Dokumentacja projektu wykonywanego w ramach zajęć  
BAZY DANYCH I

System zarządzania księgarnią internetową

1. **Projekt koncepcji, założenia ( 4pkt. )**
2. Zdefiniowanie tematu projektu:

Zrealizowanie bazy danych przetrzymującej informacje typowe dla księgarni internetowej z systemem oceniania książek przez klientów, oraz aplikacji webowej, która zasymuluje podstawowe funkcjonalności panelu administratora takiej księgarni.

1. Analiza wymagań użytkownika:

Założeniem projektu jest to, że do aplikacji webowej dostęp będzie miał tylko administrator (będzie ona hostowana lokalnie). Zarządzanie bazą danych (dostępną dla pracowników i klientów księgarni) będzie odbywać się będzie poprzez API napisane w stylu RESTful, co w znacznym stopniu ułatwi dalszy rozwój aplikacji . Takie rozwiązanie poprawi wydajność w zarządzaniu większą ilością danych poprzez ich wizualizacje oraz szybsze wykrywanie i rozwiązywanie problemów.

Administrator w podstawowej wersji aplikacji webowej będzie miał możliwość

* Bezpośredniego wglądu do „surowych danych” w każdej encji.
* Bezpośredniego wprowadzania „surowych danych” do każdej encji.
* Dodawania nowych książek wraz z autorami i wydawnictwem w specjalnie do tego przygotowanym formularzu.
* Dodawania nowych magazynów w specjalnie do tego przygotowanym formularzu.
* Dodawania książek do magazynów w specjalnie do tego przygotowanym formularzu.
* Wgląd do średniej ocen książek i autorów
* Wgląd do ilości książek sprzedanych przez dane wydawnictwo.
* Wgląd do najpopularniejszych książek.
* Wgląd do najpopularniejszych autorów.
* Wglądu do stanu magazynowego książek.
* Informację jeżeli w magazynach jest mniej niż 100 szt. danej książki oraz dane do kontaktu z wydawnictwem.

1. Zaprojektowanie funkcji:

Aby zapewnić integralność danych w tabeli zaprojektowane zostały specjalne funkcje:

* W encji klient e-mail jest wartością unikalną, umożliwia to stworzenie systemu autoryzacji w księgarni.
* W encji książka ISBN jest wartością unikalną.
* Funkcjonalności dostępne w „Zaawansowane Formularze”
  + Podczas dodawania książek do magazynu następuje sprawdzenie, czy dane książki są już przechowywane w magazynie, jeżeli tak, to zostaną one dodane do aktualnej ilości , zamiast tworzyć nowy rekord. Wykonywana jest też podstawowa walidacja danych, jeżeli dane pole nie spełnia wymagań to odpowiednia wiadomość zostaje wysłana do klienta.
  + Podczas dodawania nowego magazynu następuje sprawdzenie, czy podany magazyn, adres oraz kierownik już istnieją, jeżeli tak, to zostaną one użyte. Ma to na celu zredukowanie ilości rekordów w tabelach. Podczas dodawania następują podstawowa walidacja danych.
  + Podczas dodawania książek do magazynu wyboru dokonuje się z listy. Jeżeli dana książka znajduje już się w danym magazynie, to zamiast tworzyć nowy rekord nowe książki zostaną dodane do istniejącej wartości.

1. **Projekt diagramów (konceptualny) ( 4pkt. )**
2. ~~Budowa i analiza diagramu przepływu danych (DFD):~~
3. Zdefiniowanie encji (obiektów) oraz ich atrybutów:

* Autor: autor\_id – klucz podstawowy, imie, nazwisko, data\_urodzenia, opis
* Książka: ksiazka\_id – klucz podstawowy, wydawca\_id – podstawowy klucz obcy, tytul, isbn, gatunek, liczba\_stron, data\_wyadani, cena, opis
* Wydawca: wydawca\_id – klucz podstawowy, nazwa, telefon, strona\_www
* Magazyn: magazyn\_id – klucz podstawowy, adres\_id – podstawowy klucz obcy, kierownik\_id – klucz obcy, nazwa, telefon
* Kierownik: kierownik\_id – klucz podstawowy, imie, nazwisko, telefon, email
* Adres: adres\_id – klucz podstawowy, kraj, miasto, kod\_pocztowy, ulica, nr\_budynku, nr\_mieszkania
* Klient: klient\_id – klucz podstawowy, adres\_id – klucz obcy, imie, nazwisko, telefon, email, haslo
* Ocena: ksiazka\_id – podstawowy klucz obcy, wydawca\_id – podstawowy klucz obcy, klient\_id – podstawowy klucz obcy, ocena, komentarz
* Zamówienie: zamówienie\_id – klucz podstawowy, adres\_id – podstawowy klucz obcy, metoda\_id - podstawowy klucz obcy, ksiazka\_id – podstawowy klucz obcy, klient\_id – podstawowy klucz obcy, wydawca\_id – podstawowy klucz obcy, ilość, data\_zamowienia, status, data\_wyslania
* Metoda Dostawy: metoda\_id – klucz podstawowy, nazwa, cena, czas\_dostawy\_od, czas\_dostawy\_do

1. Zaprojektowanie relacji pomiędzy encjami:

Relacje zostały opisane w kolejnej sekcji w punkcie 7.

1. **Projekt logiczny ( 4pkt. )**
2. Projektowanie tabel, kluczy, indeksów:

Obraz zawierający tekst, paragon, zrzut ekranu

Opis wygenerowany automatycznie

Autor: posiada atrybuty opisujące oraz połączony jest relacją n:n z książką, gdyż jedna książka może mieć wielu autorów, a jeden autor może napisać wiele książek.

Książka: posiada atrybuty opisujące oraz połączona jest relacją n:n z magazynem, ponieważ w jednym magazynie może być wiele książek, a jedna książka może być w wielu magazynach rozsianych na terenie całego kraju. W relacji tej zawarta jest również informacja o ilości sztuk danej książki w magazynie. Posiada również ID wydawcy danej książki.

Wydawca: posiada atrybuty opisujące oraz połączony jest z książką relacją 1 : n, ponieważ jeden wydawca może wydać wiele książek, ale jedna książka może być wydana tylko przez jednego wydawcę.

Magazyn: posiada atrybuty go opisujące, ID adresu oraz ID kierownika. Połączony jest relacją n:n z książką (opisane wyżej).

Kierownik: posiada atrybuty go opisujące, połączony jest relacją 1:n z magazynem, ponieważ jeden kierownik może zarządzać wieloma magazynami znajdującymi się w bliskiej odległości od siebie

Adres: Posiada informacje go opisujące oraz połączony jest relacjami 1:n z magazynem, zamówieniem oraz klientem, ponieważ jeden adres może zostać użyty więcej niż jeden raz.

Klient: Posiada atrybuty go opisujące, ID adresu oraz połączony jest relacją Ocena (n:n) z Książką, ponieważ jeden klient może ocenić wiele książek, a jedna książka może zostać oceniona przez wielu klientów.

Zamówienie: Posiada atrybuty je opisujące, atrybuty opisujące zamawianą książkę, ades oraz klienta.

Metoda Dostawy: Posiada atrybuty go opisujące, oraz połączony jest relacją 1:n z zamówieniem, ponieważ jedna metoda dostawy może zostać użyta w wielu zamówieniach.

1. Słowniki danych:

* Autor:
* autor\_id – klucz podstawowy
* imie – VARCHAR NOT NULL
* nazwisko – VARCHAR NOT NULL
* data\_urodzenia – DATE NOT NULL
* miejsce\_urodzenia -VARCHAR NOT NULL
* Opis - VARCJAR
* Autor\_Książka:
  + autor\_id – podstawowy klucz obcy
  + ksiazka\_id – podstawowy klucz obcy
  + wydawca\_id - podstawowy klucz obcy
* Książka:
  + ksiazka\_id – klucz podstawowy
  + wydawca\_id – podstawowy klucz obcy
  + tytul – VARCHAR NOT NULL
  + isbn – VARCHAR (wersja 10 cyfrowa lub 13 cyfrowa) NOT NULL UNIQUE
  + gatunek – VARCHAR NOT NULL
  + liczba\_stron – INTEGER NOT NULL
  + data\_wydania – DATE NOT NULL
  + cena – FLOAT NOT NULL
  + opis - VARCHAR
* Wydawca:
  + wydawca\_id - klucz podstawowy
  + nazwa – VARCHAR NOT NULL
  + telefon – VARCHAR (3cyfry-3cyfry-3cyfry) NOT NULL
  + strona\_www - VARCHAR
* Magazyn\_Książka:
  + ksiazka\_id – podstawowy klucz obcy
  + wydawca\_id – podstawowy klucz obcy
  + magazyn\_id – podstawowy klucz obcy
  + adres\_id - podstawowy klucz obcy
  + ilosc – INTEGER NOT NULL
* Magazyn:
  + magazyn\_id – klucz podstawowy
  + adres\_id – podstawowy klucz obcy
  + kierownik\_id – podstawowy klucz obcy
  + nazwa – VARCHAR NOT NULL
  + telefon – VARCHAR (3cyfry-3cyfry-3cyfry) NOT NULL
* Kierownik:
  + kierownik\_id – klucz podstawowy
  + imie – VARCHAR NOT NULL
  + nazwisko - VARCHAR NOT NULL
  + telefon – VARCHAR (3cyfry-3cyfry-3cyfry) NOT NULL
  + email - VARCHAR
* Adres:
  + adres\_id – klucz podstawowy
  + kraj - VARCHAR NOT NULL
  + miasto - VARCHAR NOT NULL
  + kod\_pocztowy - VARCHAR NOT NULL
  + ulica - VARCHAR NOT NULL
  + nr\_budynku - VARCHAR NOT NULL
  + nr\_mieszkania - VARCHAR NOT NULL
* Klient:
  + klient\_id – klucz podstawowy
  + adres\_id – klucz obcy
  + imie - VARCHAR NOT NULL
  + nazwisko - VARCHAR NOT NULL
  + telefon – VARCHAR (9 cyfr) NOT NULL
  + email - VARCHAR NOT NULL UNIQUE
  + haslo - VARCHAR NOT NULL
* Ocena:
  + ksiazka\_id - podstawowy klucz obcy
  + wydawca\_id - podstawowy klucz obcy
  + klient\_id - podstawowy klucz obcy
  + ocena - VARCHAR NOT NULL
  + komentarz - VARCHAR NOT NULL
* Zamówienie
  + zamowienie\_id - podstawowy klucz obcy
  + adres\_id - podstawowy klucz obcy
  + metoda\_id - podstawowy klucz obcy
  + ksiazka\_id - podstawowy klucz obcy
  + klient\_id - podstawowy klucz obcy
  + wydawca\_id - podstawowy klucz obcy
  + ilosc – INTEGER NOT NULL
  + data\_zamowienia – DATE NOT NULL
  + status – VARCHAR NOT NULL („Przygotowywane” , „Wysłane”)
  + data\_wyslania – DATE NOT NULL
* Metoda\_dostawy:
  + metoda\_id – klucz podstawowy
  + nazwa – VARCHAR NOT NULL
  + cena – VARCHAR NOT NULL
  + czas\_dostawy\_od – INTEGER NOT NULL
  + czas\_dostawy\_do – INTEGER NOT NULL

1. Zaprojektowanie operacji na danych:

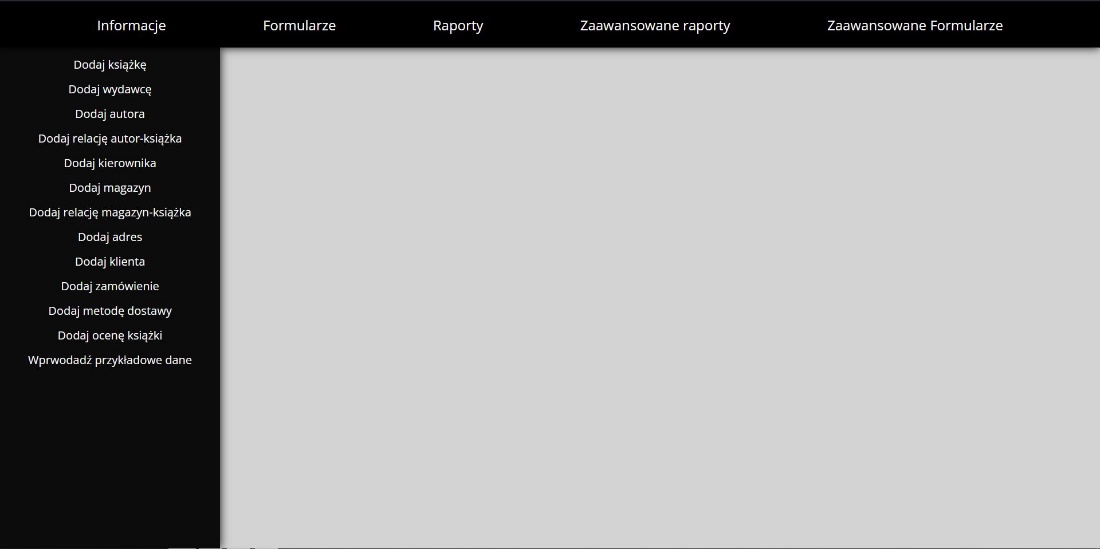
Użyte widoki oraz funkcje znajdują się w folderze SQL – plik ViewsAndFunctions.sql

1. **Projekt funkcjonalny ( 4pkt. )**
2. Strona główna z panelem sterowania

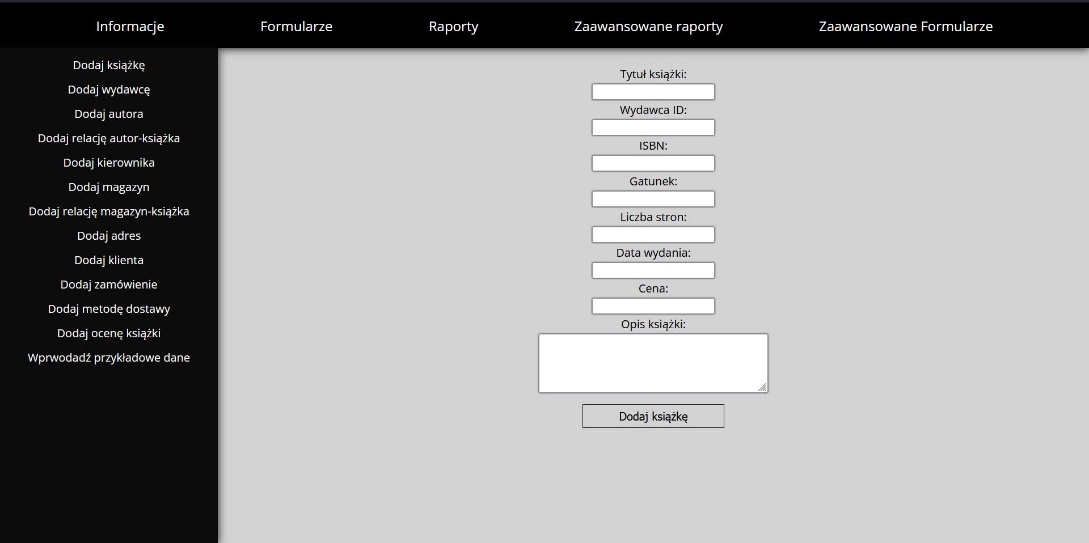
Obraz zawierający tekst

Opis wygenerowany automatycznie

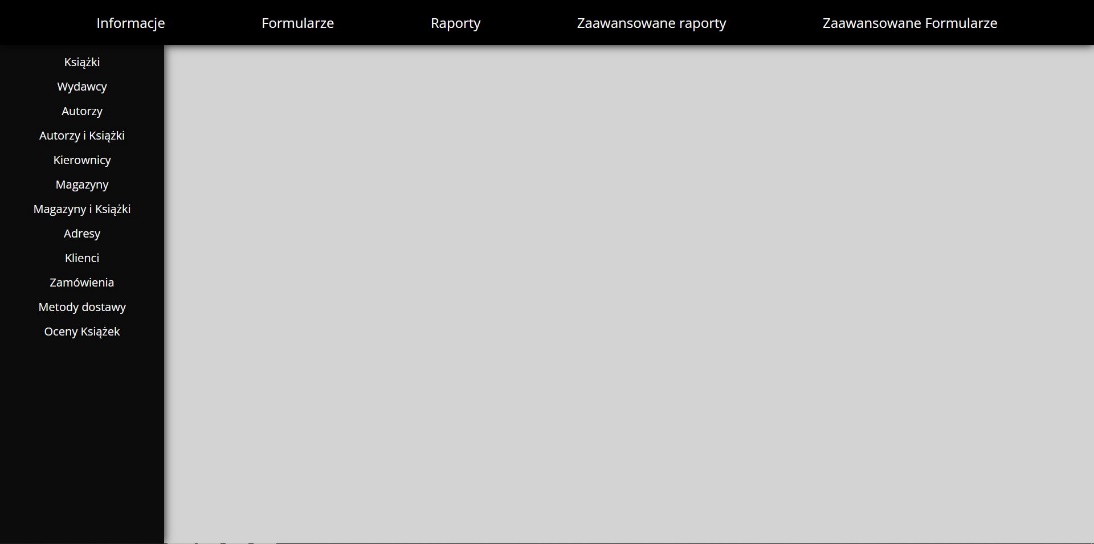
1. Przycisk Formularze powoduje przejście do panelu nawigacyjnego z formularzami, które służą do wprowadzania „surowych danych” do bazy.



Przykładowy formularz:



1. Przycisk Raporty powoduje przejście do panelu nawigacyjnego z raportami, które wyświetlają zawartość poszczególnych tabeli w bazie.

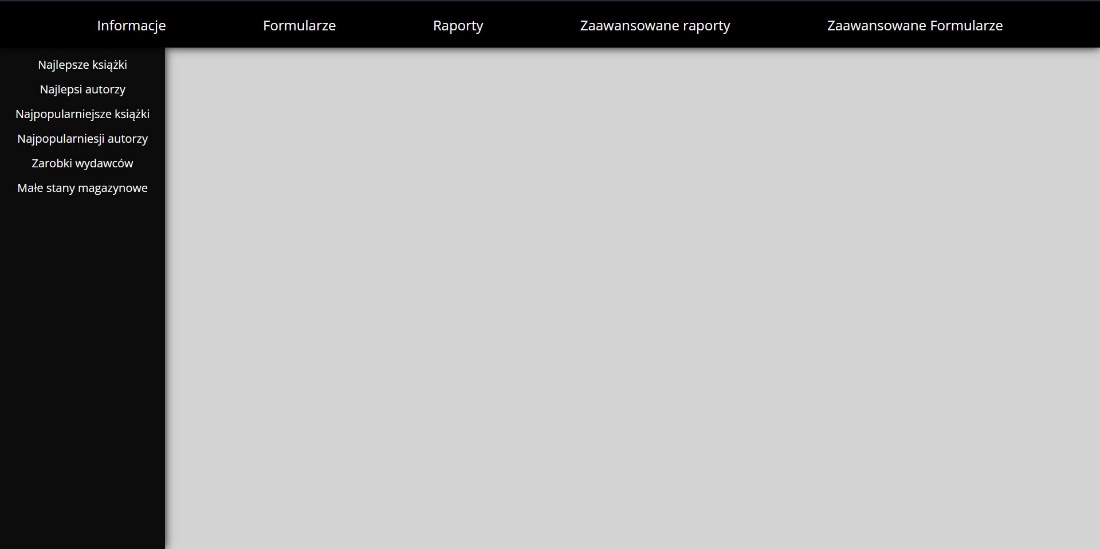


Przykładowy raport:

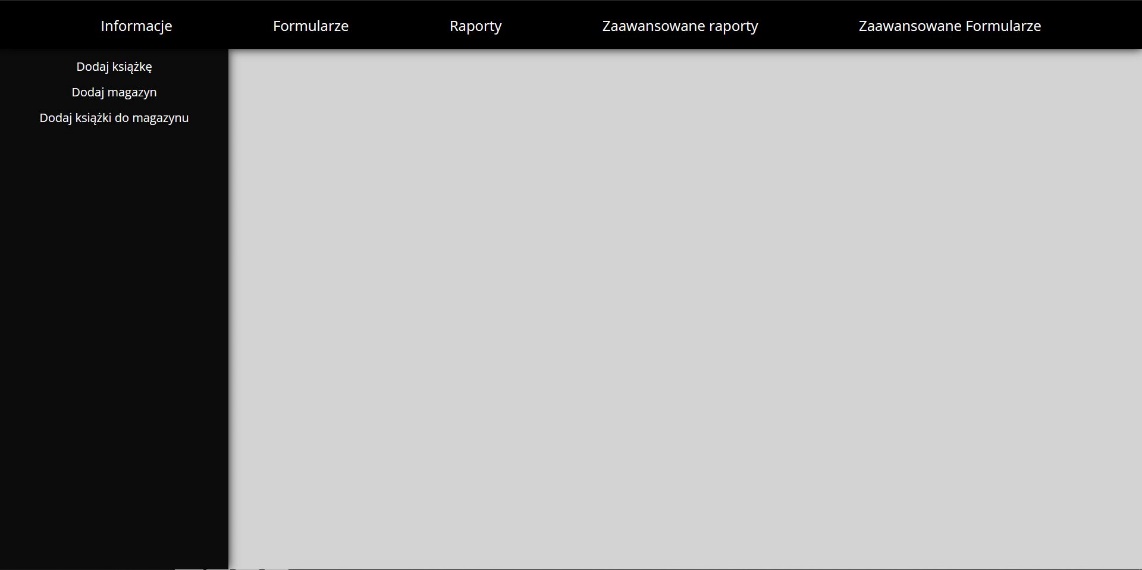
Obraz zawierający stół

Opis wygenerowany automatycznie

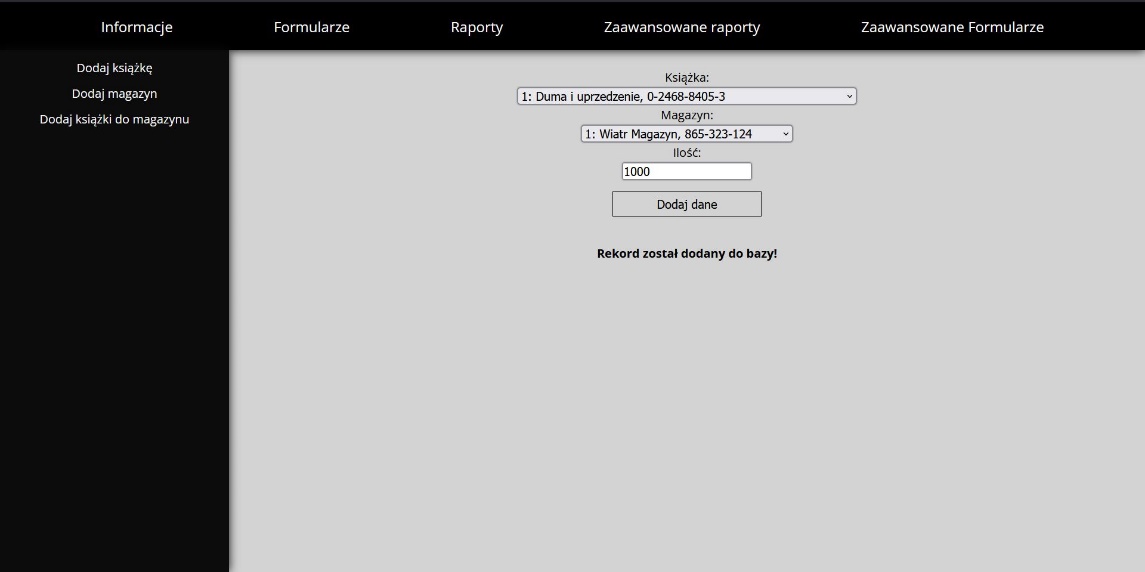
1. Przycisk zaawansowanie raporty powoduje przejście do panelu nawigacyjnego z raportami, które wyświetlają najważniejsze informacje potrzebne administratorowi księgarni internetowej.



1. Przycisk Zaawansowane Formularze powoduje przejście do formularzy, które gwarantują wprowadzenie danych do bazy w sposób względnie bezpieczny. Używanie ich powoduje ograniczenie duplikacji danych w bazie, gdyż korzystają one z danych już wprowadzonych w bazie, jeżeli pokrywają się z nowymi danymi.



1. Użytkownik końcowy jest informowany, czy dane do bazy zostały wprowadzone pomyślnie.



1. Użytkownik końcowy jest informowany, jeżeli podczas wprowadzania danych pojawił się błąd, oraz jest informowany o rodzaju błędu.

Obraz zawierający tekst

Opis wygenerowany automatycznie

1. **Dokumentacja (4pkt. )**
2. Wprowadzanie danych:

Dane do każdej tabeli, oprócz tabeli autor\_ksiazka, muszą zostać wprowadzone ręcznie.

1. Dokumentacja użytkownika:

Jak uruchomić aplikację:

1. Pierwszym krokiem jest stworzenie bazy danych PostgreSQL, której struktura znajduje się w folderze SQL w pliku Structure.sql oraz dodanie do niej widoków i funkcji z pliku ViewsAndFunctions.sql
2. Należy pobrać ze strony <https://nodejs.org/en/> node.js (najlepiej wersję 16.13.2 – na niej tworzony był projekt)
3. Z poziomu konsoli wejść do folderu backend oraz zainstalować potrzebne moduły za pomocą komendy **npm install**
4. Przed uruchomieniem serwera należy skonfigurować dane logowania do bazy PosgtreSQL. Plik konfiguracyjny znajduje się w folderze backend: config.js:

dbHost – Nazwa hosta

dbPort - Port, na którym działa baza danych

dbUser - Nazwa użytkownika

dbPass: - Hasło do bazy danych

dbName: - Nazwa bazy danych

1. Kolejnym krokiem jest uruchomienie serwera: **node .\server.js**

Serwer należy uruchomić na porcie 8000, w innym wypadku należy ręcznie zmienić port we wszystkich url w części frontend-owej

1. Następnie w nowej konsoli należy wejść do folderu frontend i użyć polecenia **npm install** w celu pobrania wymaganych modułów
2. Następnie należy użyć komendy **npm start** w celu uruchomienia serwera. Zostanie on uruchomiony na porcie 3000.

Do aplikacji dołączone są przykładowe dane, które można wprowadzić bezpośrednio w bazie danych (plik Data.sql w folderze SQL) lub poprzez aplikację webową (Formularze -> Przykładowe dane). Najlepiej dane wprowadzić z poziomu aplikacji webowej w celu uniknięcia problemów z kodowaniem.

**Jeżeli już raz się wprowadziło przykładowe dane nie należy robić tego ponownie! Operacja ta nie jest w żadnym stopniu zabezpieczona w aplikacji webowej.**

1. Opracowanie dokumentacji technicznej:

Aplikacja została napisana w stylu RESTful, istnieje więc podział na backend, który komunikuje się z frontendem poprzez zapytania GET oraz POST.

* Frontend:

Stworzony został przy użyciu REACT’a (wersja 17.0.2). Każdy komponent jest odpowiedzialny za osobną stronę w aplikacji

* + Index.js – jest to plik startowy, odpowiada za inicjalizacje aplikacji
  + App.js – służy jako pośrednik w nawigowaniu aplikacją

Folder layout:

* + Layout.js – odpowiedzialny jest za układ strony
  + MainNavigation.js – główny panel nawigacyjny aplikacji
  + Information.js – strona informacyjna
  + FormsNavigation.js – nawigacja po prostych formularzach, które pozwalają wprowadzić dane bezpośrednio do bazy
  + ReportsNavigation.js – nawigacja po prostych raportach, które pokazują zawartość bazy danych
  + AdvancedFormNavigation.js – nawigacja po zaawansowanych formularzach, dane do bazy danych są wprowadzane poprzez funkcje
  + AdvancedReportsNavigation.js nawigacja po zaawansowanych raportach, dane pochodzą z widoków
  + \*.module.css – Pliki o tej samej nazwie jak komponenty odpowiadają za ich wygląd

Folder reports:

Każdy plik odpowiedzialny jest za wysyłanie zapytania do serwera metodą Get i wyświetlenia otrzymanych informacji w postaci graficznej.

* + AddressesReport.js – „Surowe dane” z tabeli adres
  + AuthorsBooksrepor.js – „Surowe dane” z tabeli autor\_ksiazka
  + BooksReport.js – „Surowe dane” z tabeli ksiazka
  + ClientReport.js – „Surowe dane” z tabeli klient
  + DeliveryMethodsReport.js – „Surowe dane” z tabeli metoda\_destowy
  + ManagersReport.js – „Surowe dane” z tabeli kierownik
  + OrdersReport.js – „Surowe dane” z tabeli zamowienie
  + PublisherReport.js – „Surowe dane” z tabeli wydawca
  + ScoreReport.js – „Surowe dane” z tabeli ocena
  + WarehousesBooksReport.js – „Surowe dane” z tabeli magazyn\_ksiazka
  + WareHousesReport.js – „Surowe dane” z tabeli magazyn
  + Reports.module.css – wygląd wyświetlonych danych

Folder forms:

Każdy plik odpowiedzialny jest za wysyłanie danych do serwera metodą POST i wyświetlenia otrzymanej odpowiedzi w postaci graficznej pod formularzem.

* + AddAddressForm.js – Dodanie “surowych danych” do tabeli adres
  + AddAuthorBookForm.js – Dodanie “surowych danych” do tabeli autor\_ksiazka
  + AddAuthorForm.js – Dodanie “surowych danych” do tabeli autor
  + AddBookForm.js – Dodanie “surowych danych” do tabeli ksiazka
  + AddClientForm.js – Dodanie “surowych danych” do tabeli klient
  + AddDeliveryMethodForm.js – Dodanie “surowych danych” do metoda\_dostawy
  + AddExemapleDataForm.js – Dodanie przykładowych danych do wszystkich tabel w bazie danych
  + AddManagerForm.js – Dodanie “surowych danych” do tabeli kierownik
  + AddOrderForm.js – Dodanie “surowych danych” do tabeli zamówienie
  + AddPublisherForm.js – Dodanie “surowych danych” do tabeli wydawca
  + AddScoreForm.js – Dodanie “surowych danych” do tabeli ocena
  + AddWarehouseBookForm.js – Dodanie “surowych danych” do tabeli magazyn\_ksiazka
  + AddWarehouseForm.js – Dodanie “surowych danych” do tabeli magazyn
  + AddForm.module.css – wygląd formularzy

Folder advanced-reports:

Każdy plik odpowiedzialny jest za wysyłanie zapytania do serwera metodą Get i wyświetlenia otrzymanych informacji w postaci graficznej.

* + BestAuthorsReport.js – Wyświetla dane na podstawie widoku
  + BestBooksReport.js – Wyświetla dane na podstawie widoku
  + LowStocksReport.js – Wyświetla dane na podstawie widoku
  + PopularAuthorsReport.js – Wyświetla dane na podstawie widoku
  + PublishersEarningsReport.js – Wyświetla dane na podstawie widoku
  + PopularBooksReport.js – Wyświetla dane na podstawie widoku
  + Reports.module.css – wygląd danych
  + advanced-forms:

Każdy plik odpowiedzialny jest za wysyłanie danych do serwera metodą POST i wyświetlenia otrzymanej odpowiedzi w postaci graficznej pod formularzem

* + AdvancedAddBook.js – Dodanie książki wraz z jej autorem i wydawcą
  + AdvancedAddBooksToWarehouse.js – Dodanie książek do magazynu
  + AdvancedWarehouseForm.js – Dodanie magazynu wraz z kierownikiem i adresem
  + AddForm.module.css – wygląd formularzy
* Backed:

Stworzony przy użyciu technologii node.js (wersja 16.32). Dodatkowe moduły użyte do budowy aplikacji to: cors(wersja 2.8.5), date-and-time (wersjia 2.1.0), express (wersja 4.7.12), nodemon (wersja 2.0.15), pg (wersja 8.7.1). Aplikacja została podzielona na moduły

* + server.js – odpowiedzialny za inicjalizację bazy danych oraz przekierowywanie zapytań do odpowiednich modułów
  + config.js – przechowuje konfiguracje dla bazy danych i express
  + response.js – moduł odpowiedzialny za przechowywanie wiadomości wysyłanej przez serwer w ramach odpowiedzi na zapytania typu POST

W folderze routes znajdują się wyspecjalizowane moduły odpowiedzialne za odbieranie zapytań, odczytywanie danych z bazy oraz wysyłanie odpowiedzi klientowi

* + address.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli adres
  + advanced-forms.js – odpowiedzialny za odbieranie danych z formularzy metodą POST i zapisywanie ich do bazy danych.
  + author-book.js - – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli autor\_ksiazka
  + author.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli autor
  + book.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli ksiazka
  + client.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli klient
  + delivery-method.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli metoda\_dostawy
  + examples.js – odpowiedzialny za dodanie wcześniej przygotowanych danych przykładowych do bazy, jeżeli otrzyma odpowiednie zapytanie metodą POST
  + manager.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli kierownik
  + order.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli zamówienie
  + publisher.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli wydawca
  + score.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli ocena
  + views.js – odpowiedzialny za realizację zapytań typu GET na widokach w bazie danych
  + warehouse-book.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli magazyn\_ksiazka
  + warehouse.js – odpowiedzialny za realizację zapytań POST oraz GET, które dotyczą tylko tabeli magazyn

1. Wykaz literatury:
   * <https://stackoverflow.com/>
   * <https://nodejs.org/en/docs/>
   * <https://docs.npmjs.com/>
   * <https://pl.reactjs.org/docs/getting-started.html>
   * <https://node-postgres.com/>
   * <https://www.postgresql.org/docs/>