POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

PROJEKT Z BAZ DANYCH

Baza danych książek

księgarnia internetowa

Termin zajęć: Środa, 9:15–11:00

AUTORZY:
Kinga Kraczkowska
Indeks: 218589
E-mail: 218589@student.pwr.edu.pl
Jędrzej Kozal
Indeks: 218557
E-mail: 218557@student.pwr.edu.pl

PROWADZĄCY ZAJĘCIA:
dr inż. Roman Ptak, W4/K9

Wrocław, 2017 r.

Spis treści:

1. Wstęp

1.1. Cel projektu

Cele związane z projektem:

- Stworzenie bazy danych dla średniej wielkości księgarni internetowej ma poprawić jakości oferowanych usług oraz usprawnienie zarządzania asortymentem.
- Ułatwienie użytkownikom korzystania z księgarni internetowej.

Cele osobiste:

- Zdobycie podstawowych umiejętności w zakresie projektowania i wykonywania baz danych oraz aplikacji webowych.
- Zapoznanie się z podstawowymi narzędziami do tworzenia baz danych.

1.2. Zakres projektu

- W tracie projektu przewiduje się realizację aplikacji sieciowej, z wykorzystaniem bazy danych. Przewiduje się realizację funkcjonalności przedstawionych na rys. 2 w dziale 3.1.2 Model logiczny i normalizacja.
- Projektowana baza będzie zawierać dane dotyczące parametrów i ilość dostępnych książek. Baza danych będzie obejmować informacje na temat konta klientów, zamówienia wraz z ich statusem. Na podstawie takich informacji można podejmować odpowiednie kroki zarządcze oraz kontrolne.
- Realizacja transakcji bankowych i przesyłek kurierskich nie wliczają się w zakres projektu, ale należy projektować system w taki sposób, aby ułatwić rozwijanie i dodawanie nowych funkcjonalności.

2. Analiza wymagań

2.1. Opis działania i schemat logiczny systemu

System ma być wykonany dla księgarni internetowej, a zaimplementowane rozwiązania powinny wspomagać współpracę z firmą kurierską w zakresie realizacji zamówień, oraz umożliwiać dostęp do podstawowych funkcjonalności księgarni internetowej jak realizacja zamówień oraz uiszczanie opłat w bezpieczny sposób.

2.2. Wymagania funkcjonalne

- Jednym z użytkowników jest osoba obsługująca bazę danych, czyli sprzedawca lub administrator sklepu. Ma dostęp do wszystkich parametrów książek oraz ilości i ceny jak i informacje o zamówieniach.

- Klient zalogowany może przeglądać dostępny asortyment księgarni (nazwa, ilość, cena książek), składać zamówienia, oraz pisać i czytać recenzje do książek.
- Klient niezalogowany może czytać recenzje do książek oraz przeglądać asortyment
- Administrator posiada prawo do wprowadzania zmian w bazie: usuwania, dodawania oraz edytowania książek i użytkowników.
- Dodatkowo administratorzy będą mogli aktualizować informacje na temat stanu magazynu księgarni.
- Zamówione książki powinny trafić do schowka, w którym pozostają, dopóki nie zapłaci za nie. Po zapłacie do oddziału księgarni oraz do firmy kurierskiej powinny zostać wysłane odpowiednie powiadomienia.

2.3. Wymagania niefunkcjonalne

Ważnym aspektem jest estetyka oraz intuicyjna obsługa umożliwiająca bezproblemowe wyszukiwanie rekordów z bazy. Priorytetem przy tworzeniu tego typu aplikacji powinny być bezpieczeństwo oraz szybkość. Przewiduje się że serwer będzie działał na systemie operacyjnym Ubuntu, natomiast użytkownicy będą mogli się logować z dowolnego systemu operacyjnego. Narzędzia dobrano do rozmiaru bazy oraz do wymagań, które ma spełniać dany system.

2.3.1. Wykorzystywane technologie i narzędzia

- MySQL wolnodostępny system zarządzania relacyjnymi bazami danych, rozwijany jest przez firmę Oracle
- phpMyAdmin – za pomocą przeglądarki internetowej jako narzędzie administracyjne
- Django – framework w języku Python do tworzenia aplikacji webowych.

2.3.2. Wymagania dotyczące rozmiaru bazy danych

Szacuje się liczba klientów nie przekroczy 10 tys., a liczba książek nie przekroczy 100 tys. Zakładając że każda książka średnio posiada 3 recenzje liczbę recenzji można oszacować na 300 tys. Podane szacunki są oparte na podstawie danych dotyczących innych księgarni internetowych znalezionych
w Internecie.

2.3.3. Wymagania dotyczące bezpieczeństwa systemu

Kluczowym wymaganiem w tego typu systemach jest umożliwienie przeprowadzenia bezpiecznych transakcji bankowych – odsyłanie do systemu umożliwiającego realizację transakcji (takie jak PayU bądź systemy związane z płatnościami kartą kredytową).

Podstawowym sposobem na zapewnienie ochrony danych osobowych jest zabezpieczenie odpowiednich kont loginami i hasłami. Dane użytkowników takie jak imię, nazwisko, adres, adres email czy numer telefonu powinny być odpowiednio chronione – inny użytkownik nie powinien mieć do nich dostępu.

Ważne są również zabezpieczenia np. ograniczające dostęp klientom do usuwania lub dodawania asortymentu. w tym celu wyróżnia się konta specjalne - adminów i zwyczajnych użytkowników.

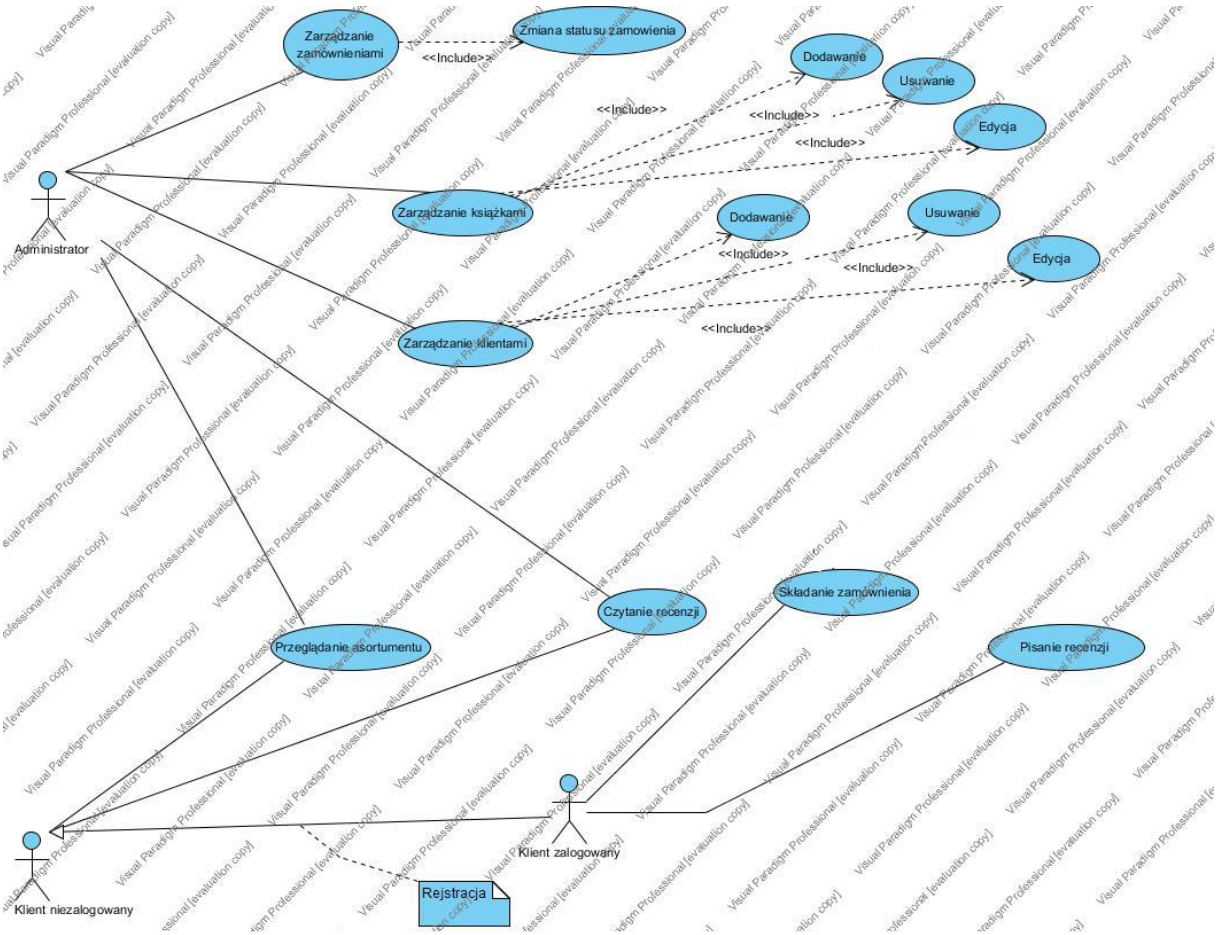
2.4. Przyjęte założenia projektowe

W ramach projektu ma zostać zrealizowana internetowa aplikacja z dostępem do bazy danych. Aplikacja wraz z bazą mają realizować cechy funkcjonalne wymienione w sekcji 2.

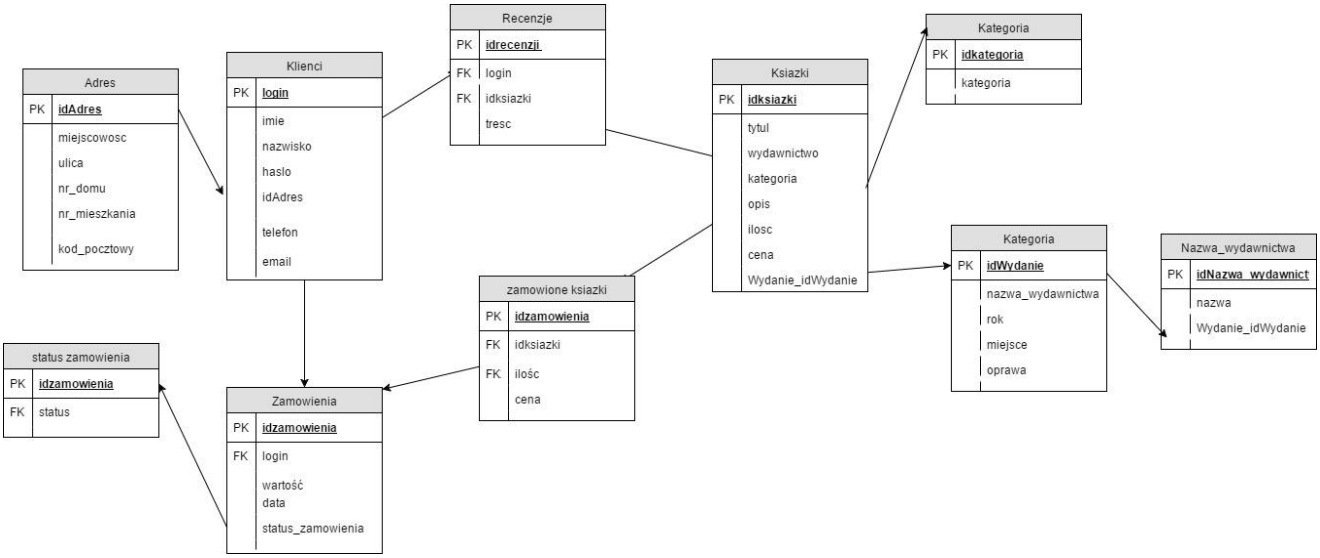
3. Projekt systemu

3.1. Projekt bazy danych

3.1.1. Analiza rzeczywistości i uproszczony model konceptualny

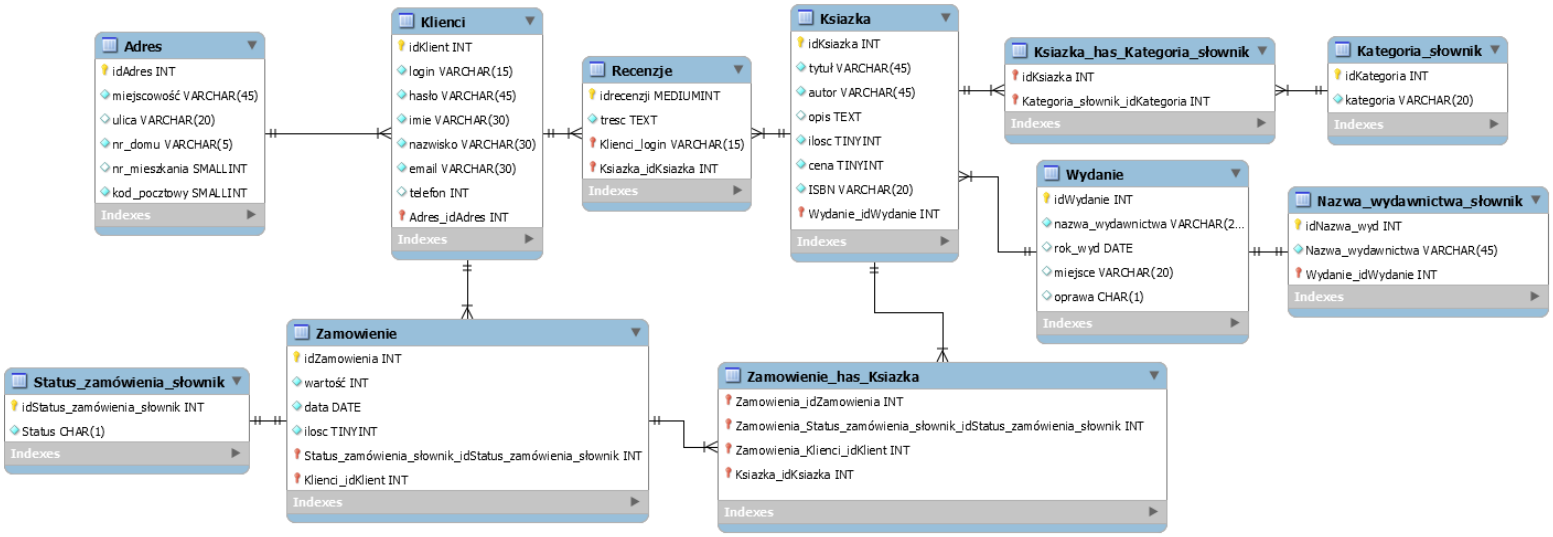


Rysunek 1 Model przypadków użycia



Rysunek 2 Diagram związków, model konceptualny

3.1.2. Model logiczny i normalizacja, model fizyczny i ograniczenia integralności danych



Rysunek 3 Model logiczny i fizyczny

Model sprowadzono do trzeciej postaci normalnej, przez co modele logiczny i fizyczny są tożsame.

Pomiędzy Klientem a recenzjami i zamówieniami jest relacja jeden-do-wielu, ponieważ recenzja może mieć jednego autora, a zamówienie może złożyć jedna osoba. Pomiędzy książką a recenzją jest relacja jeden do wielu, ponieważ może istnieć wiele recenzji tej samej książki. Pomiędzy Zamówieniem a książką jest relacja wiele do wielu. Dodatkowo wykorzystano tabele słownikowe: nazwa wydawnictwa, kategoria oraz status zamówienia.

Kluczami w tabelach są osobne identyfikatory, ponieważ przeszukiwanie po indywidualnych numerach będących zmiennymi typu INT jest łatwiejsze i bardziej wydajne niż przeszukiwanie po bardziej skomplikowanych typach danych np. varchar.

3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych

Zakłada się wykorzystanie indeksu na kolumny tytuł oraz autor tabeli Książka. Są to dwa najczęściej stosowane kryteria przy wyszukiwaniu książek, więc zwiększenie wydajności sortowania oraz wyszukiwania jest kluczowe dla prawidłowego działania systemu.

Przewiduje się wykorzystanie trzech widoków, odpowiedzialnych za prezentacje danych dotyczących książki, zamówienia oraz klienta. Ponieważ zastosowano podział na tabele aby zachować postać normalną, wygodnie jest zbierać dane dla aplikacji w widokach łączących wszystkie dostępne informacje dotyczące danych obiektów z informacjami wydzielonymi z poszczególnych tabel oraz tabelami słownikowymi.

Tabela 1 Pola wybrane do utworzenia widoków.

Nazwa widoku	Nazwa wykorzystanych tabel	Wybrane kolumny
widokKsiazka	Ksiazka	tytul, autor, cena
widokKlienci	Klienci	login, imie, nazwisko
widokZamowienie	Zamowienie	wartość, Status_zamówienia_słownik_idStatus_zamówienia_słownik, data
	Klienci	login, imie, nazwisko, email

Triggery powinny nadawać id obiektowi w momencie zarejestrowania akcji INSERT we właściwej dla obiektu tabeli – wykonanie przed zdarzeniem. Trigger tego typu w przypadku działania zamówienia powinien ustawić odpowiedni status zamówienia korzystając z tabeli słownikowej.

3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

Podstawowym środkiem ochrony jest ograniczenie dostępu normalnym użytkownikom. Mają oni dostęp jedynie do funkcjonalności przedstawionych w modelu konceptualnym (rys. 1). Aby zalogować się należy podać login oraz hasło. Hasła mają być przechowywane w bazie zaszyfrowane funkcją skrótu SHA-2.

Użytkownicy Dwie podstawowe grupy to użytkownik i admin. MySQL umożliwia nadawanie określonych ról (np. MaintenanceAdmin, ProcessAdmin, UserAdmin, SecurityAdmin, MonitorAdmin, DBManager, DBDesigner, ReplicationAdmin, BackupAdmin oraz DBA). Możliwe jest również wybranie przywilejów (privileges) przypisanych do konkretnego konta. Zakłada się że admin będzie posiadał uprawnienia DBA, czyli zagwarantowane wszystkie przywileje, co znacząco ułatwia projektowanie bazy.

Uprawnienia użytkownika powinny być zróżnicowane w zależności od tabel, oraz od tego czy użytkownik jest zalogowany, czy nie.

Tabela 2 Role przyznawane użytkownikom w zakresie podanych tabel.

Użytkownik	Adres	Klient	Recenzje	Książka	Wydawnictwo	Zamówienia	Zamówione książki
Klient zalogowany	owner	owner	table.insert	table.readonly	table.readonly	table.insert	table.readonly
Klient niezalogowany	-	-	table.readonly	table.readonly	table.readonly	-	-
Admin	owner	owner	owner	owner	owner	owner	owner

Do pozostałych tabel użytkownicy nie mają dostępu, ponieważ zostały stworzone na użytek systemu – są to tabele słownikowe.

Dla roli wyszczególnionych w tabeli 1 przewiduje się następujące przywileje:

- owner: INSERT, SELECT, UPDATE, CREATE, DELETE
- table.readonly: SELECT
- table.insert: INSERT, SELECT, UPDATE

3.2. Projekt aplikacji użytkownika

3.2.1. Architektura aplikacji i diagramy projektowe

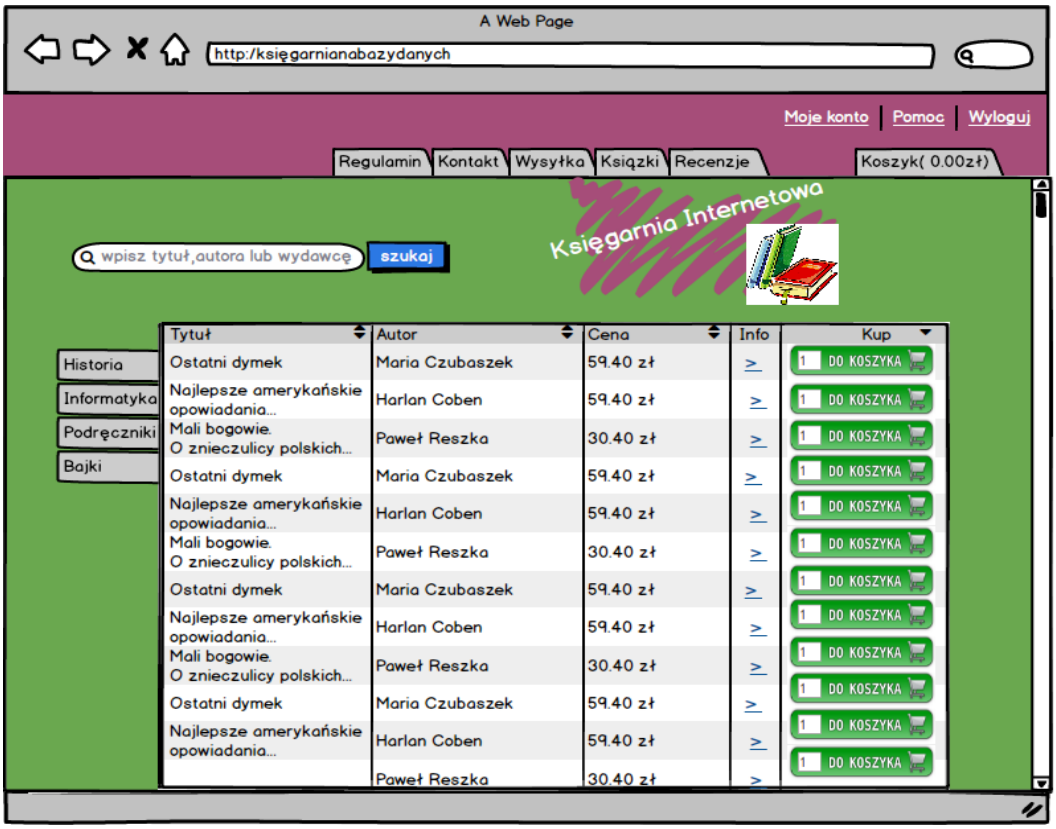
Wybór architektury jest mocno związany z wybranymi narzędziami. Powszechnie stosowany jest wzorzec projektowy MVC (Model-View-Controller) i jest uznawany za najlepszą metodę do tworzenia aplikacji webowych. Jako framework do realizacji projektu wybrano Django, które przestrzega modelu MVC z małymi różnicami. Warstwa C (Controller) jest obsługiwana przez sam framework, dlatego Django jest określane jako MTV framework (Model-Template-View):

- Model – warstwa dostępu do danych, zawiera wszystkie informacje o danych: jak zyskać dostęp do danych, jak dane się zachowują, jakie są relacje między danymi
- Template – warstwa prezentacji. Zawiera informacje na temat jak informacje powinny być wyświetlane na stronie internetowej.
- View – warstwa logiki biznesowej. Odpowiada za logikę która ma dostęp do modelu i odnosi się do poszczególnych szablonów. Stanowi pewnego rodzaju pomost pomiędzy modelem a warstwą prezentacji.

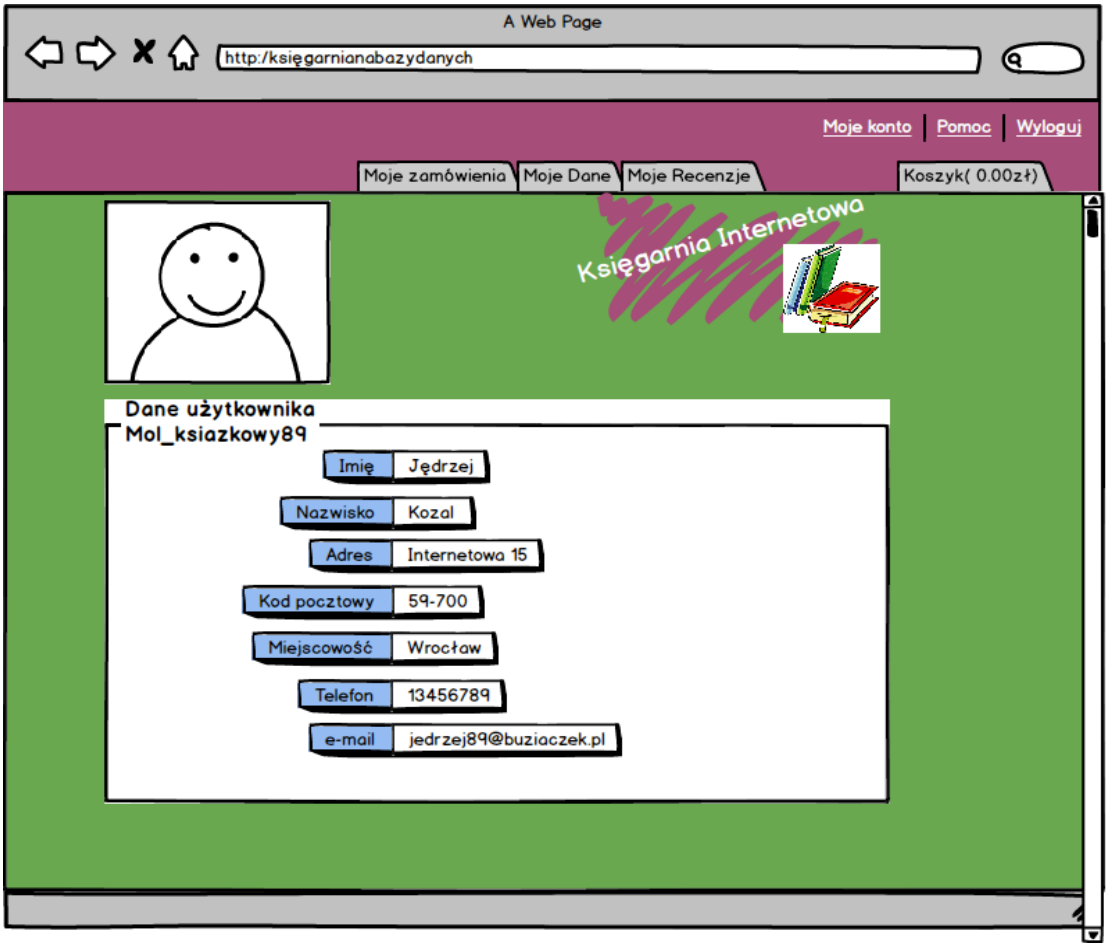
3.2.2. Interfejs graficzny i struktura menu

Graficzny interfejs użytkownika do bazy danych (strona klienta) ma umożliwić klientowi składanie zamówień, oglądanie asortymentu oraz czytanie i dodawanie opinii. Ważna jest intuicyjność obsługi czyli czytelne i wymowne przyciski pozwalające np. szybko dodać interesujący produkt do koszyka oraz widoczna cena. Prawidłowy dobór kolorów ma istotne znaczenie dla wyników sprzedażowych. Kolory wpływają na klimat i charakter sklepu, a zatem powinny idealnie współgrać z ofertą. Zieleń z purpurą w tej sytuacji prezentować się będzie wyglądać gustownie i z klasą.

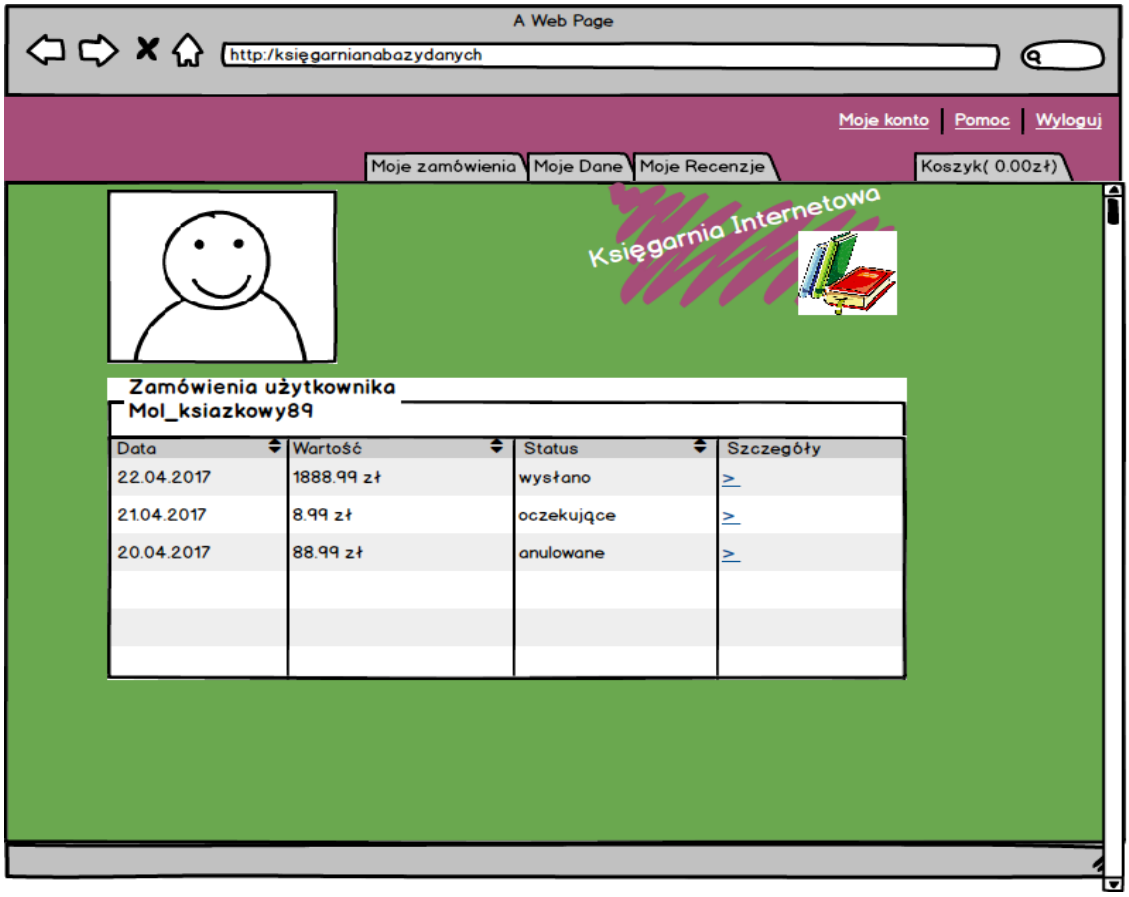
Rysunek 4 Prototyp aplikacji. Strona logowania



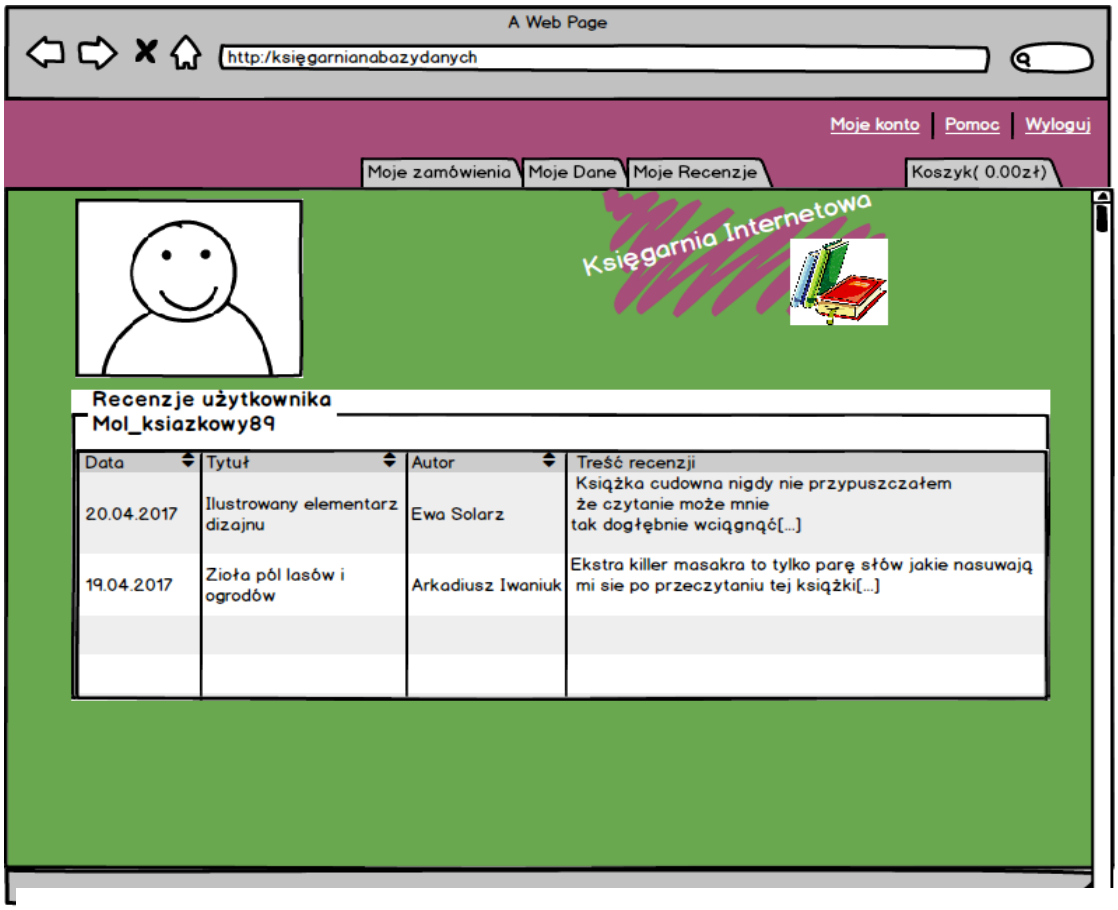
Rysunek 5 Prototyp aplikacji. Strona z asortymentem



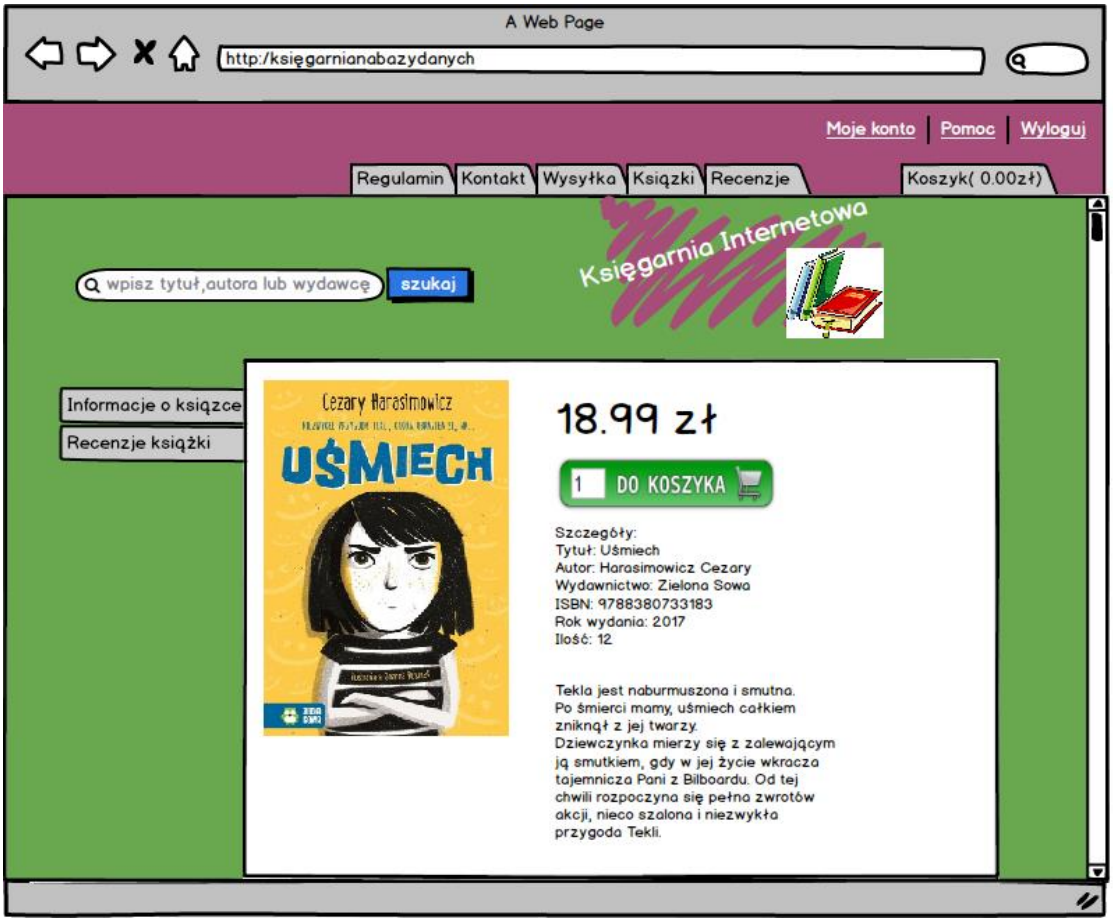
Rysunek 6 Prototyp aplikacji. Użytkownik zalogowany. Strona z danymi użytkownika



Rysunek 7 Prototyp aplikacji. Użytkownik zalogowany. Strona z zamówieniami użytkownika



Rysunek 8 Prototyp aplikacji. Użytkownik zalogowany. Strona z recenzjami użytkownika



Rysunek 9 Prototyp aplikacji. Informacje o książce

3.2.3. Projekt wybranych funkcji systemu

W projekcje przewidziane są następujące funkcje aktualizujące: rejestracja, dodawanie, edycja i usuwanie asortymentu, zarządzanie użytkownikami, zmiana statusu zamówienia, dodawanie recenzji, składanie zamówienia oraz jego potwierdzenie.

Istotne są również funkcje zapytań: wyszukiwanie książek po parametrze (autor, tytuł, itp.). sortowanie asortymentu wg parametru np., cena lub alfabetycznie, umożliwienie administratorowi wyszukiwanie klienta, oraz wgląd w stany zamówień i sortowanie ich po dacie lub stanie.

Administrator ma możliwość wprowadzać danych do bazy danych, aktualizować i usuwać dane z bazy danych, wyszukiwać dane w bazie danych.

3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych

Sposób podłączania do bazy danych jest ściśle związany z wybranym frameworkiem. Django wspiera komunikację z MySQL wersja 5.5 lub nowsza. Ponadto wymusza wykorzystanie kodowania Unicode (UTF-8). Aktualnie zalecanym API do komunikacji Django z MySQL jest mysqlclient.

Niezależnie od wybranego systemu bazodanowego Django korzysta ze trwałych połączeń z bazą danych, co pozwala na uniknięcie odnawiania połączenia za każdym razem gdy wysyłane jest zapytanie. Maksymalny czas trwania połączenia z bazą jest kontrolowany przez marametr CONN_MAX_AGE, który określa liczbę sekund przez które trwa połączenie. Wstępnie szacuje

się wartość parametru na 10 minut, ale dopiero testy aplikacji umożliwią wyznaczenie właściwej wartości.

3.2.5. Projekt zabezpieczeń na poziomie aplikacji

Aplikacja łączy się z bazą, a następnie dokonuje w niej uwierzytelnienia przy użyciu danych uwierzytelniających. Szyfrowanie haseł odbywa się na poziomie aplikacji funkcją skrótu SHA-2. w bazie danych ma być przechowywana zaszyfrowana wersja hasła.Aplikacja nie będzie korzystać z apletów, lokalizacji użytkownika, flash player, więc dodatkowe uprawnienia w przeglądarce nie będą wymagane.

4. Implementacja systemu baz danych

4.1. Tworzenie tabel i definiowanie ograniczeń

Do generacji bazy danych wykorzystano skrypt generowany przez MySQL Workbench, w którym przygotowano model fizyczny bazy przedstawiony we wcześniejszych podpunktach. Jest to wygodny sposób na pominięcie żmudnych i powtarzalnych czynności, jednocześnie odpowiednio wygenerowany skrypt daje możliwość programiście samemu decydować o szczegółach takich jak mechanizmy przetwarzania bazy. Sam skrypt zawiera ponad 300 linii, więc nie umieszczano go w tej sekcji dokumentacji.

Fragment kodu realizujący tworzenie tabeli Książka:

```
-- -----------------------------------------------------
-- Table `Ksiegarnia`.`Ksiazka`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Ksiegarnia`.`Ksiazka` ;
CREATE TABLE IF NOT EXISTS `Ksiegarnia`.`Ksiazka` (
  `idKsiazka` INT NOT NULL AUTO_INCREMENT,
  `tytul` VARCHAR(45) NOT NULL,
  `autor` VARCHAR(45) NOT NULL,
  `opis` TEXT NULL,
  `ilosc` TINYINT UNSIGNED NOT NULL,
  `cena` TINYINT UNSIGNED NOT NULL,
  `ISBN` VARCHAR(20) NOT NULL,
  `Wydanie_idWydanie` INT NOT NULL,
  `Wydanie_Nazwa_wydawnictwa_słownik_idNazwa_wyd` INT NOT NULL,
  PRIMARY KEY (`idKsiazka`, `Wydanie_idWydanie`,
`Wydanie_Nazwa_wydawnictwa_słownik_idNazwa_wyd`),
  CONSTRAINT `fk_Ksiazka_Wydanie1`
    FOREIGN KEY (`Wydanie_idWydanie`)
    REFERENCES `Ksiegarnia`.`Wydanie` (`idWydanie`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Ksiazka_Wydanie2`
    FOREIGN KEY (`Wydanie_Nazwa_wydawnictwa_słownik_idNazwa_wyd`)
    REFERENCES `Ksiegarnia`.`Wydanie` (`Nazwa_wydawnictwa_słownik_idNazwa_wyd`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

W powyższym kodzie większość zdeklarowanych kolumn zawiera specyfikator NOT NULL, który jest podstawowym ograniczeniem zapewniającym właściwą formę przechowywanych danych. Kolumny z tym specyfikatorem nie mogą być wypełnione wartością NULL, ale można jest pominąć przy podawaniu wartości poleceniem INSERT. Poza deklaracją kolumn następuje także określenie kluczy głównych oraz obcych. Klucz własny (idKsiazka) jest opatrzony specyfikatorem AUTO_INCREMENT, co powoduje że klucz główny nie musi być podawany przy wywoływaniu INSERT. Przy definiowaniu kluczy obcych należy podać nazwę tabel do których się odnoszą.

4.2. Implementacja mechanizmów przetwarzania danych

W trakcie implementacji bazy danych proponowane wcześniej wykorzystanie triggerów do inkrementowania kolumn id tabel okazało się zbędne (taki sam efekt można uzyskać poprzez dodanie specyfikatora AUTO_INCREMENT przy odpowiednich kolumnach). Wykorzystano triggery do automatycznego ustawiania statusu zamówień na 'nieaktywne' przed utworzeniem tabel Zamowienie oraz Zamowienie_has_Ksiazka.

W przypadku innych mechanizmów implementacja jest zgodna z Tabelą 2.

Kod realizujący dane funkcjonalności ze skryptu create_database.sql:

```
/*------------------------------TRIGGERY------------------------------*/
DELIMITER $$
CREATE TRIGGER ustaw_status_zamowienie
  BEFORE INSERT ON Zamowienie
    FOR EACH ROW
      BEGIN
        SET NEW.Status_zamówienia_słownik_idStatus_zamówienia_słownik = 1;
    END$$


CREATE TRIGGER ustaw_status_zamowienie_has_ksiazka
  BEFORE INSERT ON Zamowienie_has_Ksiazka
    FOR EACH ROW
      BEGIN
        SET NEW.Zamowienia_Status_zamówienia_słownik_idStatus_zamówienia_słownik = 1;
    END$$

DELIMITER ;

/*------------------------------INDEKSY------------------------------*/
CREATE UNIQUE INDEX Ksiazka_index
ON Ksiazka (tytul, autor);


/*------------------------------WIDOKI------------------------------*/
CREATE OR REPLACE VIEW widokKsiazka AS SELECT tytul, autor, cena FROM Ksiazka;
CREATE OR REPLACE VIEW widokKlienci AS SELECT login, imie, nazwisko, email FROM Klienci;
CREATE OR REPLACE VIEW widokZamowienie AS SELECT A.wartość,
A.Status_zamówienia_słownik_idStatus_zamówienia_słownik, A.data, B.login, B.imie,
B.nazwisko, B.email FROM Zamowienie A, Klienci B;
```

4.3. Implementacja uprawnień i innych zabezpieczeń

Uprawnienia dla użytkowników zostały dodane zgodnie z tabelą 1 w dziale 3.1.5. Kod przydzielający uprawnenia został zawarty w skrypcie create_database.sql:

```
GRANT ALL ON `Ksiegarnia`.* TO 'admin';

GRANT INSERT, SELECT, UPDATE, CREATE, DELETE ON `Ksiegarnia`.`Adres` TO 'klient_zalog';
GRANT INSERT, SELECT, UPDATE, CREATE, DELETE ON `Ksiegarnia`.`Klient` TO 'klient_zalog';
GRANT INSERT, SELECT, UPDATE ON `Ksiegarnia`.`Recenzje` TO 'klient_zalog';
GRANT SELECT ON `Ksiegarnia`.`Ksiazka` TO 'klient_zalog';
GRANT SELECT ON `Ksiegarnia`.`Kategoria_słownik` TO 'klient_zalog';
GRANT SELECT ON `Ksiegarnia`.`Wydanie` TO 'klient_zalog';
GRANT SELECT ON `Ksiegarnia`.`Nazwa_wydawnictwa_słownik` TO 'klient_zalog';
GRANT INSERT, SELECT, UPDATE ON `Ksiegarnia`.`Zamowienie` TO 'klient_zalog';
GRANT SELECT ON `Ksiegarnia`.`Status_zamówienia_słownik` TO 'klient_zalog';

GRANT SELECT ON `Ksiegarnia`.`Recenzje` TO 'klient_niezalog';
GRANT SELECT ON `Ksiegarnia`.`Ksiazka` TO 'klient_niezalog';
GRANT SELECT ON `Ksiegarnia`.`Kategoria_słownik` TO 'klient_zalog';
GRANT SELECT ON `Ksiegarnia`.`Wydanie` TO 'klient_niezalog';
GRANT SELECT ON `Ksiegarnia`.`Nazwa_wydawnictwa_słownik` TO 'klient_niezalog';
```

4.4. Testowanie bazy danych na przykładowych danych

Do przetestowania wszystkich funkcjonalności został stworzony skrypt dodający przykładowe dane oraz wypisujący zawartość kolumn po dodaniu. Przed wykonaniem jakichkolwiek czynności skrypt usuwa poprzednią zawartość bazy (polecenie TRUNCATE), w celu uniknięcia dodawania tych samych danych przy ponownym uruchomieniu skryptu. Jest bardzo prosta metoda na przetestowanie operacji wykonywanych na bazie, podobna do testów jednostkowych w programowaniu. Stworzenie takiego skryptu ułatwia wprowadzanie modyfikacji w bazie, ponieważ znacząco przyśpiesza znajdowanie błędów.

Fragment kod ze skryptu test.sql:

```
INSERT INTO Nazwa_wydawnictwa_słownik (Nazwa_wydawnictwa) VALUES
    ('Amber'),('PWN'),('Prószynski i S-ka'),
    ('Znak'),('Biały Kruj'),('Rebis');

SELECT * FROM Nazwa_wydawnictwa_słownik;


INSERT INTO  Wydanie (rok_wyd, miejsce, oprawa, Nazwa_wydawnictwa_słownik_idNazwa_wyd)
VALUES
  ('2016-05-01','Warszawa','t',1),
  ('2015-12-21','Białystok','m',4);
SELECT * FROM Wydanie;

SELECT * from Nazwa_wydawnictwa_słownik WHERE idNazwa_wyd IN (SELECT
Nazwa_wydawnictwa_słownik_idNazwa_wyd FROM Wydanie);
```

Ostatnie polecenie SELECT wyszukuje nazwy wydawnictw ze słownika, obecne w tabeli Wydanie.

Efekt wykonania kodu:

```
Query OK, 6 rows affected (0.03 sec)
Records: 6  Duplicates: 0  Warnings: 0

+-------------+--------------------+
| idNazwa_wyd | Nazwa_wydawnictwa  |
+-------------+--------------------+
|           1 | Amber              |
|           2 | PWN                |
|           3 | Prószynski i S-ka  |
|           4 | Znak               |
|           5 | Biały Kruj         |
|           6 | Rebis              |
+-------------+--------------------+
6 rows in set (0.00 sec)

Query OK, 2 rows affected (0.04 sec)
Records: 2  Duplicates: 0  Warnings: 0

+-----------+------------+------------+--------+-----------------------------------------+
| idWydanie | rok_wyd    | miejsce    | oprawa | Nazwa_wydawnictwa_słownik_idNazwa_wyd  |
+-----------+------------+------------+--------+-----------------------------------------+
|         1 | 2016-05-01 | Warszawa   | t      |                                       1 |
|         2 | 2015-12-21 | Białystok  | m      |                                       4 |
+-----------+------------+------------+--------+-----------------------------------------+
2 rows in set (0.00 sec)

+-------------+-------------------+
| idNazwa_wyd | Nazwa_wydawnictwa |
+-------------+-------------------+
|           1 | Amber             |
|           4 | Znak              |
+-------------+-------------------+
```

5. Implementacja i testy aplikacji

MySQLWorkbench został wykorzystany do stworzenia fizycznego modelu bazy danych, a następie za jego pomocą stworzono skrypt SQL generujący tabele.

Do stworzenia aplikacji wykorzystano Django, czyli wolny i otwarty framework przeznaczony do tworzenia aplikacji internetowych, napisany w Pythonie. Testowanie funkcjonalności odbywało się przede wszytskim bezpośrednio po ich implementacji. w każdym z widoków użytkownika użyto zmiennej error_msg. Jej użycie pozwolilo na komunikowanie użytkownika o błędnym wypelnieniu formularza.

5.1. Instalacja i konfigurowanie systemu

Pierwszym krokiem jest instalacja Pythona:

```
$ sudo apt-get install python3.4
```

Następnie należy stworzyć nowy katalog bazydanych:

```
$ mkdir bazydanych
$ cd bazydanych
```

Stworzono nowe środowisko wirtualne o nazwie `env`. Polecenie ma następujący format:

```
$ virtualenv env
```

Uruchomiono wirtualne środowisko:

```
$ source env/bin/activate
```

Zainstalowano Djngo, a także mysql-python:

```
$ pip install django
$ pip install mysql-python
```

Skorzystano z dostarczonych przez Django skryptów, które tworzą szkielet projektu Django. Ten szkielet to zbiór katalogów i plików.

```
$ django-admin startproject ksiegarnia.
```

Wprowadzono parę zmian w pliku `mysite/settings.py.` Zmieniono domyślną bazę danych do przechowywania informacji z aplikacji na naszą bazę danych sql

```
DATABASES = {

                    'default': {
                        'ENGINE': 'django.db.backends.mysql',
                        'NAME': 'Ksiegarnia',
                        'USER': 'root',
                        'PASSWORD': 'tymbark12',
                        'HOST': 'localhost',   # Or an IP Address that your DB is hosted on
                        'PORT': '3306',
                        #'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
                    }
                }
```

Do testowania aplikacji tylko na lokalnym komputerze, przechodząc do katalogu wewnętrznego księgarnia używamy:

```
$ python manage.py runserver
```

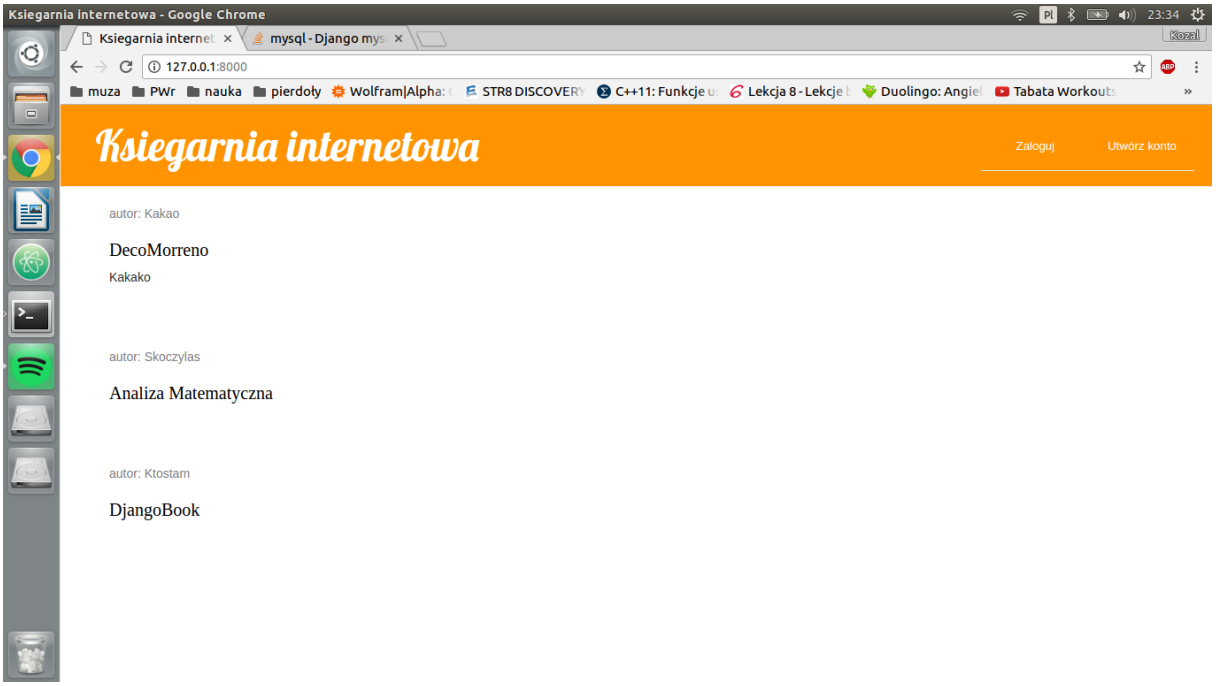
Po uruchomieniu serwera należy w przeglądarce połączyć się localhostem na porcie 8000:

```
http://127.0.0.1:8000/
```

5.2. Instrukcja użytkowania aplikacji

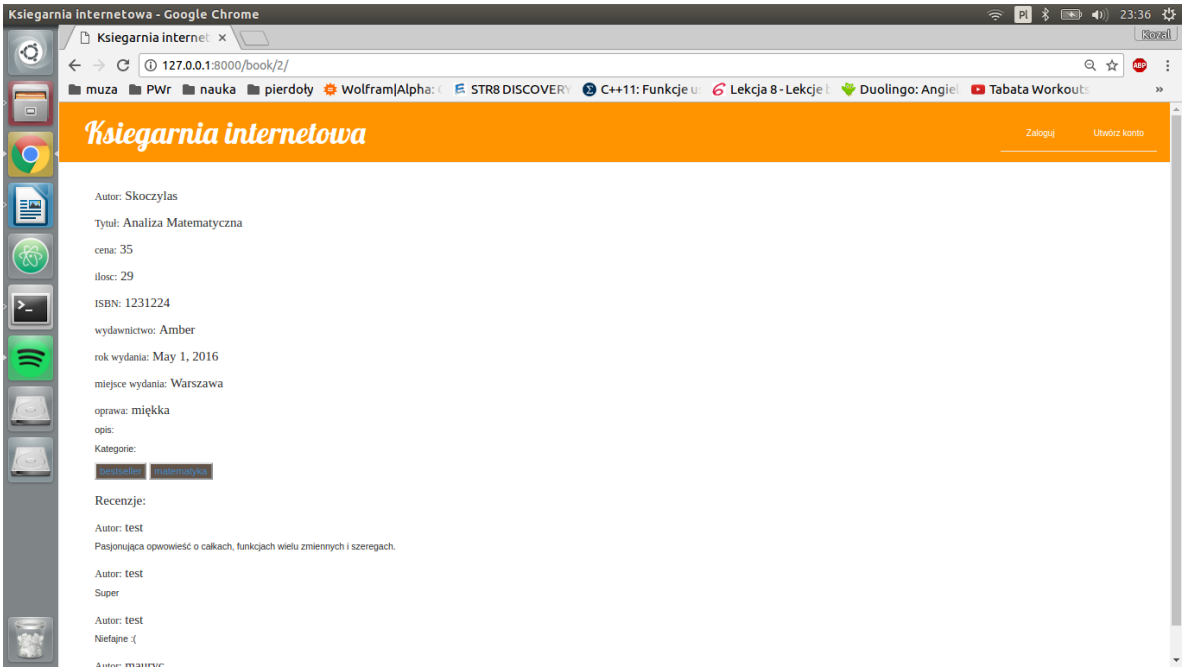
5.2.1. Instrukcja dla użytkownika

Na stronie startowej można przeglądać książki dostępne w ofercie księgarni.



Rysunek 10 Strona startowa

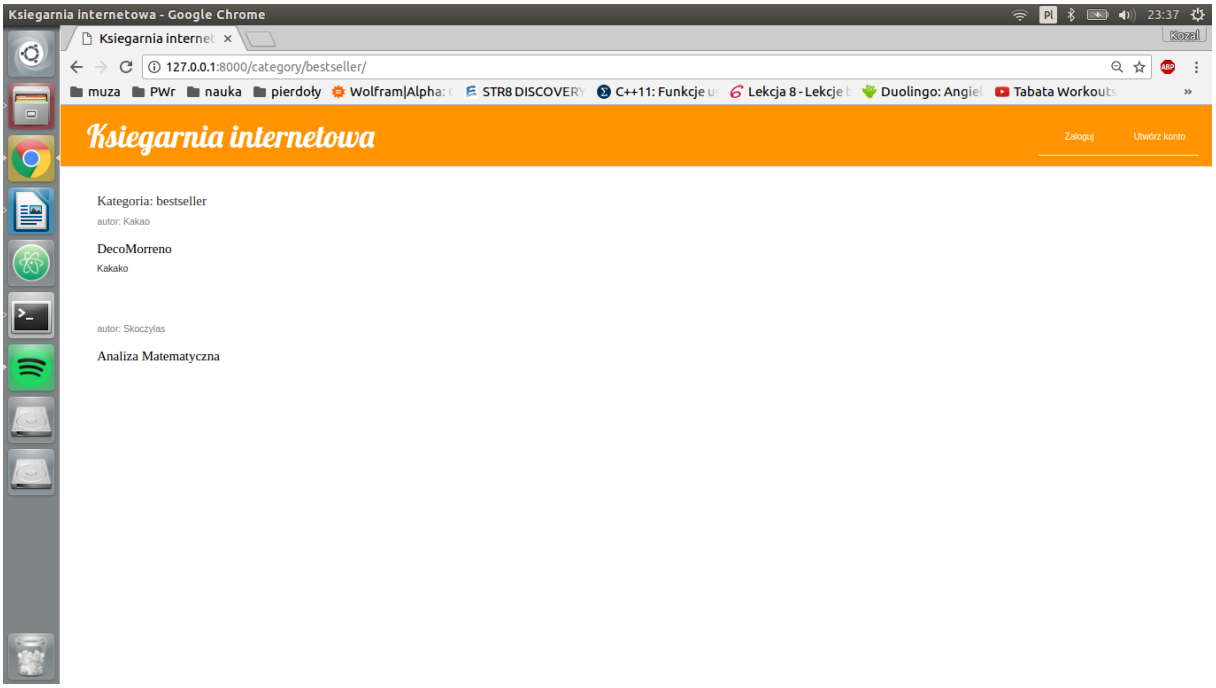
Dostępne są jedynie informacje o autorze, tytule oraz krótki opis książki. Po naciśnięciu na wybraną pozycję otrzymujemy bardziej szczegółowe informacje.



Rysunek 11 Szczegółowe dane na temat książki

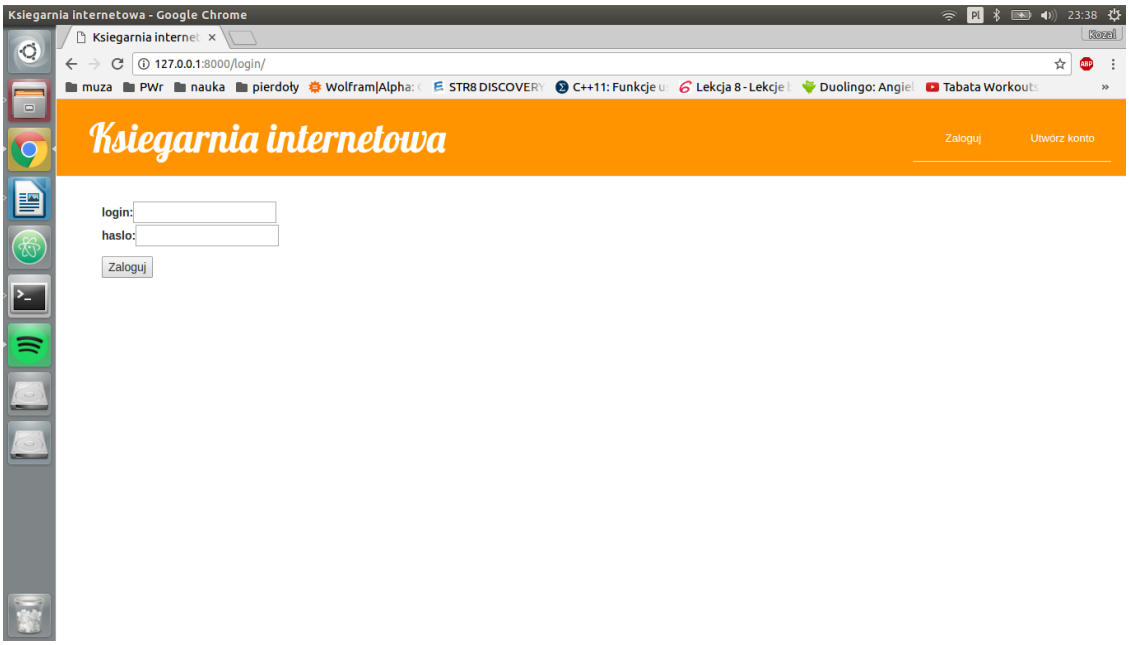
Dostępne są szczegóły takie jak cena, ilość egzemplarzy w magazynie oraz informacje na temat wydania. Dla danej książki są również udostępnione recenzje, pisane przez użytkowników.

Na powyższym screenie widać także kategorie jakie zostały przypisane książce. Klikając na daną kategorię można wyszukać wszystkie książki z danej kategorii.



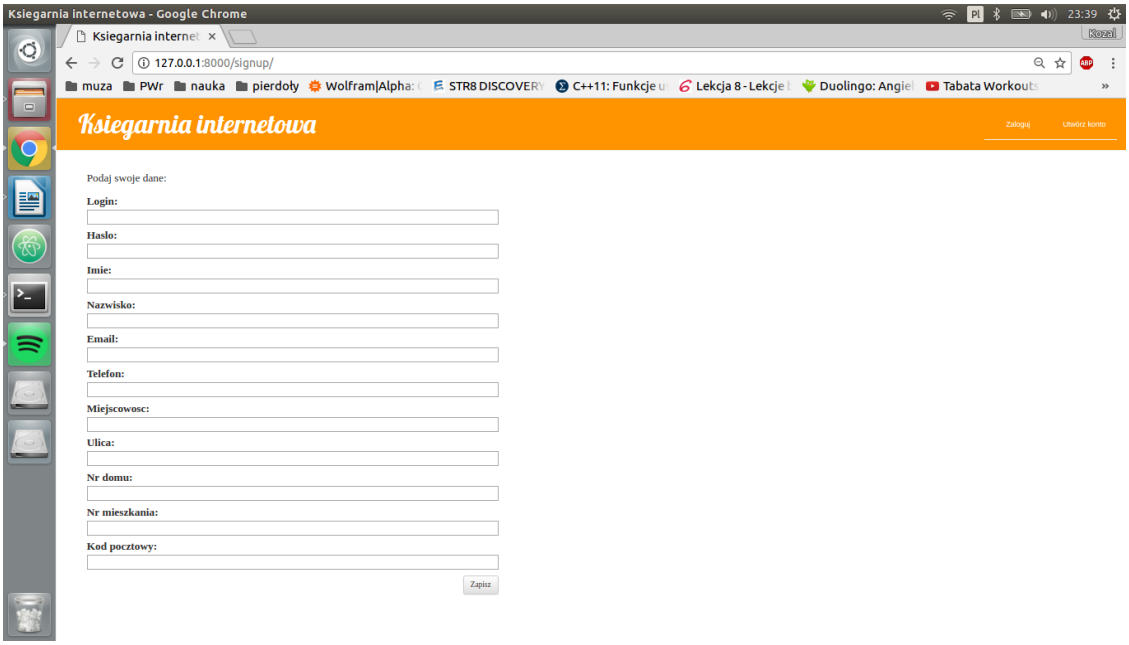
Rysunek 12 Kategoria bestseller

Aby zyskać dostęp do funkcjonalności związanych z zamawianiem książek i pisaniem recenzji należy się zalogować. Przycisk „zaloguj” i „utwórz konto” zawsze znajdują się w prawym górnym rogu ekranu.



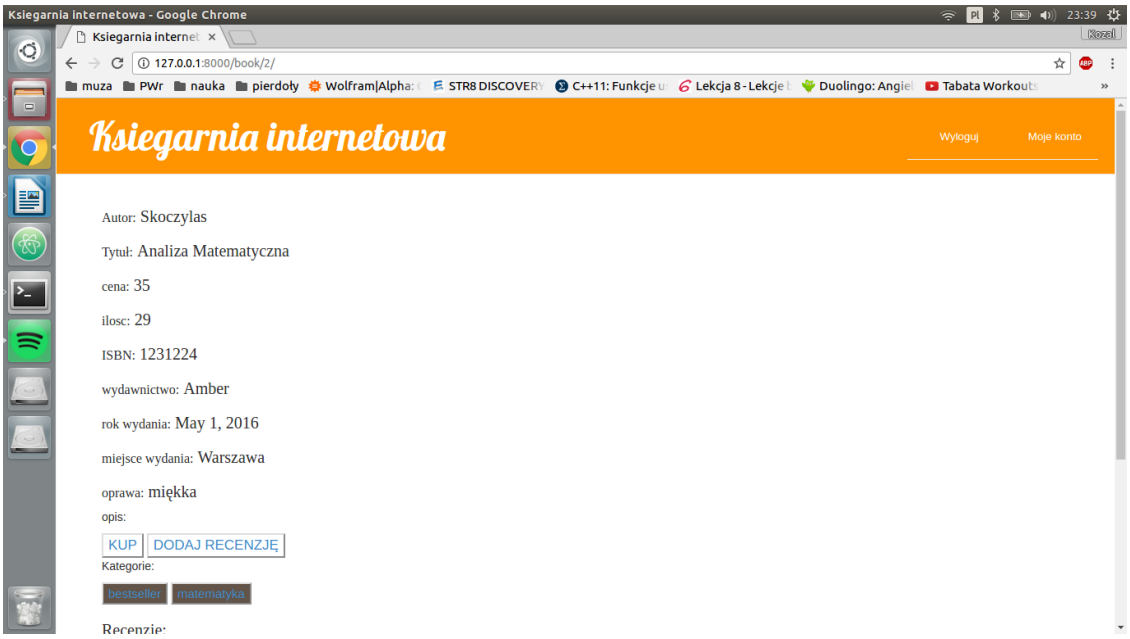
Rysunek 13 Ekran logowania

Gdy użytkownik nie posiada jeszcze konta musi je utworzyć podając dane:



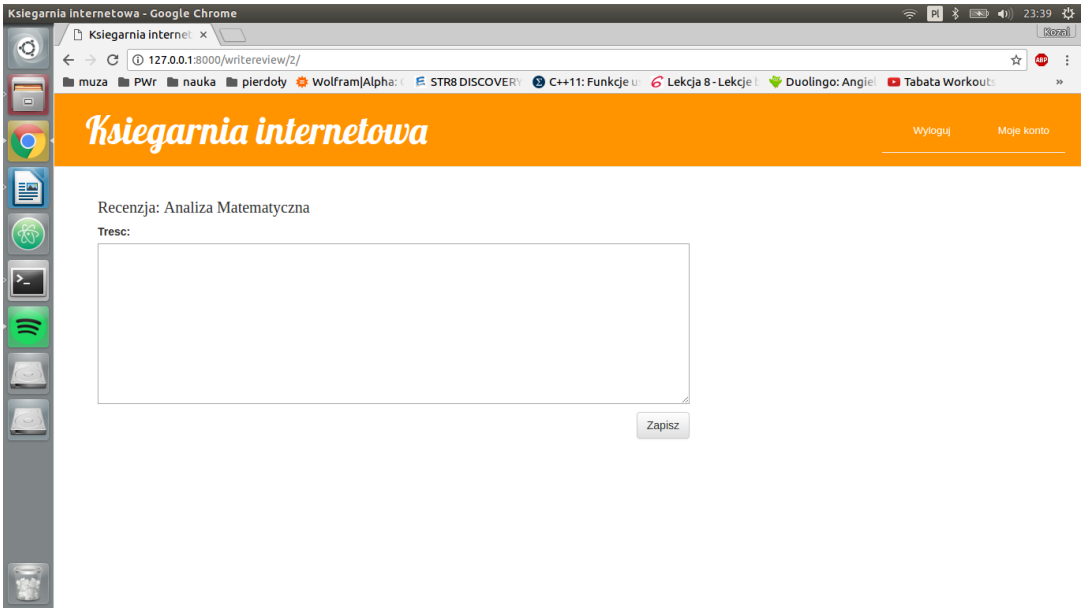
Rysunek 14 Ekran tworzenia konta

Po zalogowaniu na stronie ze szczegółami danej pozycji książkowej zyskujemy dostęp do przycisków kup i dodaj recenzję.



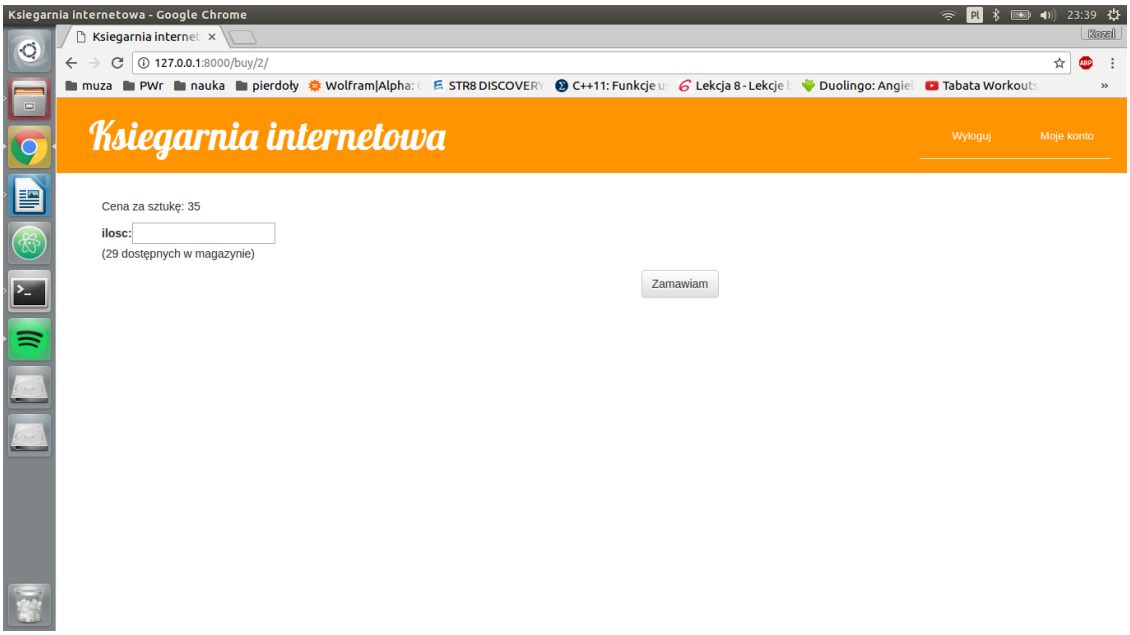
Rysunek 15 Przyciski KUP i DODAJ RECENZJĘ

Dodając recenzję podajemy jej treść, następnie zapisujemy. Po zapisaniu recenzja pojawi się na stronie z loginem zalogowanego użytkownika jako autorem.



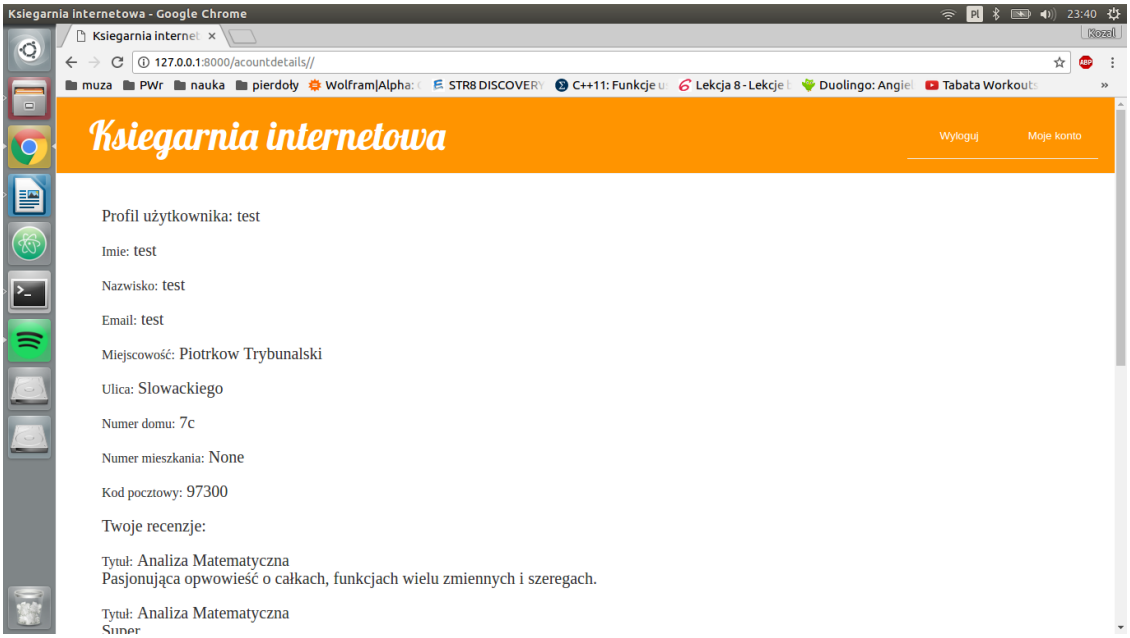
Rysunek 16 Dodawanie recenzji do książki

Zamówienie polega na sprecyzowaniu ilości książek, które chcemy zamówić. Nie może ona przekroczyć ilości książek w magazynie.

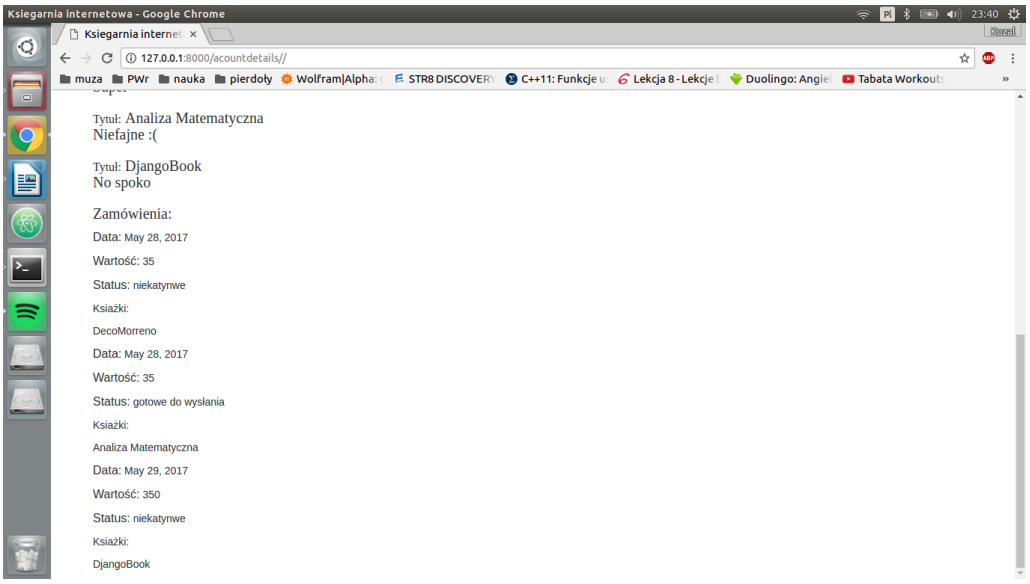


Rysunek 17 Składanie zamówienia

Po wciśnięciu przycisku Zamawiam możemy obserwować status naszego zamówienia w zakładce „Moje konto”. Po zalogowaniu pojawia się ona w prawym górnym rogu, obok przycisku „Zaloguj”. Są tutaj przedstawione szczegóły na temat konta użytkownika: jego dane osobowe, napisane recenzje oraz historia zamówień:



Rysunek 18 Informacje na temat profilu użytkownika

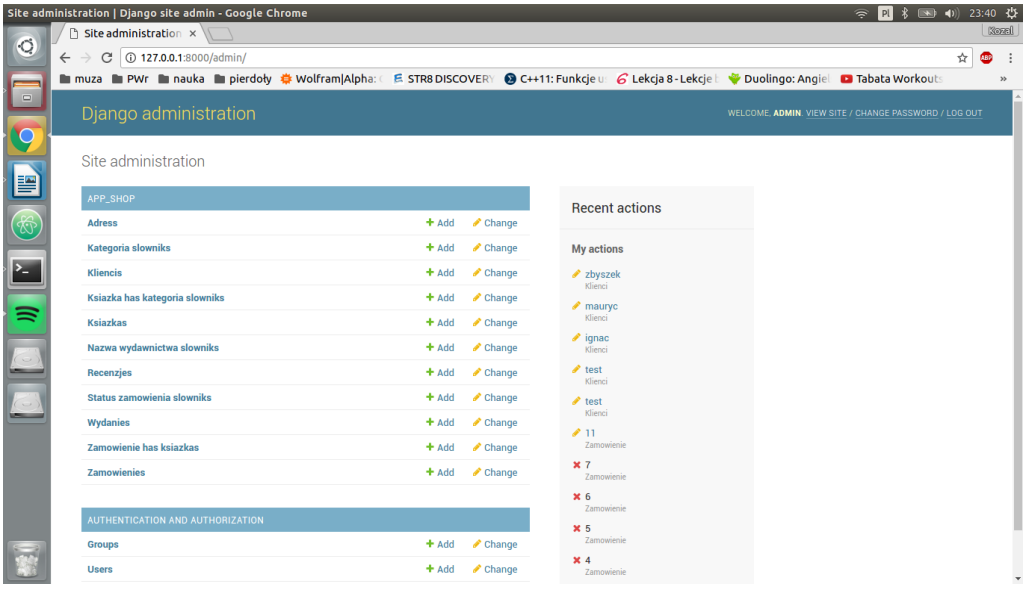


Rysunek 19 Informacje na temat profilu użytkownika

W tym ekranie można obserwować status zamówień, datę złożenia, co składa się na dane zamówienie oraz inne szczegóły.

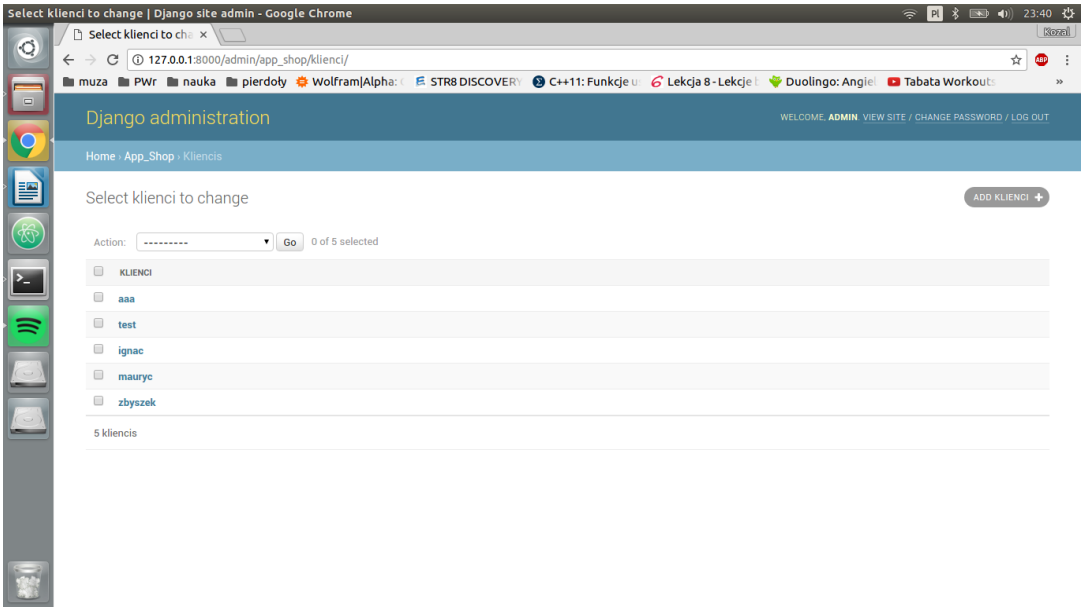
5.2.2. Instrukcja dla administratora

Dodawanie nowych pozycji książkowych, usuwanie kont użytkowników, zmiana statusów zamówień oraz wiele innych czynności może być wykonane przez panel administatora django. Dostęp do panelu zyskuje się poprzez dodanie do adresu strony /admin/ Mamy tutaj dostęp do wszystkich tabel z bazy danych.



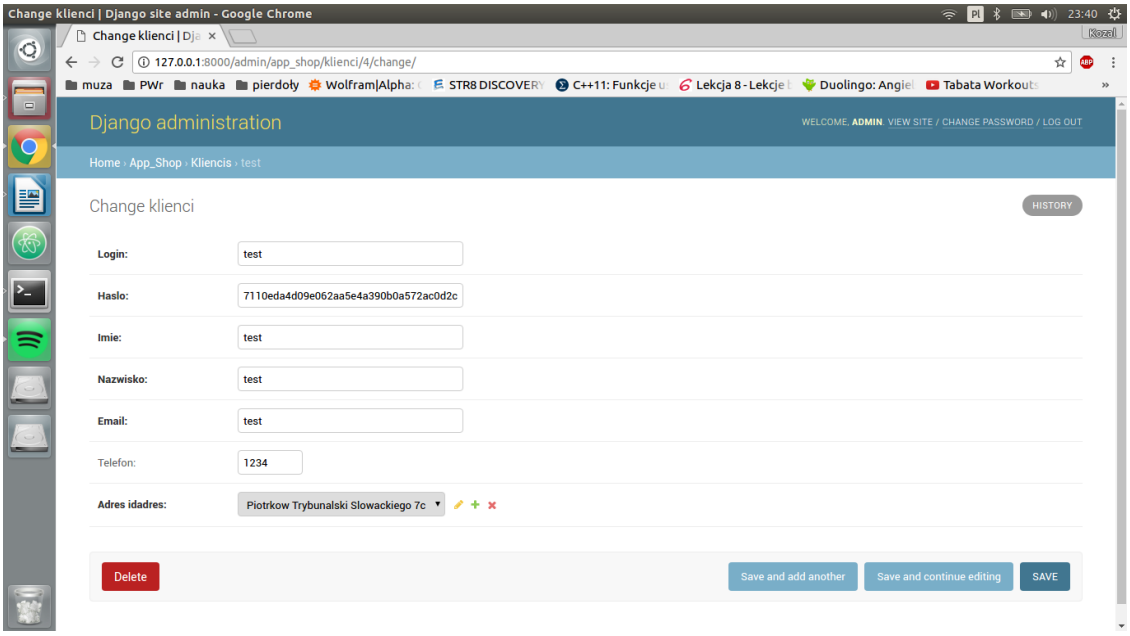
Rysunek 20 Panel administratora

Po naciśnięciu na wybraną tabelę widzimy wszystkie kolumny, które są aktualnie przechowywane w bazie:



Rysunek 21 Kolumny tabeli książka

W prawym górnym rogu znajduje się przycisk add new do dodawania nowych danych, po naciśnięciu na wybraną kolumnę zyskujemy informację na przechowywanych w niej danych:



Rysunek 22 Szczegółowe dane klienta

Aby zapisać zmiany należy wcisnąć jeden z przycisków z napisem save. Do usuwania danej kolumny służy przycisk delete w lewym dolnym rogu ekranu.

Jest to bardzo wygodny i intuicyjny sposób zarządzania danymi. Pozwala na szybkie i bezproblemowe dodawanie, usuwanie i modyfikację danych. Jest to jednocześnie jego wadą, ponieważ bardzo łatwo jest usunąć potrzebne dane. Posługiwanie się tym systemem wymaga od administratora podstawowej znajomości zasad działania systemu oraz uwagi. Nieuważny administrator może spowodować wiele szkód.

5.3. Testowanie opracowanych funkcji systemu

Testowanie funkcjonalności odbywało się przed wszystkim bezpośrednio po ich implementacji. Było to ułatwione przez uporządkowany schemat pracy ze wzorcem projektowym View-Model-Template (najpierw należy zdefiniować URL oraz widok, do którego się odnosi, następnie zaimplementować widok powodujący wyświetlenie przez przeglądarkę odpowiedniej strony – pliku z rozszerzeniem .html). Każdorazowe dodanie funkcjonalności można bezpośrednio obserwować na ekranie przeglądarki, co znacząco ułatwia sprawdzanie poprawności kodu. Oprócz tego przy sprawdzaniu poprawności działania systemu wykorzystywano panel administratora, do sprawdzania jakie dane są dodawane bądź modyfikowane przez aplikację w bazie danych. Inną metodą sprawdzania poprawności było przeglądanie zawartości bazy z wykorzystaniem programu konsolowego mysql.
Testowano również funkcjonalności aplikacji przez symulowanie zachowania użytkownika. Podstawowym testem aplikacji było błędne wprowadzenie hasła zarejestrowanego użytkownika, w efekcie pojawił się komunikat o niepoprawnym haśle. w każdym z widoków użytkownika użyto zmiennej error_msg. Jej użycie pozwoliło na komunikowanie użytkownika o błędnym wypełnieniu formularza tak jak w przypadku błędnego hasła. Niektóre z przykładów komunikatów przedstawiono poniżej:

```
        error_msg = "Podany login jest już zajęty"

        error_msg = "Podany email jest juz zajety"

        error_msg = "Podaj pola obwiązkowe!"

        error_msg = "Zle haslo!"

        error_msg = "Nieprawidlowy login!"

        error_msg = "Podaj login i haslo!"

        error_msg = "W magazynie nie ma tak wielu ksiazek!"

        error_msg = "Podaj ile egzemplaży chcesz kupić!"

        error_msg = "Podaj treść recenzji!"
```

Innym przykładem takiego testu była próba wprowadzenia większej liczby książek w pole ilość przy składaniu zamówienia, niż ta dostępna na magazynie. Wówczas również pojawił się komunikat o tym ile książek pod tym tytułem jest na magazynie. Wprowadzono ujemną liczbę w to pole, w tej sytuacji liczba ujemna zamówienia zamieniona jest na 0 zaś liczba książek w magazynie się nie zmienia . Przy próbie wpisywania tekstu w pole „ilość" nie pojawiają się znaki.

5.4. Omówienie wybranych rozwiązań programistycznych

5.4.1. Implementacja interfejsu dostępu do bazy danych

W Django dostęp do bazy danych odbywa się na bardzo wysokim poziomie abstrakcji. Informacje na temat encji są przechowywane w specjalnych klasach dziedziczących po klasie Model. Zawierają one specyfikację poszczególnych atrybutów encji wraz z wyszczególnieniem ich typów. Poniżej przedstawiono przykładowy kod źródłowy modelu:

```
class Klienci(models.Model):
idklient = models.AutoField(db_column='idKlient', primary_key=True)  # Field name made lowercase.
    login = models.CharField(max_length=15)
    haslo = models.CharField(max_length=45)
    imie = models.CharField(max_length=30)
    nazwisko = models.CharField(max_length=30)
    email = models.CharField(max_length=30)
    telefon = models.IntegerField(blank=True, null=True)
    adres_idadres = models.ForeignKey(Adres, models.DO_NOTHING, db_column='Adres_idAdres')  # Field name made
lowercase.
    class Meta:
        managed = False
        db_table = 'Klienci'
        unique_together = (('idklient', 'adres_idadres'),)
```

Modele mogą zostać stworzone na dwa sposoby. Pierwszym sposobem jest samodzielnie zdefiniowanie modeli i dokonanie migracji na bazę danych. Druga metoda została opracowana w przypadku pracy z bazami danych typu legacy (istniejącymi wcześniej). Jest to polecenie inspectdb (po uprzednim uzupełnieniu informacji na temat wykorzystywanego silnika, nazwy, etc.). w projekcie wykorzystano drugą metodę, ponieważ przed rozpoczęciem tego etapu baza danych była już gotowa. Modele są później wykorzystywane w widokach do wyszukiwania, modyfikowania i usuwania informacji. Według struktury aplikacji w django widoki umieszczone są w pliku views.py. Poniżej przedstawiony został jeden z mniej skomplikowanych widoków z tego pliku:

```
def books_list(request, loged_in=False):
    books = Ksiazka.objects.all
    if read_id() > 0:
        loged_in = True
    else:
        loged_in = False
    return render(request, 'app_shop/books_list.html', {'books': books, 'loged_in':
loged_in})
```

Powyszyższy widok został przekazany w pliku urls.py w następujący sposób:

```
urlpatterns = [
                    url(r'^$', views.books_list, name='books_list'), …
```

Zapytania są realizowane z wykorzystaniem tzw QuerySet-ów, które są zbiorami obiektów typu model spełniającymi warunki zapytania. QuerySety mogą przekazywane do plików .html, co bardzo ułatwia prezencję danych użytkownikowi.

5.4.2. Implementacja wybranych funkcjonalności systemu

Wszystkie funkcjonalności z rysunku 1 z działu 3.1.1 zostały zaimplementowane i przetestowane.

Większość funkcjonalności została zrealizowana w ten sam sposób, co jest związane z restrykcyjnym przestrzeganiem wyżej wspomnianego wzorca projektowego Model-View-Template. Użytkownik wypełnia formularz na stronie (Django forms), który następnie jest przekazywany do widoku, który wcześniej wywołał renderowanie aktualnego szablonu. Mechanizm ten wykorzystuje metodę POST zarówno po stronie wzorca jak i widoku. w widoku następuje walidacja danych oraz przekierowanie do innych adresów url/widoków.

Rozwiązania na które warto zwrócić uwagę to logowanie do systemu. Django posiada własny system użytkowników, który nie mógł być wykorzystany, ponieważ dane klientów są przechowywane w osobnej tabeli w bazie danych. Problem ten rozwiązano poprzez tworzenie pliku tymczasowego po stronie serwera zapisującego id użytkownika.

Innym problemem było dodawanie tabeli reprezentujących relację wielu do wielu. Django posiada inny sposób modelowania tej relacji niż tabele typu Object1_has_Object2 zawierające klucze obu obiektów. Jednocześnie każda tabela w modelu Django musi posiadać klucz główny. Podczas tworzenia modelu pierwszy klucz obcy w tabelach „has" został oznaczony jako główny. Tak jak poniżej:

```
class KsiazkaHasKategoriaSlownik(models.Model):
    ksiazka_idksiazka = models.ForeignKey(Ksiazka, models.DO_NOTHING,
db_column='Ksiazka_idKsiazka', primary_key=True)
```

Skutkowało to brakiem możliwości dodawania wielu tabel z takim samym kluczem obcym ponieważ klucze własne muszą być unikatowe (aby być bardziej obrazowym np. do danej książki kategoria mogła być przypisana tylko raz). Aby zachować oryginalną postać bazy do omawianego typu tabel dodano pole przechowujące klucz główny.

5.4.3. Implementacja mechanizmów bezpieczeństwa

Django zabezpiecza aplikację przed wieloma rodzajami ataków jak SQL Injection czy Clickjacking. w przypadku np. ataków typu XSS django nie zapewnia całkowitej ochrony. Podstawowym źródłem ochrony danych osobowych jest ochrona hasłem, które jest przechowywane zaszyfrowane w bazie danych. Dostęp do określonych operacji dla użytkowników został zrealizowany poprzez sprawdzanie czy wariant strony ma być wyświetlany dla użytkownika zalogowanego czy niezalogowanego. Kod html wygenerowany dla użytkownika niezalogowanego nie umożliwia przejście do sprzedaży lub dodawania recenzji.

6. Podsumowanie i wnioski

Wykonanie niniejszego projektu pozwoliło na zapoznanie się z nowymi technologiami. Było to bardzo rozwijające, ponieważ nie mieliśmy z nimi styczności wcześniej, a wykorzystanie w praktyce nowych narzędzi to jeden z szybszych i przyjemniejszych sposób nauki.

Przed projektowaniem całego systemu warto mieć świadomość możliwości i ograniczeń jakie niosą ze sobą technologię, ponieważ będzie to miało istotny wpływ na przebieg całego projektu. Tyczy się to zarówno baz danych jak i aplikacji. Na etapie projektowania bardzo pomagają narzędzia umożliwiające porządkowanie i planowanie. w tym projekcie były to MySQLWorkbench zastosowany do stworzenia modelu bazy danych oraz Balasmiq do stworzenia makiety graficznej aplikacji.

Bazy danych to bardzo istotna część dzisiejszych programów i nie sposób pomyśleć o poważnych aplikacjach, które by z nich nie korzystały. z tego powodu każdy szanujący się programista powinien znać podstawy ich funkcjonowania. w większości języków programowania istnieją wysokopoziomowe frameworki, które znacznie ułatwiają pracę programistom, jednakże warto mieć świadomość tego jak realizowane są operacje na bazie danych oraz jaki efekt będzie miał nasz kod.

Zaimplementowana aplikacja nie jest bardzo skomplikowana, ale realizuje wszystkie założenia projektowe. Django jest frameworkiem do którego łatwo dodawać nowe funkcjonalności, co połączone z dobrze napisanym i uporządkowanym kodem może powodować że aplikacja będzie łatwo rozszerzalna i modyfikowalna. Można wręcz odnieść wrażenie, że Django zostało stworzone aby sprostać dynamicznie zmieniającym się wymaganiom biznesowym oraz aby współgrać ze zwinnymi metodykami wytwarzania oprogramowania.

Literatura

Benyon-Davies, P.: *Systemy baz danych*. WNT, Warszawa, 1998. ISBN 83-204-2257-4.

http://www.tutorialspoint.com/mysql/

https://docs.djangoproject.com/en/1.11/

https://tutorial.djangogirls.org/

http://miroslawzelent.pl/kurs-mysql/

Spis rysunków

Spis tabel