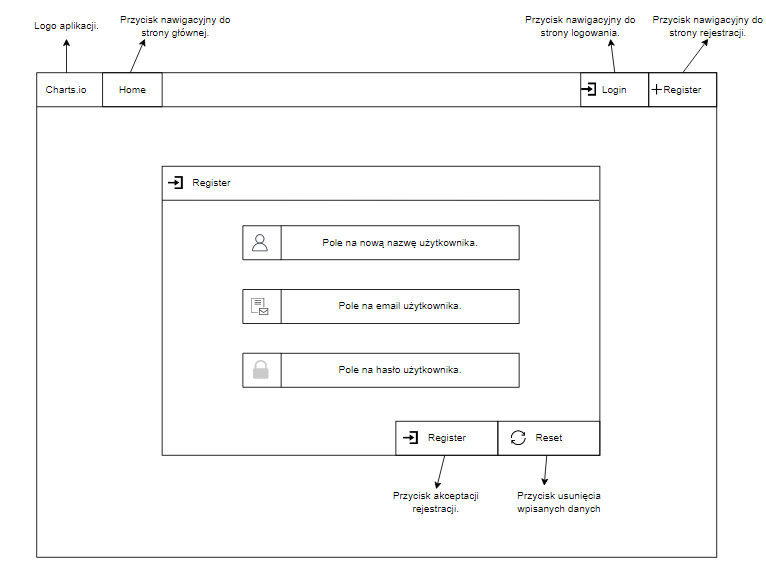
Strona główna:



**Rysunek 3.5.** Strona główna dla aplikacji Charts.io

Źródło: opracowanie własne

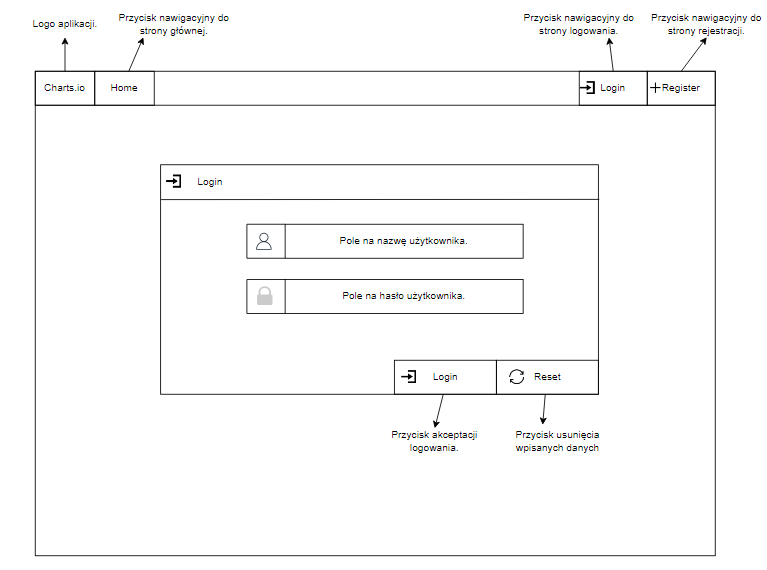
Strona rejestracji nowego użytkownika:



**Rysunek 3.6.** Strona rejestracji nowego użytkownika dla aplikacji Charts.io

Źródło: opracowanie własne

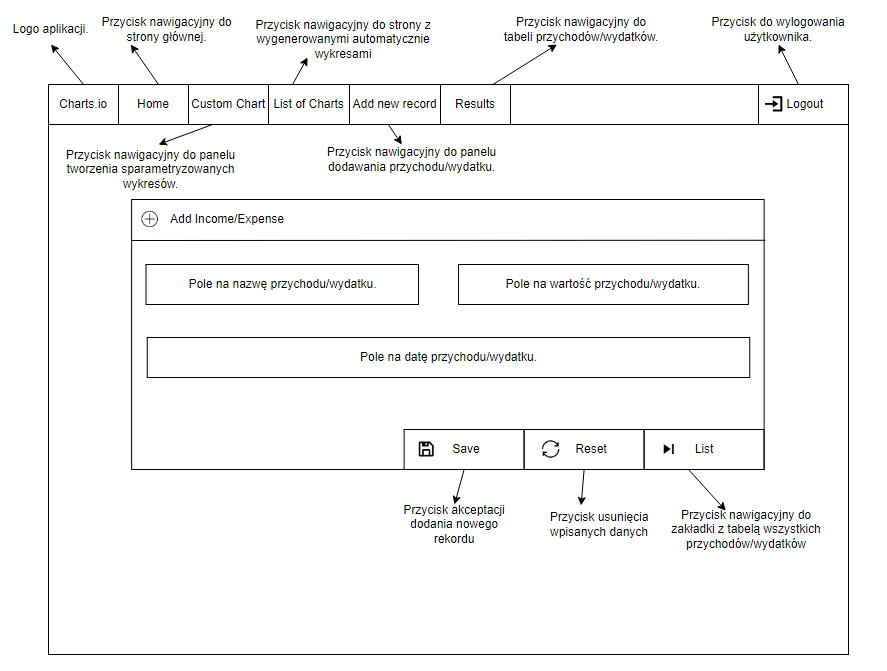
Strona logowania:



**Rysunek 3.7.** Strona logowania dla aplikacji Charts.io

Źródło: opracowanie własne

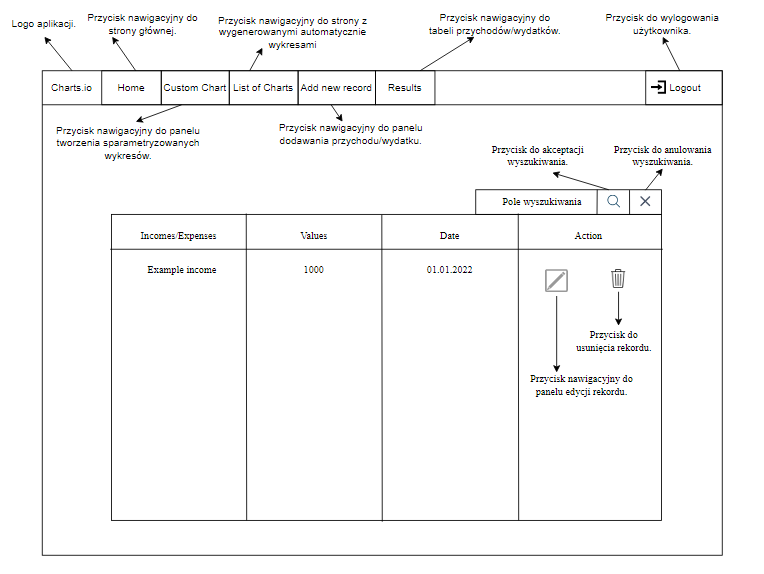
Pole dodawania i edycji przychodu/wydatku:



**Rysunek 3.8.** Strona dodawani przychodu/wydatku dla aplikacji Charts.io

Źródło: opracowanie własne

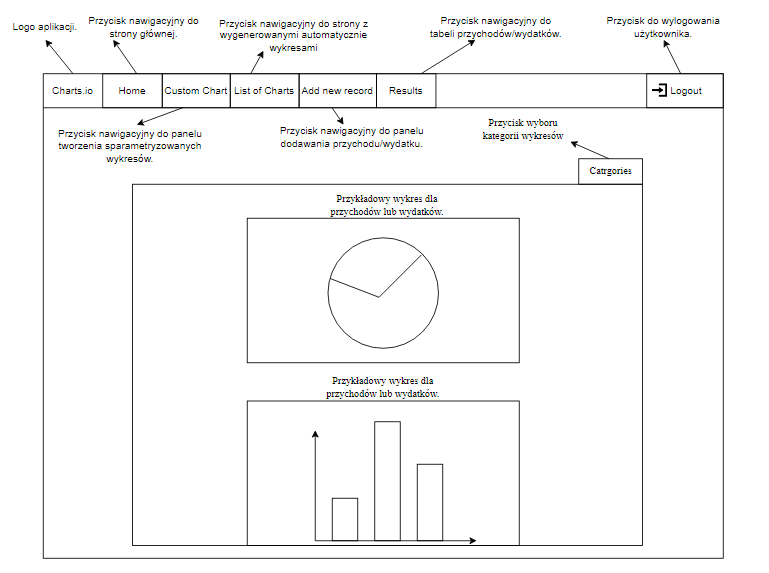
Zakładka z tabelą przychodów/wydatków:



**Rysunek 3.9.** Zakładka z tabelą przychodów/wydatków użytkownika

Źródło: opracowanie własne

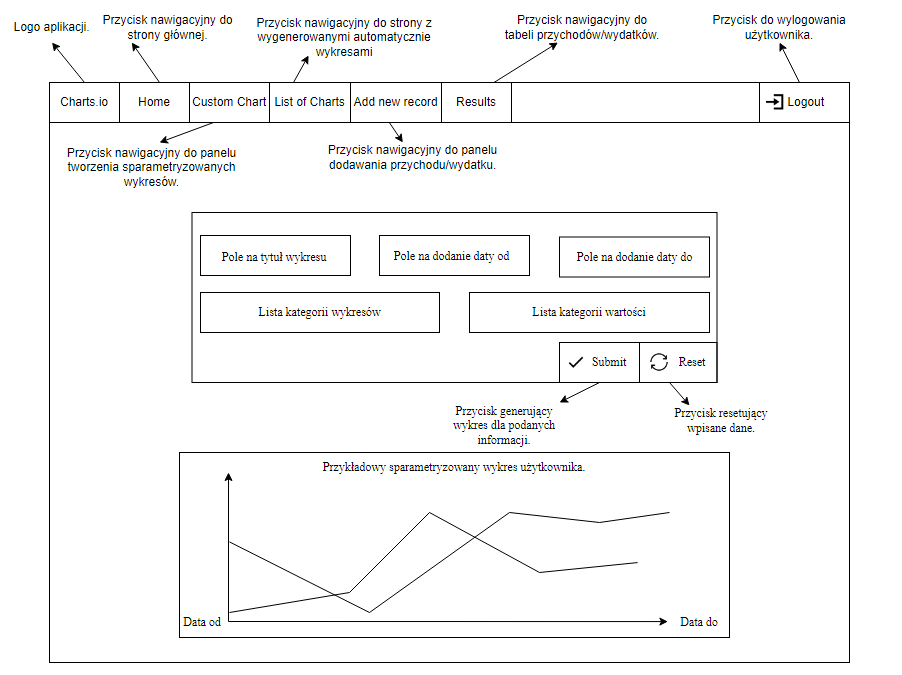
Wizualizacja wygenerowanych wykresów:



**Rysunek 3.10.** Wizualizacja wygenerowanych wykresów

Źródło: opracowanie własne

Panel tworzenia sparametryzowanych wykresów:



**Rysunek 3.11.** Panel tworzenia sparametryzowanych wykresów

Źródło: opracowanie własne

## **4. Implementacja i testowanie**

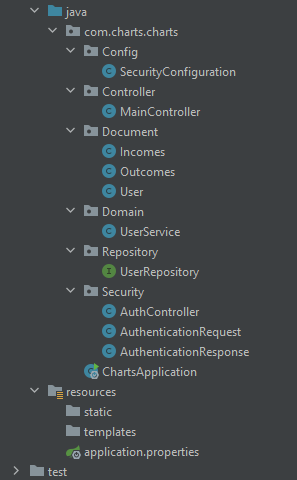
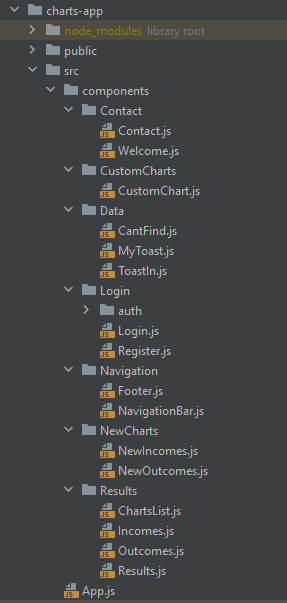
Niniejszy rozdział poświęcony jest implementacji aplikacji internetowej Charts.io. Jego głównym zadaniem będzie zaprezentować oraz scharakteryzować najważniejsze elementy implementacji oprogramowania zestawiając ze sobą wykorzystane technologie ostatecznie zwracając gotowy produkt w postaci aplikacji do graficznego przedstawiania przychodów i wydatków. Rozdział skupia się na schemacie projektu oraz jego działaniu. Wyróżnione zostało środowisko, w którym napisano aplikacje, elementy składowe projektu a także opis implementacji bazy danych dodatkowo wykonano testy manualne, jednostkowe i integracyjne. Kolejne podrozdziały przedstawiają interfejs użytkownika wykonany na podstawie projektu zaprezentowanego w poprzednim rozdziale oraz napotkane problemy implementacyjne wraz z ich rozwiązaniami.

## **4.1 Środowisko**

Aplikacja internetowa Charts.io powstała na komputerze stacjonarnym i przenośnym marki HP korzystających z systemu operacyjnego Windows 10, synchronizacja kodu odbywała się za pomocą systemu kontroli wersji GIT oraz serwisu GitHub. Kod w całości napisany został w zintegrowanym środowisku programistycznym IntelliJ IDEA w wersji 2021.2.2.

## **4.2 Elementy projektu**

Poniższe rysunki przedstawiają zrzut ekranu struktury projektu, w którego skład wchodzą pakiety zawierające zespoły klas odpowiedzialne za określone zadania tworzące w pełni działające oprogramowanie (powstałe zgodnie z warstwowym wzorcem architektonicznym). Przedstawione diagramy prezentują część odpowiedzialną za wygląd strony oraz jej animacje (frontend) oraz tą zajmującą się połączeniem oraz operacjami na bazie danych (backend) skonstruowane w celu zobrazowania separacji tych dwóch płaszczyzn wykorzystujących różne technologie. Oprócz przedstawionych pakietów projekt składa się również z plików konfiguracyjnych oraz katalogu zawierającego testy integracyjne i jednostkowe.

**Rysunek 4.1** Diagram pakietów Charts.io (backend) **Rysunek 4.2** Diagram pakietów Charts.io (frontend)

Źródło: opracowanie własne Źródło: opracowanie własne

## **4.3 Implementacja bazy danych**

Poniższy kod przedstawia plik docker-compose.yaml służący do konfiguracji nowej bazy danych za pomocą kontenera postawionego za pomocą platformy Docker. Oprogramowanie odizolowuje proces konfiguracji serwera MongoDB pozwalając tym samym na dostęp do pełnej funkcjonalności bazy bez obawy o potencjalne błędy powstałe w kodzie źródłowym. Sam proces kontaktowania się kodu z bazą jest obsługiwany za pomocą odpowiednich klas java zawierających wymagane adnotacje, które kontroluje framework Spring Boot. Kolejne fragmenty kodu przedstawiają pliki bazodanowe do generowania nowych dokumentów oraz operowania rekordami bazy.

**Kod źródłowy 4.1** Konfiguracja bazy danych MongoDB

Źródło: opracowanie własne

1. version: "3.8"
2. services:
3. mongodb:
4. image: mongo
5. container\_name: chartsdb
6. ports:
7. - 27017:27017
8. volumes:
9. - data:/data
10. environment:
11. - MONGO\_INITDB\_ROOT\_USERNAME=rootuser
12. - MONGO\_INITDB\_ROOT\_PASSWORD=rootpass
13. mongo-express:
14. image: mongo-express
15. container\_name: mongo-express
16. restart: always
17. ports:
18. - 8081:8081
19. environment:
20. - ME\_CONFIG\_MONGODB\_ADMINUSERNAME=rootuser
21. - ME\_CONFIG\_MONGODB\_ADMINPASSWORD=rootpass
22. - ME\_CONFIG\_MONGODB\_SERVER=mongodb
23. volumes:
24. data: {}
26. networks:
27. default:
28. name: mongodb\_network

**Kod źródłowy 4.2** Implementacja klasy odpowiedzialnej za dane użytkownika

Źródło: opracowanie własne

1. package com.charts.charts.Document;
3. import lombok.Data;
4. import org.springframework.data.annotation.Id;
5. import org.springframework.data.mongodb.core.index.Indexed;
6. import org.springframework.data.mongodb.core.mapping.Document;
7. import java.util.ArrayList;
9. @Data
10. @Document
11. public class User {
13. @Id
14. private String id;
15. @Indexed(unique = true)
16. private String userName;
17. private String password;
18. @Indexed(unique = true)
19. private String email;
20. private ArrayList<Incomes> incomes;
21. private ArrayList<Outcomes> outcomes;
23. public User(String userName, String password, String email) {
24. this.userName = userName;
25. this.password = password;
26. this.email = email;
27. }
29. }

**Kod źródłowy 4.3** Implementacja klasy odpowiedzialnej za przychody użytkownika

Źródło: opracowanie własne

1. package com.charts.charts.Document;
3. import lombok.Data;
5. import java.math.BigDecimal;
6. import java.time.LocalDate;
8. @Data
9. public class Incomes {
11. private Integer id;
12. private LocalDate date;
13. private String incomes;
14. private BigDecimal value;
16. public Incomes(LocalDate date, String incomes, BigDecimal value) {
17. this.date = date;
18. this.incomes = incomes;
19. this.value = value;
20. }
21. }

**Kod źródłowy 4.4** Implementacja klasy odpowiedzialnej za wydatki użytkownika

Źródło: opracowanie własne

1. package com.charts.charts.Document;
3. import lombok.Data;
5. import java.math.BigDecimal;
6. import java.time.LocalDate;
8. @Data
9. public class Outcomes {
11. private Integer id;
12. private LocalDate date;
13. private String outcomes;
14. private BigDecimal value;

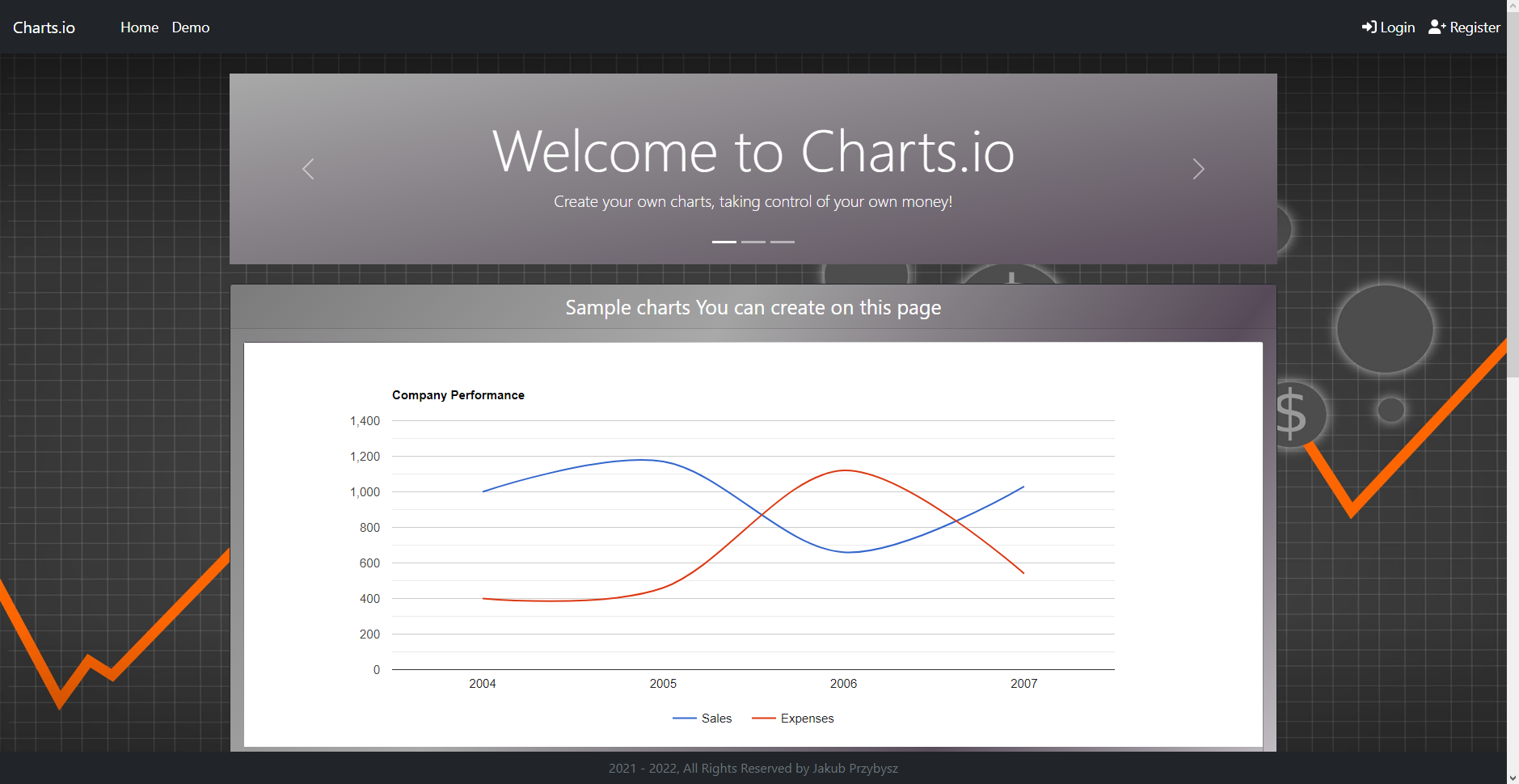
17. public Outcomes(LocalDate date, String outcomes, BigDecimal value) {
18. this.date = date;
19. this.outcomes = outcomes;
20. this.value = value;
21. }
22. }

## **4.4 Interfejs użytkownika**

Interfejs aplikacji Charts.io został stworzony na podstawie projektu zawartego w poprzednim rozdziale uwzględniając wszystkie wymagania funkcjonalne oraz dodatkowe funkcjonalności mające na celu ułatwić oraz uatrakcyjnić korzystanie ze strony potencjalnemu użytkownikowi.

## **4.4.1 Strona główna**

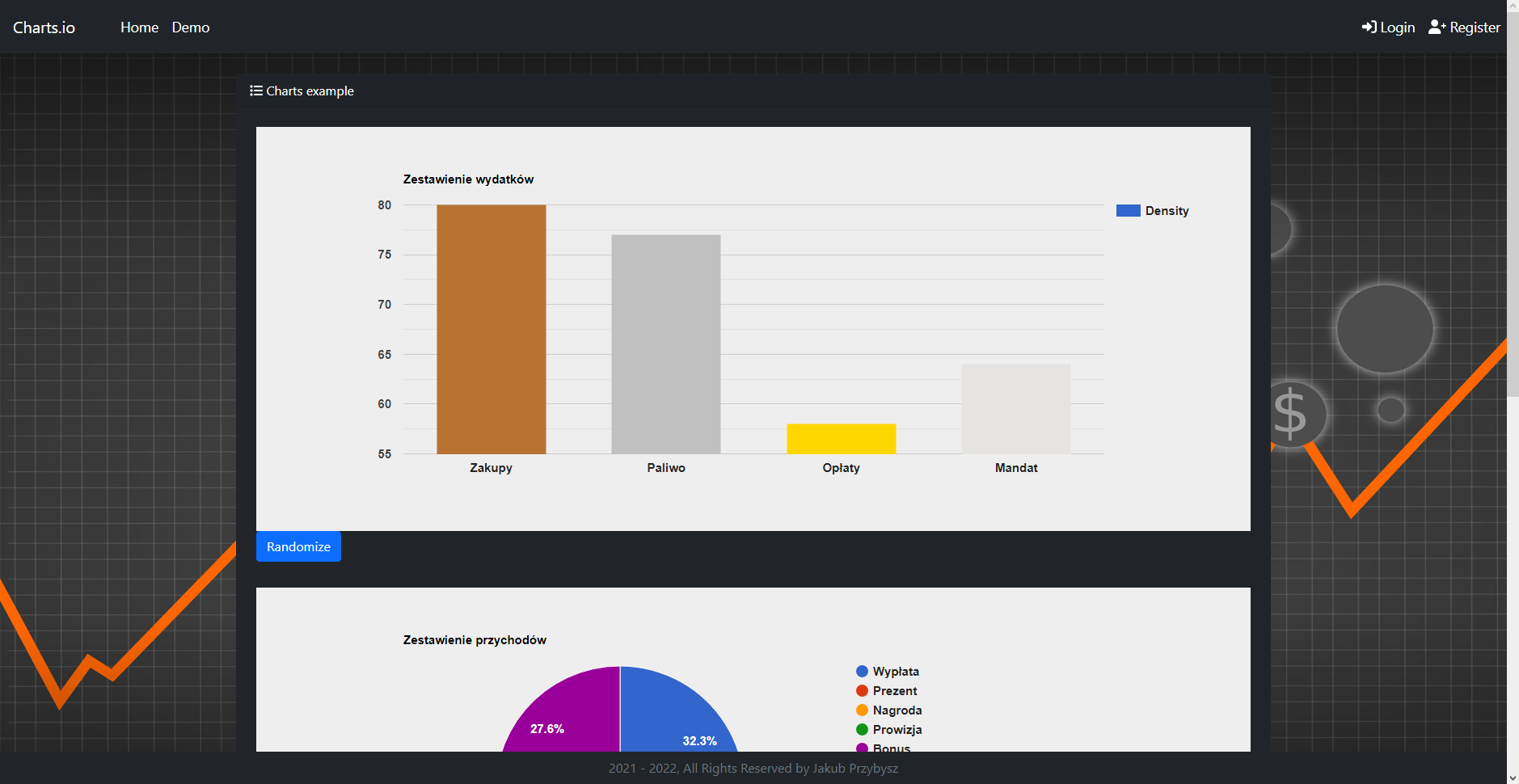
Poniższy rysunek przedstawia stronę główną aplikacji Charts.io. Pasek menu na samej górze składa się z zakładki ‘Home’ (nawigacja do strony głównej), ‘Demo’ (nawigacja do strony demonstracyjnej) oraz ‘Login’ (nawigacja do strony logowania) i ‘Register’ (nawigacja do strony rejestracji), sama strona zawiera dwa elementy: kartę ze zmieniającymi się informacjami na temat aplikacji, których zadaniem jest zachęcenie nowego użytkownika do korzystania z serwisu oraz listy wykresów wygenerowanych statycznie prezentujących jedynie część możliwości serwisu Charts.io.

****

**Rysunek 4.3.** Strona główna aplikacji Charts.io

Źródło: opracowanie własne

Zakładka Demo przenosi użytkownika do strony, która składa się z różnych wykresów, pod którymi znajdują się przyciski do losowania przypadkowych wartości. Osoba korzystająca z aplikacji po raz pierwszy może sprawdzić, w jaki sposób generowane są wizualizacje i jak owe grafiki zmieniają się wraz z nowymi wartościami (dane dla przedstawionych wykresów są generowane losowo).



**Rysunek 4.4.** Zakładka Demo aplikacji Charts.io

Źródło: opracowanie własne

## **4.4.2 Strona rejestracji użytkownika**

Rysunek znajdujący się poniżesz przedstawia stronę rejestracji nowego użytkownika. Panel składa się z trzech pól, które należy wypełnić. Pozytywne wprowadzenie wszystkich informacji przenosi użytkownika do strony logowania.



**Rysunek 4.5.** Strona rejestracji aplikacji Charts.io

Źródło: opracowanie własne

## **4.4.3 Strona logowania użytkownika**

Strona logowania podobnie jak strona rejestracji składa się z jednego elementu tj. panelu logowania pozwalającego na uwierzytelnienie tożsamości użytkownika. Pozytywne wprowadzenie danych przenosi osobę korzystającą z aplikacji do strony głównej udostępniając wszystkie narzędzia oraz funkcjonalności serwisu.

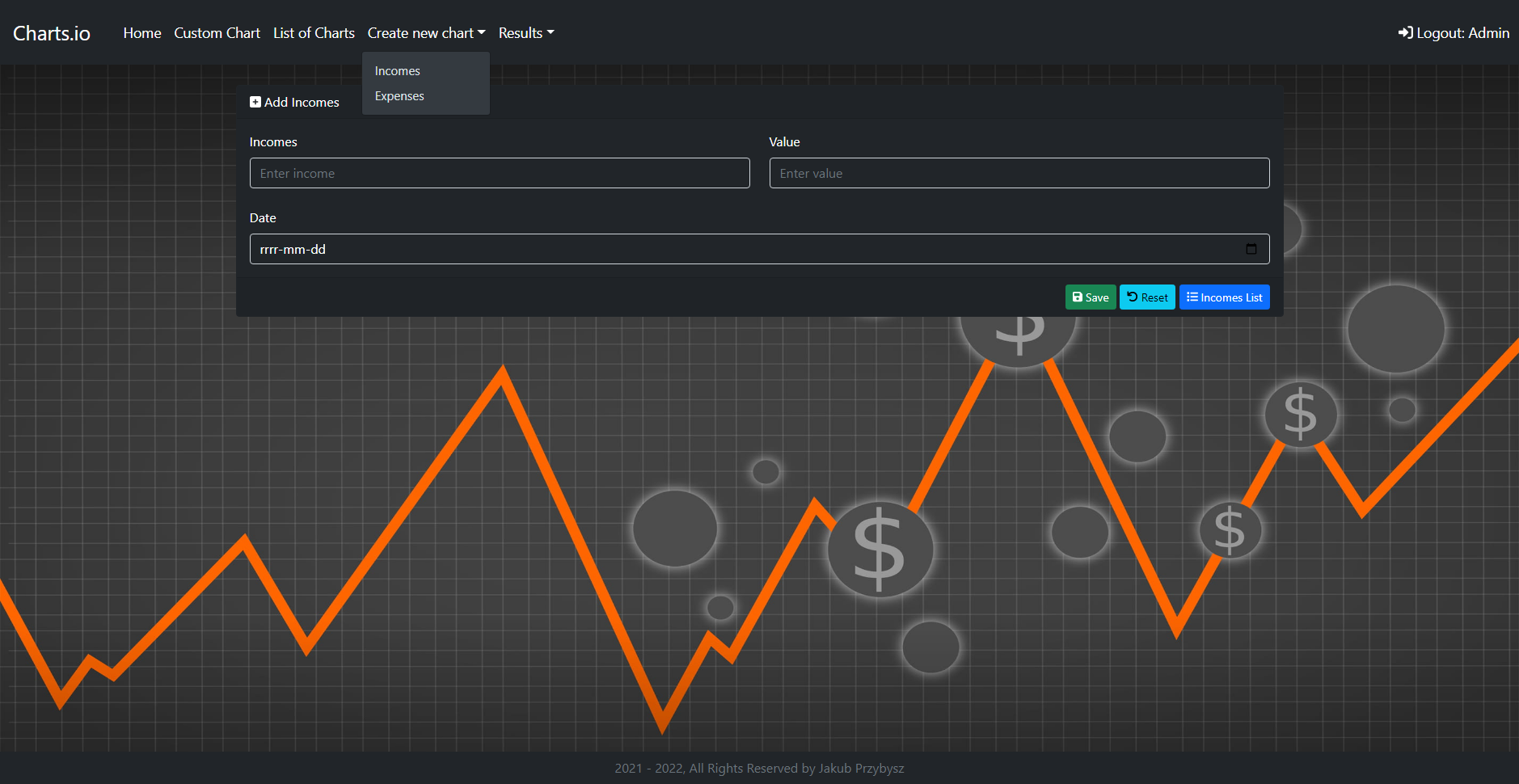


**Rysunek 4.6.** Strona dodawania przychodów w aplikacji Charts.io

Źródło: opracowanie własne

## **4.4.4 Strona dodawania przychodów i wydatków**

Poniższy rysunek przedstawia stronę dodawania przychodów. Zakładka składa się z pól, które należy wypełnić, aby zapisać nową wartość, przyciski znajdujące się poniżej pozwalają na zapis rekordu, reset podanych informacji oraz przekierowanie do zakładki ze wszystkimi dodanymi przychodami. Aby dodać wydatek wybieramy zakładkę znajdującą się, poniżej, która różni się jedynie przypisaniem dodanej wartości do tabeli z wydatkami.

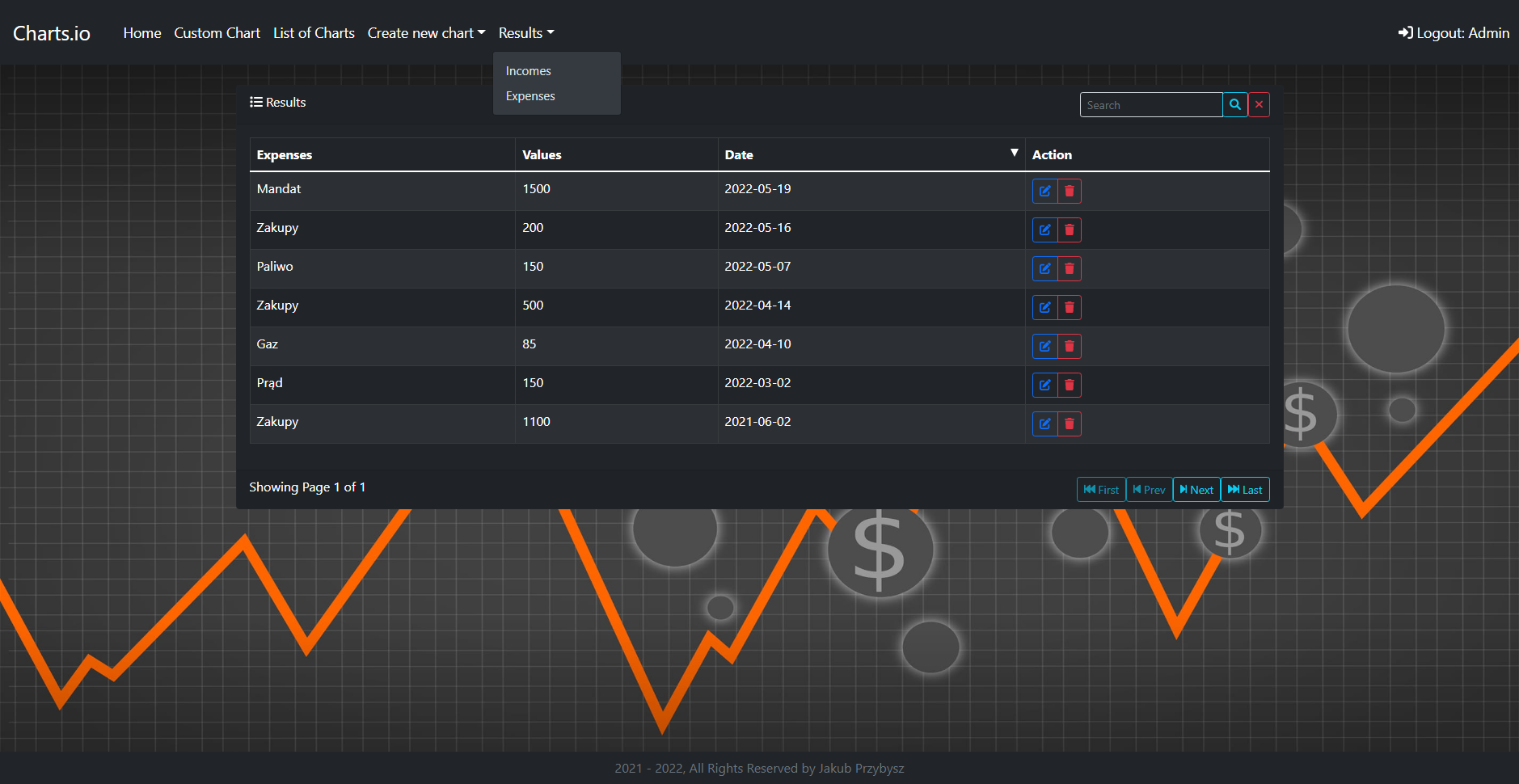


**Rysunek 4.7.** Strona dodawania przychodów w aplikacji Charts.io

Źródło: opracowanie własne

## **4.4.5 Strona z danymi użytkownika**

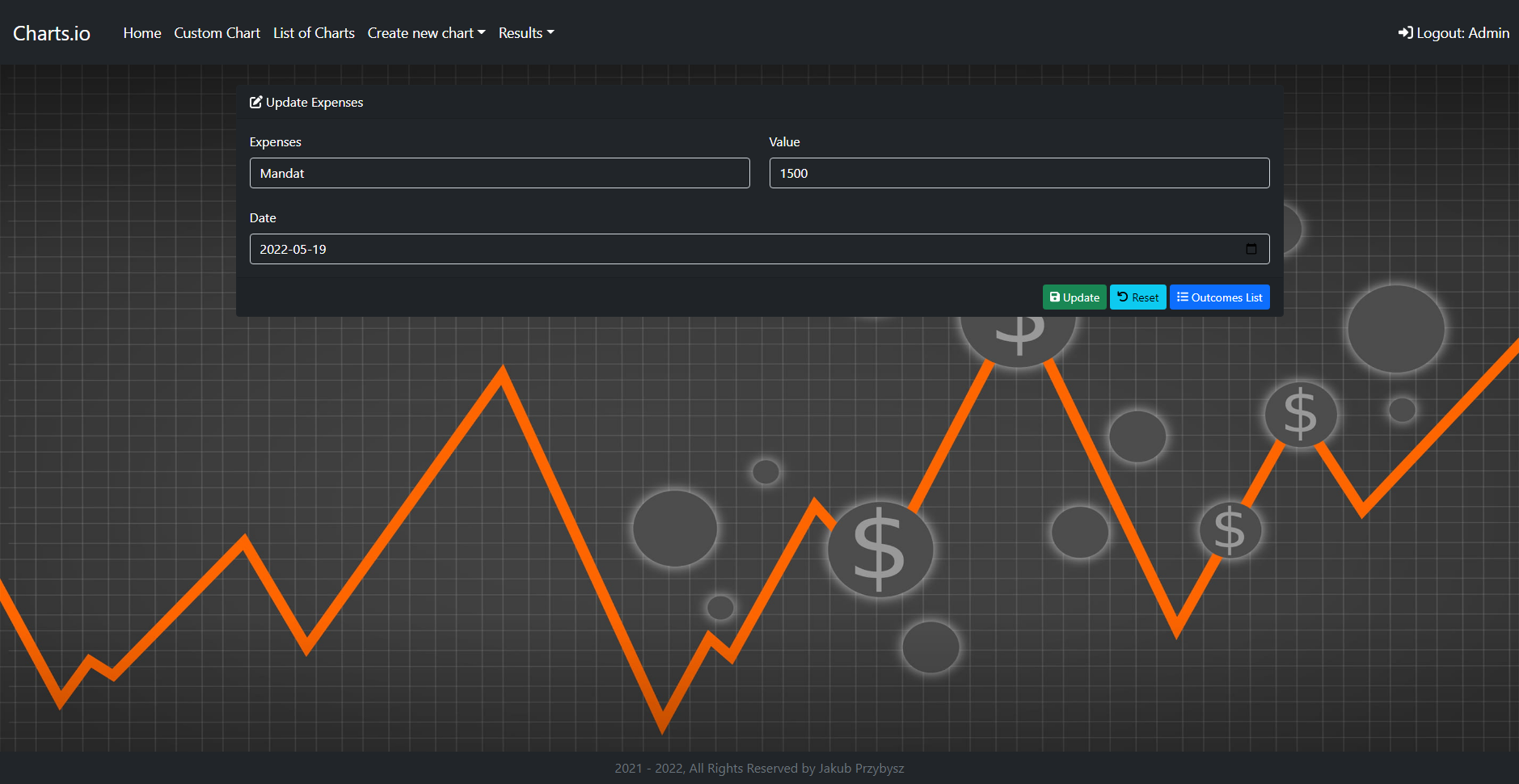
Strona z danymi użytkownika jest w formie tabeli, przedstawia dodane wydatki lub przychody wraz z ich wartościami oraz datą. Strona posiada możliwość wyszukania rekordu po nazwie, istnieje opcja usunięcia danych, przy odpowiedniej ilości wprowadzonych informacji dla czytelniejszego interfejsu dodawane są kolejne strony przychodów lub wydatków. Użytkownik ma możliwość posortowania danych w różnej kolejności.

****

**Rysunek 4.8.** Strona wszystkich wydatków użytkownika w aplikacji Charts.io

Źródło: opracowanie własne

Istnieje również funkcjonalność edycji poszczególnych rekordów, naciśnięcie odpowiedniego przycisku w kolumnie ‘Action’ przenosi osobę korzystającą z aplikacji do ekranu przypominającego zakładkę dodawania przychodu lub wydatku z uzupełnionymi już wartościami i odmienną funkcjonalnością.

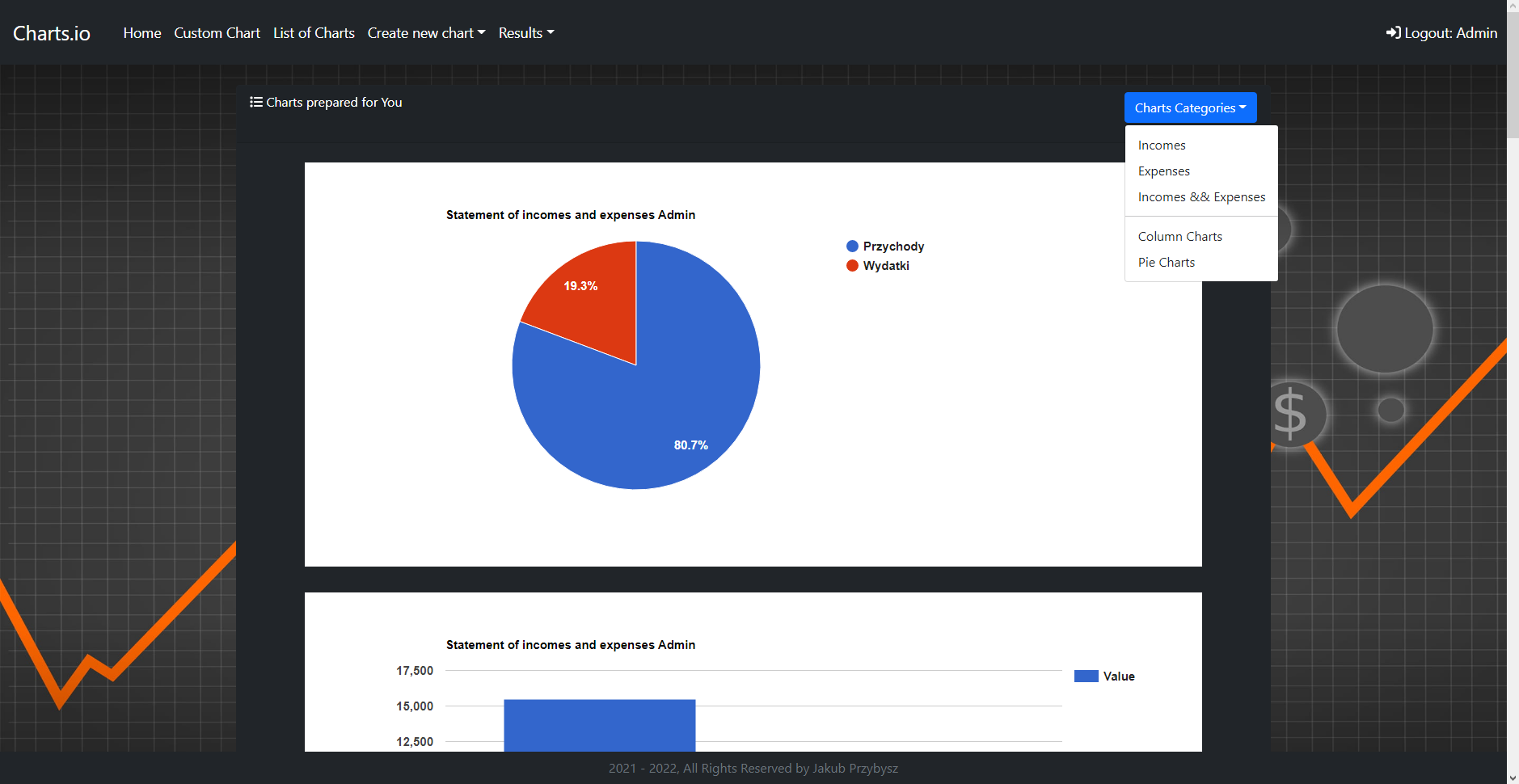
****

**Rysunek 4.9.** Strona edycji wydatku użytkownika w aplikacji Charts.io

Źródło: opracowanie własne

## **4.4.6 Strona z wykresami przychodów i wydatków**

Poniższy rysunek przedstawia stronę z automatycznie wygenerowanymi wykresami przychodów i wydatków w różnej konfiguracji. Witryna wyświetla najpopularniejsze zestawienia rekordów ułatwiając tym samym korzystanie z serwisu. Użytkownik chcąc porównać sumę wszystkich swoich przychodów i wydatków lub sprawdzić zestawienie wszystkich wartości, które zaliczają się do danej grupy nie musi tworzyć nowego wykresu. Aby ograniczyć ilość wyświetlanych grafów dodana została lista kategorii sortująca odpowiednio wyniki.

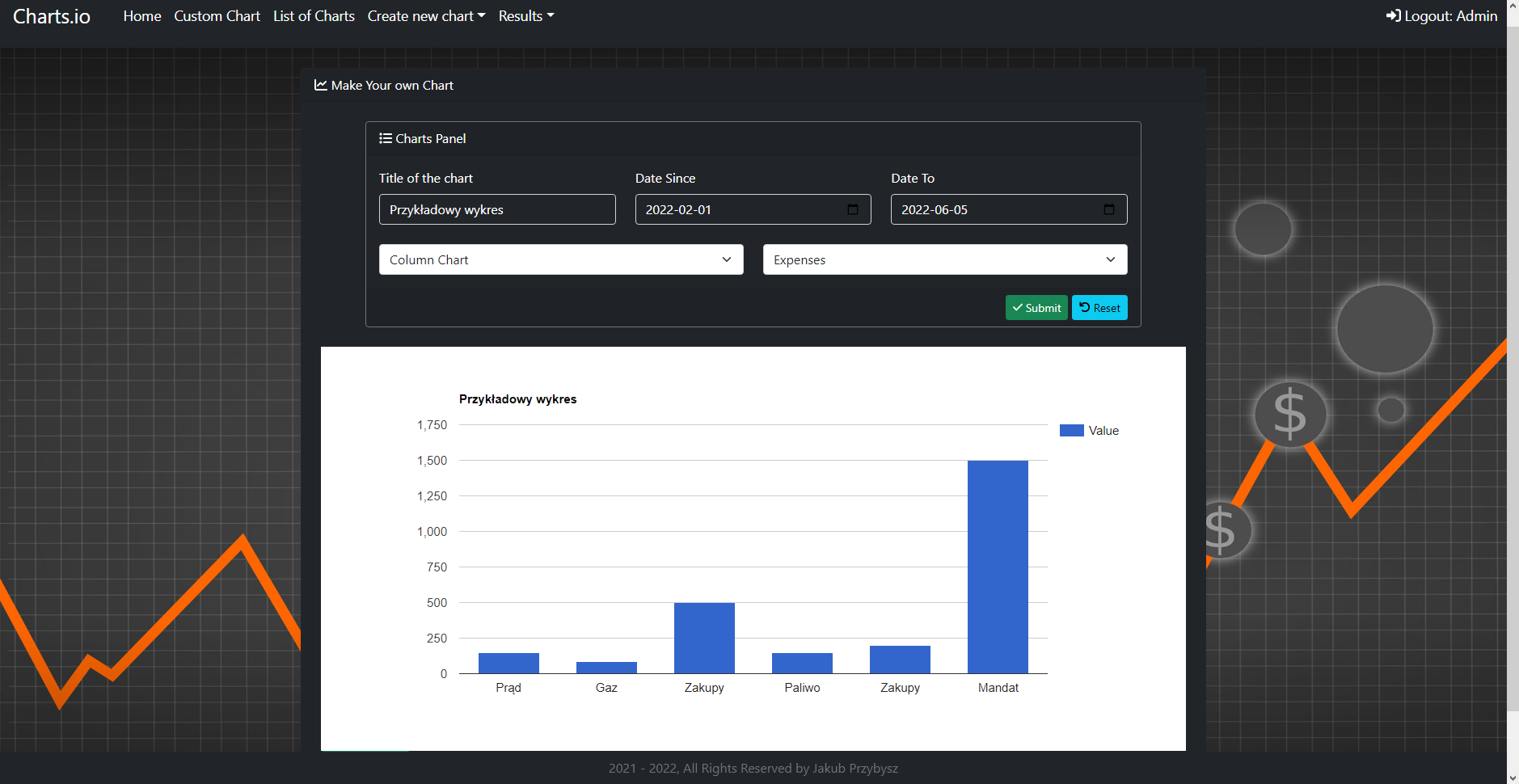


**Rysunek 4.10.** Strona z wykresami przychodów i wydatków aplikacji Charts.io

Źródło: opracowanie własne

### 4.4.7 Strona dodawania sparametryzowanego wykresu

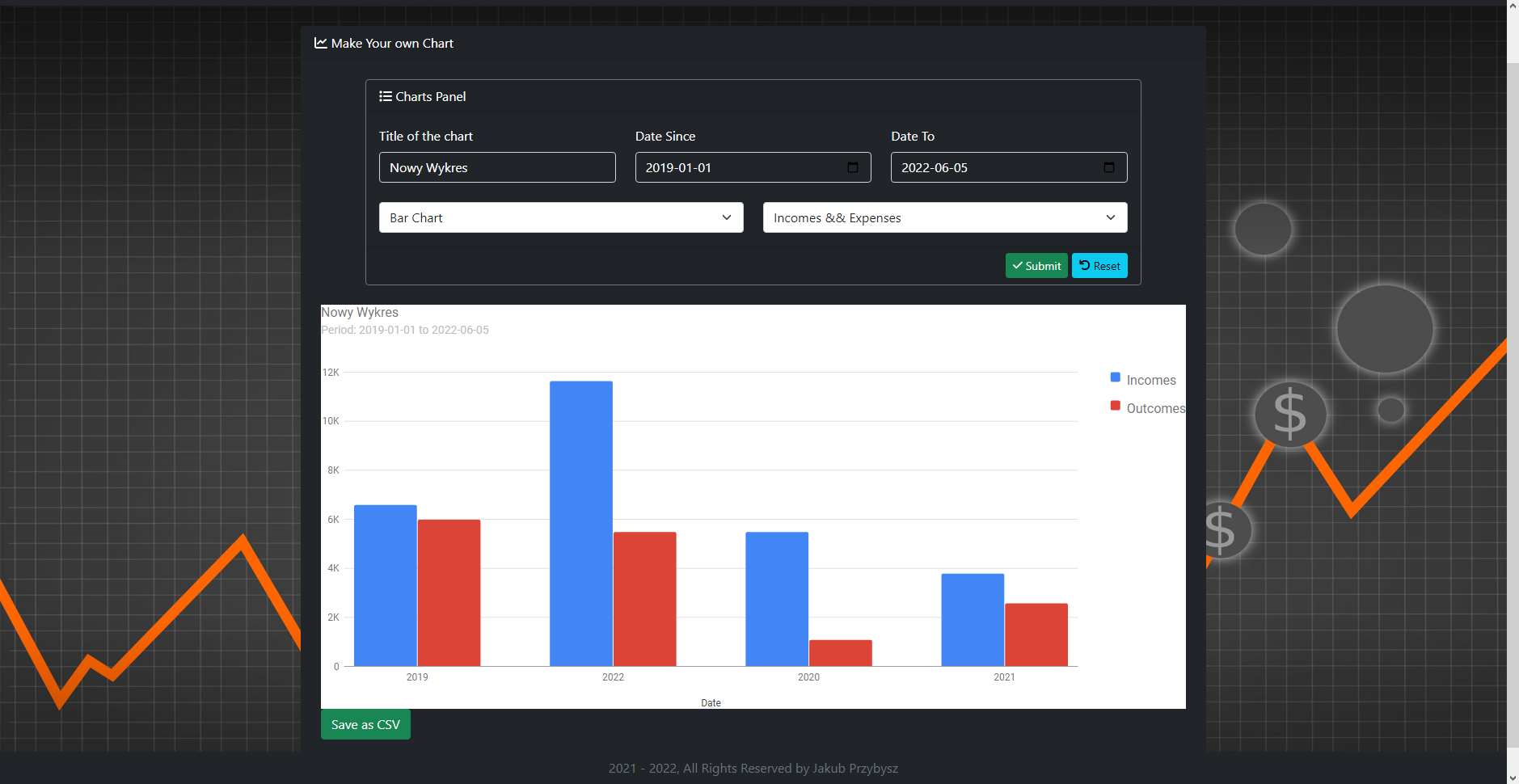
Główną funkcjonalność aplikacji internetowej prezentuje zakładka do tworzenia sparametryzowanych wykresów. Poniższy rysunek obrazuje panel generowania nowego grafu oraz efekt finalny w formie odpowiedniej wizualizacji. Użytkownik może wprowadzić nazwę swojego wykresu, datę, która wyznacza granice początkową pobierania danych, datę końcową oraz kategorie wykresów wraz z rodzajem wyświetlanych wartości.



**Rysunek 4.11.** Strona dodawania sparametryzowanego wykresu w aplikacji Charts.io

Źródło: opracowanie własne

Dodatkowo pod wygenerowanym wykresem dodany został przycisk umożliwiający zapisanie danych z grafu w formacie CSV. Ilość oryginalnych wizualizacji oraz wartości, jakie można zestawiać ze sobą w aplikacji Charts.io połączona z praktycznie dowolnym zakresem dat pozwala na uzyskanie kilkudziesięciu różnych i unikalnych wykresów przedstawiających przychody i wydatki użytkownika.



**Rysunek 4.12.** Strona dodawania nowego wykresu z możliwością zapisu w formacie CSV w aplikacji Charts.io Źródło: opracowanie własne

## **4.5 Testy**

Nieodłącznym zadaniem programisty podczas tworzenia oprogramowania jest pisanie testów, które są narzędziem pozwalającym na sprawdzenie poprawności działania kodu. Testy pozwalają ocenić czy dany fragment kodu działa zgodnie z założeniem i tym samym dają pewność, że wdrożona aplikacja będzie działać prawidłowo. Najpopularniejszymi formami weryfikacji kodu są testy jednostkowe, integracyjne oraz manualne, wszystkie wymienione zastosowano podczas implementacji Charts.io. Struktura testów składa się z trzech części, dostarczenie potrzebnych obiektów lub danych (given), wykonanie akcji np. użycie napisanej wcześniej metody (when) oraz sprawdzenie oczekiwanych parametrów końcowych (then). Do przeprowadzenia testów jednostkowych wykorzystano JUnit5, czyli platformę do testowania rozwiązań programistycznych dla języka Java. JUnit cechują adnotację, dzięki którym możliwe jest zidentyfikowanie metod testowych, co sprawia, że pisanie kodu jest o wiele szybsze. Dodatkowo narzędzie cechuje o wiele mniej skomplikowana składnia w porównaniu do podobnych rozwiązań, co przekłada się bezpośrednio, na jakość oraz czytelność testów[[26]](#footnote-26). Największym udogodnieniem dla dewelopera korzystającego z JUnit jest możliwość uruchomienia serii zestawów sprawdzających określone przypadki testowe, które informują o nieprawidłowościach i błędach powstałych podczas uruchomienia. Do testów integracyjnych posłużył Spring Boot a dokładnie adnotacja @SpringBootTest, która pozwala na wyszukanie głównej klasy konfiguracyjnej i wykorzystanie jej do uruchomienia wszystkich zależności aplikacji Spring. Największą jej zaletą jest możliwość buforowania kontekstu między testami, co pozwala na ograniczenie kosztów uruchomienia całego zestawu metod bez względu na ilość przypadków testowych znajdujących się w klasie[[27]](#footnote-27).

## **4.5.1 Testy jednostkowe**

Głównym zadaniem testów jednostkowych jest sprawdzenie poprawności działania logiki biznesowej, niewielkich fragmentów kodu (najczęściej są to pojedyncze metody) oraz prostych obiektów. W aplikacji internetowej Charts.io testy jednostkowe odpowiedzialne są między innymi za kontrolę metod obsługujących wszelkie operacje na bazie danych oraz pojedynczych elementów kodu. Najważniejszym aspektem pisania testów jest sprawdzenie czy dana metoda działa zgodnie z oczekiwaniami, zweryfikowane powinny zostać wszystkie możliwe przypadki testowe, które sprawdzają czy dany fragment kodu zachowuje się prawidłowo. Framework JUnit pozwala na wygodne tworzenie wielu testów jednostkowych, które weryfikują wynik przeprowadzonych przypadków pozytywnych jak i negatywnych określając czy wszystkie wymagania zostały spełnione. Poniższy fragment kodu przedstawia jeden z testów aplikacji internetowej, który sprawdza funkcjonalność sumowania wszystkich przychodów użytkownika.

**Kod źródłowy 4.5** Test jednostkowy sumujący wszystkie przychody użytkownika

Źródło: opracowanie własne

1. package com.charts.charts.domain;
3. import com.charts.charts.Document.Incomes;
4. import com.charts.charts.Document.User;
5. import com.charts.charts.Domain.IncomesLogic;
6. import org.junit.jupiter.api.Test;
7. import java.math.BigDecimal;
8. import java.time.LocalDate;
9. import java.util.ArrayList;
10. import static org.junit.jupiter.api.Assertions.assertEquals;
12. public class IncomesTests {
14. @Test
15. public void sumOfIncomesTest() {
16. //given
17. Incomes incomesOne = new Incomes(LocalDate.parse("2022-02-02"), "Wypłata", new BigDecimal(3000));
18. Incomes incomesTwo = new Incomes(LocalDate.parse("2022-03-02"), "Nagroda", new BigDecimal(1000));
19. ArrayList<Incomes> incomes = new ArrayList<>();
20. incomes.add(incomesOne);
21. incomes.add(incomesTwo);
22. User user = new User("User", "\*\*\*\*\*\*", "User@.pl");
23. user.setIncomes(incomes);
25. IncomesLogic incomesLogic = new IncomesLogic();
27. //when
28. BigDecimal expectedValue = incomesLogic.sumOfIncomes(user);
30. //then
31. assertEquals(new BigDecimal(4000), expectedValue);
32. }
33. }

## **4.5.2 Testy integracyjne**

Testy integracyjne obejmują szerszy aspekt działania aplikacji, w przeciwieństwie do testów jednostkowych ich zadaniem jest sprawdzenie zgodności pomiędzy modułem a systemem a także wychwycenie potencjalnych błędów w funkcjonowaniu komponentów spełniających określone wymagania. Często do przeprowadzenia takich testów konieczne jest nawiązanie połączenia z bazą danych lub innych współpracujących z aplikacją systemów. Spring Boot posiada szereg narzędzi usprawniających cały proces przeprowadzania testów integracyjnych, które pozwalają na zweryfikowanie wielu metod a nawet wielu przypadków testowych o tej samej konfiguracji ponosząc niski koszt nakładu pamięci (@SpringBootTest). Jedną z takich adnotacji udostępnianych przez framework pozwalających na kontrolowanie pamięci przeprowadzanych operacji jest adnotacja @DirtiesContext. Poniższy fragment kodu przedstawia jeden z testów integracyjnych aplikacji internetowej Charts.io, który sprawdza poprawność dodawania przychodu użytkownika do bazy danych wykorzystując w ten sposób szereg zależności. Test pozwala również przetestować komunikacje między serwisem a bazą MongoDB.

**Kod źródłowy 4.6** Test integracyjny dodający nowy przychód do bazy użytkownika

Źródło: opracowanie własne

1. package com.charts.charts.domain;
3. import com.charts.charts.Document.Incomes;
4. import com.charts.charts.Document.User;
5. import com.charts.charts.Domain.UserService;
6. import com.charts.charts.Repository.UserRepository;
7. import org.junit.jupiter.api.Test;
8. import org.springframework.beans.factory.annotation.Autowired;
9. import org.springframework.boot.test.context.SpringBootTest;
10. import org.springframework.boot.test.mock.mockito.MockBean;
11. import java.math.BigDecimal;
12. import java.time.LocalDate;
13. import static org.junit.jupiter.api.Assertions.assertTrue;
15. @SpringBootTest
16. public class ServiceTests {
18. @MockBean
19. private UserRepository userRepository;
21. @Autowired
22. private final UserService userService = new UserService(userRepository);
24. @Test
25. public void addProperIncomes(){
26. //given
27. Incomes incomes = new Incomes(LocalDate.parse("2022-02-02"), "Wypłata", new BigDecimal(3000));
28. User user = new User("User", "password", "User@.pl");
30. //when
31. userService.addIncomes(incomes, user);
33. //then
34. assertTrue(user.getIncomes().contains(incomes));
35. }
36. }

## **4.5.3 Testy manualne**

W przeciwieństwie do powyższych testów, istnieją również alternatywne sposoby sprawdzania poprawności działania oprogramowania. Jedną z takich metod są testy manualne, których głównym założeniem jest zweryfikowanie kodu poprzez bezpośredni kontakt dewelopera lub testera z aplikacją na określonych etapach jej powstawania. W przypadku Charts.io testy manualne odbyły się podczas powstawania projektu, w trakcie, którego wielokrotnie zweryfikowano wszystkie funkcjonalności aplikacji internetowej.

## **4.6 Problemy implementacyjne**

Poniższa tabela przedstawia największe problemy implementacyjne zaistniałe podczas tworzenia aplikacji internetowej Charts.io. Tabela oprócz opisu problemu zawiera również jego rozwiązanie oraz potencjalne zagrożenia, które mogłyby wpłynąć negatywnie na korzystanie ze strony.

**Tabela 4.1.** Problemy implementacyjne wraz z ich rozwiązaniami

Źródło: opracowanie własne

|  |  |
| --- | --- |
| Opis | Rozwianie |
| Działanie aplikacji internetowej od strony kodu polega na separacji technologii odpowiedzialnych za wizualizacje strony (frontend) oraz kontakt z bazą danych (backend). Backend za pomocą frameworka Spring wystawia endpointy za pomocą, których obsługiwane są procesy w bazie, aby frontend mógł skorzystać z potrzebnych danych konieczne było dodanie obsługi bezpiecznej komunikacji. | Brak możliwości połącznie ze sobą dwóch różnych technologii blokuje tworzenie większości funkcjonalności aplikacji. W projekcie po stronie wizualnej posłużono się biblioteką ‘Axios’, której zadaniem jest pobieranie danych za pomocą adresów URL (w przypadku aplikacji Charts.io endpointów, będących REST API stworzonego serwisu). Dla bezpieczeństwa każdy projekt z wykorzystaniem Spring framework automatycznie blokuje możliwość udostępniania swoich adresów jakimkolwiek źródłom zewnętrznym. Problem został rozwiązany za pomocą jednej z adnotacji frameworka Spring - @CrossOrigin, która pozwala na udostępnianie zasobów między różnymi serwerami zewnętrznymi. Zezwala na dostęp wszelkim nagłówkom, metodom oraz adresom HTTP, dając pewność, co do źródeł korzystających z wystawionego API. Wykorzystanie adnotacji wygląda w ten sposób:  @CrossOrigin("http://localhost:3000") |
| Podczas tworzenia funkcjonalności pobierania nowo wygenerowanych wykresów w formacie CSV wszystkie dane, które zwrócił program wyświetlane były w jednej kolumnie tabeli dla oprogramowania Excel jak i Google Scheet. | Programy, które pozwalają na edycje plików w formacie CSV udostępniają szereg narzędzi, dzięki którym istnieje możliwość dowolnego modulowania komórek, tym samym pozwalając na naprawienie zaistniałego błędu, jednak problem powstały w wyniku implementacji pobierania danych powinien być obsługiwany przez aplikacje. Błędne zestawienie ze sobą informacji mogłoby negatywnie wpłynąć na ocenę całkowitą strony przez potencjalnego użytkownika. Do zaimplementowania funkcjonalności posłużył komponent CSVLink umożliwiający pobieranie odpowiednich informacji w postaci tabeli. Problemem okazał się separator, który z automatu ustawiony jest, jako ‘, ‘ a który w najnowszych wersjach oprogramowania typu Excel już nie funkcjonuje. Aby prawidłowo porozdzielać dane w czytelny sposób zastosowano nadpisanie znaku metodą ‘separator(”;”)’. Poniższy rysunek przedstawia efekt pobranych rekordów na podstawie nowo powstałego wykresu.  **Rysunek 4.13.** Wartości wygenerowanego wykresu w formacie CSV  Źródło: opracowanie własne |

## **Podsumowanie**

Celem pracy było opracowanie aplikacji internetowej do graficznego przedstawienia przychodów i wydatków. Praca jest odpowiedzią na potrzebę istnienia oprogramowania o konkretnych zastosowaniach zwłaszcza w dzisiejszych czasach jako dodatkowa forma kontroli własnych finansów. W zakres pracy wchodzi dokładna analiza rynku aplikacji posiadających podobne zastosowania oraz funkcjonalności, wykorzystane technologie potrzebne do realizacji projektu, zarys pracy, jego potencjalny wygląd oraz podstawowe wymagania. Wynikiem końcowym jest implementacja samej aplikacji zawierającej przeprowadzone testy, rozwiązane problemy, najistotniejsze elementy wyszczególnione w poprzednich rozdziałach oraz efekt finalny w postaci działającego serwisu.

Analiza rynkowa zestawiła najpopularniejsze aplikacje internetowe posiadające szereg narzędzi oraz opcji do generowania wykresów na podstawie danych użytkownika. Każda strona wyróżniała się nietypowymi funkcjonalnościami, przyciągając tym samym odpowiednie grupy zainteresowanych. Praca zwraca uwagę na największe wady serwisów wpływające na potencjalne niezadowolenie użytkownika oraz przedstawia obszary rozwoju na których skupiła się aplikacja Charts.io.

Zastosowane technologie podczas implementacji aplikacji zostały dokładnie przeanalizowane i wybrane z pośród najpopularniejszych rozwiązań dostępnych na rynku. Aby umożliwić korzystanie ze strony jak największej liczbie odbiorców należało wyselekcjonować najbardziej atrakcyjne komponenty udostępniane przez szereg frameworków oraz możliwości płynące z użytkowania konkretnego języka programowania. Istotnym elementem pracy było zwrócenie uwagi na prywatność w dostępie do danych, jako jeden z większych aspektów bezpieczeństwa strony. Projekt został skonstruowany za pomocą technologii umożliwiających tworzenie responsywnego interfejsu użytkownika cechujący się niskim nakładem pracy w momencie jego utrzymania lub rozwoju.

Praca posiada zarys projektu, który określa najważniejsze wymagania oraz scenariusze użycia. Zadaniem rozdziału było zaplanowanie od strony technicznej zarysu wyglądu całej aplikacji a także schematu bazy danych i jej funkcjonalności oraz zwrócenie uwagi na potencjalne problemy, które mogłyby się pojawić podczas implementacji strony. Wykorzystanie zarysu projektu było konieczne do wyszczególnienia wszystkich możliwości serwisu a także przeprowadzenia sprawnej implementacji.

Analiza rynku aplikacji konkurencyjnych wraz z odpowiednim doborem narzędzi technologicznych połączona ze szczegółowym projektem pracy umożliwiła zaimplementowanie założeń tworząc w pełni funkcjonalne oprogramowanie. Wykorzystane technologie dają możliwość rozszerzenia aplikacji w przyszłości o nowe funkcjonalności. Praca pozwoliła na opracowanie projektu do stworzenia wygodnego oraz bezpiecznego produktu do graficznego przedstawiania przychodów i wydatków każdemu potencjalnemu użytkownikowi.

## **Bibliografia**

Książki:

1. Balicki A., Analiza rynku., Wydawnictwo Wyższej Szkoły Zarządzania w Gdańsku, Gdańsk 2002, s. 12
2. Baker M. J., The Marketing Book-Fifth Edition, Butterworth-Heinemann, Oxford 2003, s. 320
3. David Flanagan, JavaScript : the definitive guide, [O'Reilly Media](https://helion.pl/wydawca/o-reilly-media) 2011
4. James Gosling, Henry McGilton, The Java Language Environment, 2550 Garcia Avenue Mountain View, CA 94043 U.S.A. październik 1995, s. 10

Artykuły:

1. Krzysztof Czubak, Case study – analiza rynku stron internetowych, 3 czerwca 2022
2. Lokesh Gupta, What is REST, 7 kwietnia 2022
3. Gustavo Romero, What is Postman API Test, 9 czerwca 2021
4. Rachel Meltzer, What is JavaScript used for?, 3 grudnia 2020
5. Paul Krill, React: Making faster, smoother UIs for data-driven Web apps, 15 maja 2014
6. Chris Dawson, JavaScript’s History and How it Led To ReactJS, 25 lipca 2013
7. Esau Silva, How To Make create-react-app work with a Node Back-end API, 21 grudnia 2017
8. Krunal Panchal, Angular vs React Detailed Comparison 2022, 26 kwietnia 2022
9. Ubaid Pisuwala, The benefits of ReactJS and reasons to choose it for your project, 7 lutego 2022
10. Chris Wanstrath, We Launched, 17 kwietnia 2008
11. Chris Wanstrath , A bright future for GitHub, 4 czerwca 2018
12. Przemysław Bykowski, Maven I Gradle – porównanie narzędzi i ich wydajności, 3 czerwca 2020
13. Mendy Newman, JavaScript Puts 97% of Websites at Risk of Infection, 5 października 2020

Strony internetowe:

1. <https://mfiles.pl/pl/index.php/Analiza_rynku>
2. <https://www.doradcawbiznesie.pl/case-study-analiza-rynku-stron-internetowych/>
3. <https://www.mongodb.com/why-use-mongodb>
4. <https://www.programosy.pl/program,intellij-idea-community-edition.html>
5. <https://www.jetbrains.com/help/idea/apply-changes-from-one-branch-to-another.html#rebase-branch>
6. <https://aws.amazon.com/what-is/api/>
7. <https://restfulapi.net>
8. [https://smartbear.com/learn/performance-monitoring/api-endpoints](https://smartbear.com/learn/performance-monitoring/api-endpoints/)
9. <https://www.postman.com>
10. <https://www.encora.com/insights/what-is-postman-api-test>
11. <https://www.redhat.com/en/topics/api/what-is-a-rest-api>
12. <https://www.lighthouselabs.ca/en/blog/what-is-javascript-used-for>
13. <https://en.wikipedia.org/wiki/React_(JavaScript_library)>
14. <https://reactjs.org>
15. <https://www.infoworld.com/article/2608181/react--making-faster--smoother-uis-for-data-driven-web-apps.html>
16. <https://www.freecodecamp.org/news/how-to-make-create-react-app-work-with-a-node-backend-api-7c5c48acb1b0>
17. <https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html>
18. <https://hub.packtpub.com/web-development-react-and-bootstrap>
19. <https://react-bootstrap.github.io>
20. <https://en.wikipedia.org/wiki/Git>
21. <https://pl.wikipedia.org/wiki/GitHub>
22. <https://github.blog/2008-04-10-we-launched>
23. <https://github.blog/2018-06-04-github-microsoft>
24. <https://en.wikipedia.org/wiki/IntelliJ_IDEA>
25. <https://bykowski.pl/maven-i-gradle-porownanie-narzedzi-i-ich-wydajnosci>
26. <https://en.wikipedia.org/wiki/Java_(programming_language)>
27. <https://www.computerweekly.com/feature/Write-once-run-anywhere>
28. <https://docs.oracle.com/javase/tutorial/java/concepts/index.html>
29. <https://java.com/pl/download/help/whatis_java.html>
30. <https://en.wikipedia.org/wiki/JavaScript>
31. <https://marketingwsieci.pl/slownik-e-marketingu/framewor>
32. <https://en.wikipedia.org/wiki/MongoDB>
33. <https://en.wikipedia.org/wiki/JavaScript>
34. <https://blog.ericom.com/javascript-puts-websites-at-risk-of-infection>
35. <https://kinsta.com/knowledgebase/what-is-github>
36. <https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction>
37. <https://www.tutorialspoint.com/junit/junit_overview.htm>
38. <https://spring.io/guides/gs/testing-web/>

## **Spis rysunków**

* 1. Strona główna Online Charts ......................................................................................................... 12

1.2. Panel dodawania wykresu strony Online Charts............................................................................. 12

1.3. Wygenerowany wykres strony Online Charts ................................................................................ 13

1.4. Strona główna Livegap Charts ...................................................................................................... 14

1.5. Wygenerowany wykres wraz panelem edycji danych strony Livegap Charts…………………….14

1.6. Szablon Monthly budget strony Google Sheets.............................................................................. 16

1.7. Wygenerowany wykres wraz z tabelą danych strony Google Sheets............................................. 16

2.1. Drzewo robocze w aplikacji IntelliJ systemu kontroli wersji GIT................................................. 20

2.2. Strona tworzenia projektu Spring framework................................................................................. 23

2.3. Dokument nierelacyjnej bazy danych MongoDB w formacie JSON.............................................. 24

2.4. Aplikacja Postman z zapytania GET............................................................................................... 25

2.5. Aplikacja Postman z zapytania POST............................................................................................. 26

2.6. Ranking najpopularniejszych języków programowania z 2021 roku............................................. 27

3.1. Diagram przypadków użycia........................................................................................................... 33

3.2. Schemat nierelacyjnej bazy danych MongoDB dla aplikacji Charts.io nr 1…............................... 43

3.3. Schemat nierelacyjnej bazy danych MongoDB dla aplikacji Charts.io nr 2................................... 44

3.4. Diagram klas dla aplikacji Charts.io............................................................................................... 45

3.5. Strona główna dla aplikacji Charts.io............................................................................................. 47

3.6. Strona rejestracji nowego użytkownika dla aplikacji Charts.io...................................................... 48

3.7. Strona logowania dla aplikacji Charts.io......................................................................................... 49

3.8. Strona dodawani przychodu/wydatku dla aplikacji Charts.io......................................................... 50

3.9. Zakładka z tabelą przychodów/wydatków użytkownika................................................................. 51

3.10. Wizualizacja wygenerowanych wykresów................................................................................... 52

3.11. Panel tworzenia sparametryzowanych wykresów......................................................................... 53

4.1. Diagram pakietów Charts.io (backend)........................................................................................... 56

4.2. Diagram pakietów Charts.io (frontend).......................................................................................... 56

4.3. Strona główna aplikacji Charts.io................................................................................................... 60

4.4. Zakładka Demo aplikacji Charts.io................................................................................................. 60

4.5. Strona rejestracji aplikacji Charts.io............................................................................................... 61

4.6. Strona dodawania przychodów w aplikacji Charts.io..................................................................... 62

4.7. Strona dodawania przychodów w aplikacji Charts.io..................................................................... 63

4.8. Strona wszystkich wydatków użytkownika w aplikacji Charts.io ................................................. 64

4.9. Strona edycji wydatku użytkownika w aplikacji Charts.io............................................................. 65

4.10. Strona z wykresami przychodów i wydatków aplikacji Charts.io................................................ 66

4.11. Strona dodawania sparametryzowanego wykresu w aplikacji Charts.io...................................... 67

4.12. Strona dodawania nowego wykresu z możliwością zapisu w formacie CSV w aplikacji Charts.io………………………………………………………………………………………………. 68

4.13. Wartości wygenerowanego wykresu w formacie CSV................................................................. 73

## **Spis tabel**

1.1. Wady aplikacji konkurencyjnych.................................................................................................... 17

2.1. Pozostałe biblioteki JavaScript........................................................................................................ 29

3.1. PU#001 Rejestracja......................................................................................................................... 34

3.2. PU#002 Logowanie......................................................................................................................... 34

3.3. PU#003 Wyświetlanie tabeli wszystkich przychodów/wydatków.................................................. 35

3.4. PU#004 Sortowanie wszystkich przychodów/wydatków w tabeli według daty............................. 35

3.5. PU#005 Wyszukiwanie przychodów/wydatków w tabeli po nazwie…......................................... 36

3.6. PU#006 Dodanie przychodu/wydatku............................................................................................ 37

3.7. PU#007 Edytowanie przychodu/wydatku ...................................................................................... 38

3.8. PU#008 Usuwanie przychodu/wydatku.......................................................................................... 39

3.9. PU#009 Wyświetlanie wszystkich wykresów wygenerowanych automatycznie dla przychodów i wydatków............................................................................................................................................... 40

3.10. PU#010 Wyświetlanie wykresów wygenerowanych automatycznie zgodnych z wybraną kategorią ……………………………………………………………………………………………….40

3.11. PU#011 Generowanie nowego sparametryzowanego wykresu ................................................... 41

3.12. Charakterystyka podstawowych widoków aplikacji Charts.io .................................................... 46

4.1. Problemy implementacyjne wraz z ich rozwiązaniami ................................................................. 72

## **Spis kodów źródłowych**

4.1. Konfiguracja bazy danych MongoDB............................................................................................. 57

4.2. Implementacja klasy odpowiedzialnej za dane użytkownika ......................................................... 58

4.3. Implementacja klasy odpowiedzialnej za przychody użytkownika................................................ 58

4.4. Implementacja klasy odpowiedzialnej za wydatki użytkownika.................................................... 59

4.5. Test jednostkowy sumujący wszystkie przychody użytkownika ................................................... 70

4.6. Test integracyjny dodający nowy przychód do bazy użytkownika................................................. 71

1. https://mfiles.pl/pl/index.php/Analiza\_rynku [↑](#footnote-ref-1)
2. https://www.doradcawbiznesie.pl/case-study-analiza-rynku-stron-internetowych/ [↑](#footnote-ref-2)
3. https://pl.wikipedia.org/wiki/Git\_(oprogramowanie) [↑](#footnote-ref-3)
4. https://kinsta.com/knowledgebase/what-is-github [↑](#footnote-ref-4)
5. https://en.wikipedia.org/wiki/IntelliJ\_IDEA [↑](#footnote-ref-5)
6. https://bykowski.pl/maven-i-gradle-porownanie-narzedzi-i-ich-wydajnosci [↑](#footnote-ref-6)
7. https://www.programosy.pl/program,intellij-idea-community-edition.html [↑](#footnote-ref-7)
8. https://java.com/pl/download/help/whatis\_java.html [↑](#footnote-ref-8)
9. https://docs.oracle.com/javase/tutorial/java/concepts/index.html [↑](#footnote-ref-9)
10. https://en.wikipedia.org/wiki/Java\_(programming\_language) [↑](#footnote-ref-10)
11. https://marketingwsieci.pl/slownik-e-marketingu/framework [↑](#footnote-ref-11)
12. https://spring.io/projects/spring-boot [↑](#footnote-ref-12)
13. https://en.wikipedia.org/wiki/MongoDB [↑](#footnote-ref-13)
14. https://www.mongodb.com/why-use-mongodb [↑](#footnote-ref-14)
15. https://www.mongodb.com/why-use-mongodb [↑](#footnote-ref-15)
16. https://www.mongodb.com/why-use-mongodb [↑](#footnote-ref-16)
17. https://aws.amazon.com/what-is/api/ [↑](#footnote-ref-17)
18. https://www.redhat.com/en/topics/api/what-is-a-rest-api [↑](#footnote-ref-18)
19. https://www.redhat.com/en/topics/api/what-is-a-rest-api [↑](#footnote-ref-19)
20. https://blog.ericom.com/javascript-puts-websites-at-risk-of-infection [↑](#footnote-ref-20)
21. https://www.lighthouselabs.ca/en/blog/what-is-javascript-used-for [↑](#footnote-ref-21)
22. https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html [↑](#footnote-ref-22)
23. https://developer.mozilla.org/en-US/docs/Web/API/Document\_Object\_Model/Introduction [↑](#footnote-ref-23)
24. https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html [↑](#footnote-ref-24)
25. https://hub.packtpub.com/web-development-react-and-bootstrap [↑](#footnote-ref-25)
26. https://www.tutorialspoint.com/junit/junit\_overview.htm [↑](#footnote-ref-26)
27. https://spring.io/guides/gs/testing-web/ [↑](#footnote-ref-27)