# Wstęp

Asdadasd

# Teoria handlu internetowego

W rozdziale przedstawione zostały podstawowe zagadnienia związane z handlem internetowym. Zagadnienia te podzielone zostały na dwa podrozdziały, pierwszy opisujący szersze zagadnienie e-Commerce, oraz drugi skupiający się na sklepach internetowych.

## E-commerce

Jak wspomniano we wstępie do tego rozdziału e-commerce to pojęcie szersze od sklepu internetowego. E-Commerce bowiem to różnego rodzaju transakcje biznesowe, które dokonywane są z wykorzystaniem internetu[1]. Oczywiście zakupy w sklepach internetowych można przedstawić jako jeden z rodzajów takich transakcji.

E-Commerce można zdefiniować jako działania wykowywane w celu dokonania transakcji za pomocą środków elektronicznych[1]. Płatność i dostawa towarów i usług zakupionych w ten sposób odbywa się poza fizycznym miejscem występowania sklepu. Oprócz wspomnianych już sklepów internetowych do obszaru e-Commerce możemy zaliczyć również między innymi aukcje internetowe, zakłady bukmacherskie, czy bankowość elektroniczną.

E-commerce zaczął dzielić się również na mniejsze, bardziej wyspecjalizowane obszary wśród których wymienić możemy m-Commerce który skupia się na umożliwieniu użytkownikom korzystania z oferowanych usług za pomocą urządzeń mobilnych. Innym obszarem jest Q-commerce, który charakteryzuje się bardzo szybkim czasem dostawy zazwyczaj nieprzekraczającym jednej godziny, tak jak w przypadku systemów umożliwiającym dostawę jedzenia[1]. Live commerce z kolei to transmisje internetowe podczas których można dokonywać zakupów prezentowanych towarów, bądź usług.

Ze względu na strony transakcji E-commerce możemy podzielić na trzy typy:

— B2C, businnes to consumer — stroną sprzedającą jest firma, a stroną kupującą indywidualny klient

— B2B, businnes to businnes — stroną sprzedającą jest firma, a stroną kupującą inna firma

— C2C, consumer to consumer — stroną sprzedającą i kupującą są indywidualni konsumenci, przykładem takiego rozwiązania są serwisy aukcji internetowych takie jak Allegro, czy OLX

E-commerce w ostatnich latach rozwija się bardzo szybko i przewiduje się że trend ten będzie utrzymywał się również w najbliższych latach. Nie sposób nie zauważyć że duży wpływ na to miała pandemia która uniemożliwiła na pewien czas zakupy w większości sklepów stacjonarnych. Sytuacja ta zmusiła społeczeństwo do korzystania z rozwiązań dostępnych w internecie, a spora część zadowolonych klientów korzysta z tego rozwiązania w dalszym ciągu. Kolejnym czynnikiem wpływającym na rozwój e-commerce jest digitalizacja społeczeństwa, które coraz lepiej zna się na obsłudze urządzeń elektronicznych podłączonych do internetu. Metody płatności internetowych również stają się coraz prostsze, oraz bezpieczniejsze. Dzięki temu, szybkie i wygodne zakupy w internecie nie stanowią już problemu dla większości obywateli krajów rozwiniętych, w tym Polski.

Warto wspomnieć o korzyściach jakie płyną z rozwiązań e-Commerce zarówno dla kupującego jak i dla sprzedającego.

Sprzedający może prowadzić sprzedaż przez całą dobę, przez co trafia również do klientów którzy nie mają czasu na zakupy w godzinach funkcjonowania sklepów stacjonarnych. Firmy działające w e-Commerce mają również znacznie mniejsze koszty w porównaniu z tymi, które prowadzą swoje sklepy stacjonarnie. Nie muszą one płacić za wynajem lokalu, a w większości przypadków na początku działalności nie potrzebują też zatrudnionych na stałe pracowników. Działalności e-Commerce nie mają też ograniczeń rynkowych związanych z miejscem występowania sklepu, nie ma problemu żeby sklep internetowy prowadzony przez przedsiębiorcę z Warszawy sprzedawał w tym samym czasie produkty w Zakopanem i w Gdańsku. Często słyszy się też o sklepach koncentrujących swoją sprzedaż na rynkach zagranicznych i również nie istnieją ograniczenia, które mogłyby im to uniemożliwiać. Mam również przeświadczenie że marketing w branży e-Commerce jest efektywniejszy niż w przypadku tradycyjnych działalności, a to dlatego że klient po zobaczeniu reklamy w internecie może po prostu klikną w link i już jest na stronie sklepu, gotowej do sprzedaży produktu. W przypadku sklepów stacjonarnych klient musi być na tyle zachęcony reklamą aby wybrać się do sklepu osobiście i dokonać zakupów.

Już we wcześniejszym zdaniu można odczuć że kupujący również odnosi korzyści wynikające z możliwości zakupów online. Są one dużo wygodniejsze i szybsze, pozwalają na zakup potrzebnych produktów lub usług bez ruszania się z domu. Kupujący zaoszczędza w ten sposób całą masę czasu i może go spożytkować w inny sposób. Dzięki zakupom online kliencie mają również dostęp do znacznie szerszej oferty, przez co mogą porównać oferty konkurencyjnych firm lub znaleźć niszowy produkt, trudny do znalezienie w sklepach stacjonarnych. Konkurencja w branży e-Commerce sprawa że firmy obniżają marże, tak aby ich oferta był jak najbardziej atrakcyjna, oczywiście korzystają na tym klienci którzy robią zakupy taniej. Warto również zwrócić uwagę na fakt że zakupy online są dużym udogodnieniem dla osób, które nie są w stanie kupować w sklepach stacjonarnych ze względu na przykład na niepełnosprawności ruchowe.

E-commerce w ostatnich latach stał się wiodącym obszarem sprzedaży usług i towarów na świecie na co ogromny wpływ ma digitalizacja społeczeństwa. Rynek e-Commerce w Polsce charakteryzuje się dużą liczbą małych sklepów internetowych, często prowadzonych przez jedną osobę. Chociaż większe sklepy mają oczywistą przewagę nad mniejszymi, te znajdują swoje miejsce w niszach, w których duże sklepy by się nie odnalazły, ponadto są konkurencyjne w indywidulanym podejściu do klienta i korzystnej cenie[1]. W Polsce popularnymi branżami e-Commerce są kosmetyki, odzież, produkty do domu i ogrodu, elektronika, książki, dodatki i akcesoria, zdrowy styl życia, akcesoria dla zwierząt. Natomiast małe, dobrze działające sklepy internetowe można znaleźć w praktycznie wszystkich, nawet tych bardzo niszowych obszarach naszego życia.

## Sklep internetowy

Sklep internetowy możemy prosto zdefiniować jako aplikację internetową umożliwiającą sprzedaż i zakupy online. Z informatycznego punktu widzenia należy wspomnieć o tym że aplikacja taka zazwyczaj składa się z Web serwisu działającego po backendowej stronie i aplikacji klienckiej działającej po stronie frontendu. Jednak dla przeciętnego użytkownika sklep taki jest po prostu stroną internetową na którą mogą wejść i dokonać zakupów.

W sposobie tworzenia sklepu, platformy e-Commerce możemy wyróżnić trzy kategorie:

—Platformy open-source — Są to platformy, które udostępniają swoje zasoby za darmo i pozwalają na samodzielne tworzenie sklepu internetowego korzystając z gotowych wtyczek, szablonów i dostosowując proponowane rozwiązania do swojego sklepu[1]. Istnieje również możliwość uzupełnienia swojego sklepu o dodatkowe funkcjonalności w ramach konta premium. Zazwyczaj osoba chcąca założyć sklep korzystając z platformy open-source nie uczy się budowania sklepu samodzielnie, a zleca taką usługę podmiotowi zewnętrznemu. Przykładami takich platform są WooCommerce, czy Magento.

—Platformy abonamentowe — Jest to oprogramowanie zamknięte, udostępniające swoje usługi w chmurze, SaaS (Software as a Service)[1]. Za takie usługi przedsiębiorca prowadzący sklep płaci abonament. Platformy abonamentowe dają możliwość stworzenia standardowego sklepu bazując na gotowych rozwiązaniach, nie pozwala jednak na wprowadzanie zmian i dodawanie niestandardowych funkcji w sklepie. Jedną z najbardziej rozpoznawanych platform tego typu jest Shopify.

— Platformy dedykowane — Tworzone od zera oprogramowania, pisane zazwyczaj przez zespół programistów pod zamówienie firmy chcącej otworzyć sklep. Programiści mogą dodać niestandardowe funkcje jakich potrzebuje firma prowadząca sklep. Sklep po utworzeniu potrzebuje w większości przypadków również osób, które będą zajmować się jego utrzymaniem.

Sklepy internetowe w których cały proces zamawiania, płatności i dostawy realizowany jest online nazywamy sklepem bezpośrednim[1]. Istnieją również sklepy w których klient może znaleźć towar, dokonać jego rezerwacji lub nawet płatności ale forma odebrania towaru lub wykonania usługi pozostaje w tradycyjnej formie. Takie sklepy nazywamy pośrednimi.

Można również wymienić kilka modeli sklepów internetowych, tudzież sposobów w jakie firma chce wykorzystywać daną platformę.

Klasyczny sklep internetowy to taki w którym firma przedstawia swoje produkty, a klient może wybrać z oferty produkt który go interesuje, dokonać płatności i wybrać sposób dostawy. Jest to najczęściej spotykany rodzaj sklepu i charakteryzuje się tym że firma posiada lub może wyprodukować towar po zamówieniu i przekazać go w określonym czasie.

Sklep dropshippingowy jest rozpoczętym kilka lat temu trendem, który wydaje się ze obecnie jest również bardzo popularny. Z punktu widzenia klienta nie różni się on niczym. Jednak z punktu widzenia właściciela sklepu różnica jest znacząca, handluje on towarem którego nie posiada na magazynie i nie interesują go również kwestie dostawy. Wszystko to leży po stronie hurtowni która wytwarza dany towar i z którą właściciel sklepu podpisuje umowę. Z punktu widzenia tworzenia takiego sklepu ważna jest integracja danych o dostępności towaru w hurtowni z ofertą na stronie sklepu.

Sklep Marketplace to miejsce w którym wielu przedsiębiorców może wystawiać swoje oferty[1]. Sklep ten przypomina coś w stylu bazarku na którym każdy może wystawić swoje produkty, a klient korzystając ze sklepu przebiera w ofertach różnych przedsiębiorców. Znanymi marketplace w Polsce są Allegro, czy OLX.

System rezerwacji to platforma na której można dokonać rezerwacji usługi, produktu, bądź udziału w jakimś przedsięwzięciu. Idealnymi przykładami takich platform są platformy do zakupu biletów na mecz, bądź koncert, czy zarezerwowanie miejsca w hotelu. Takiego typu platformy nie muszą realizować dostaw towaru

W internecie możemy znaleźć również sklepy sprzedające kursy video bądź ebooki. W obu tych przypadkach usługa, towar jaki zamawia klient dostarczany jest natychmiastowo przez przesłanie bądź udostępnienie widoku video, czy tekstu. Sklep musi więc oprócz przedstawienia oferty sprzedażowej i obsłużenia płatności zajmować się również udostępnieniem zakupionych materiałów klientowi.

W poprzednim podrozdziale wspomniano o sklepach B2B, a więc sklepach w których stroną sprzedającą i kupującą są firmy. Takiego typu platformy w ogromnej większości muszą być tworzone jako aplikacje dedykowane do klienta i spełniać wymogi stawiane przez właściciela sklepu i firmy będące klientami. Platformy tego typu często muszą obsługiwać integracje z innymi systemami i korzystać z systemów bazodanowych.

Jak wynika z powyższego tekstu istnieje wiele rodzajów sklepów internetowych zarówno ze względu na potrzeby jak i ze względu na sposób wykonania sklepu. Ma to związek z tym że e-Commerce pojawia się w coraz to większej liczbie obszarów naszego życia i musi odpowiadać na zróżnicowane wymagania klientów i firm prowadzących sklepy.

# Technologie i narzędzia do tworzenia sklepu internetowego

W tym rozdziale przedstawiono najważniejsze technologie wykorzystywane do tworzenia sklepów internetowych w analizowanych w pracy sposobach. W pierwszym podrozdziale opisano technologie programistyczne stosowane przy tworzeniu dedykowanego sklepu internetowego w tak zwanym podejściu od zera. W drugim podrozdziale przedstawiłem możliwości korzystania z gotowych platform SaaS i dokładniej opisałem wykorzystywaną przy pisaniu pracy platformę Shopify.

## 2.1. Technologie i narzędzia programistyczne

Język programowania Java.

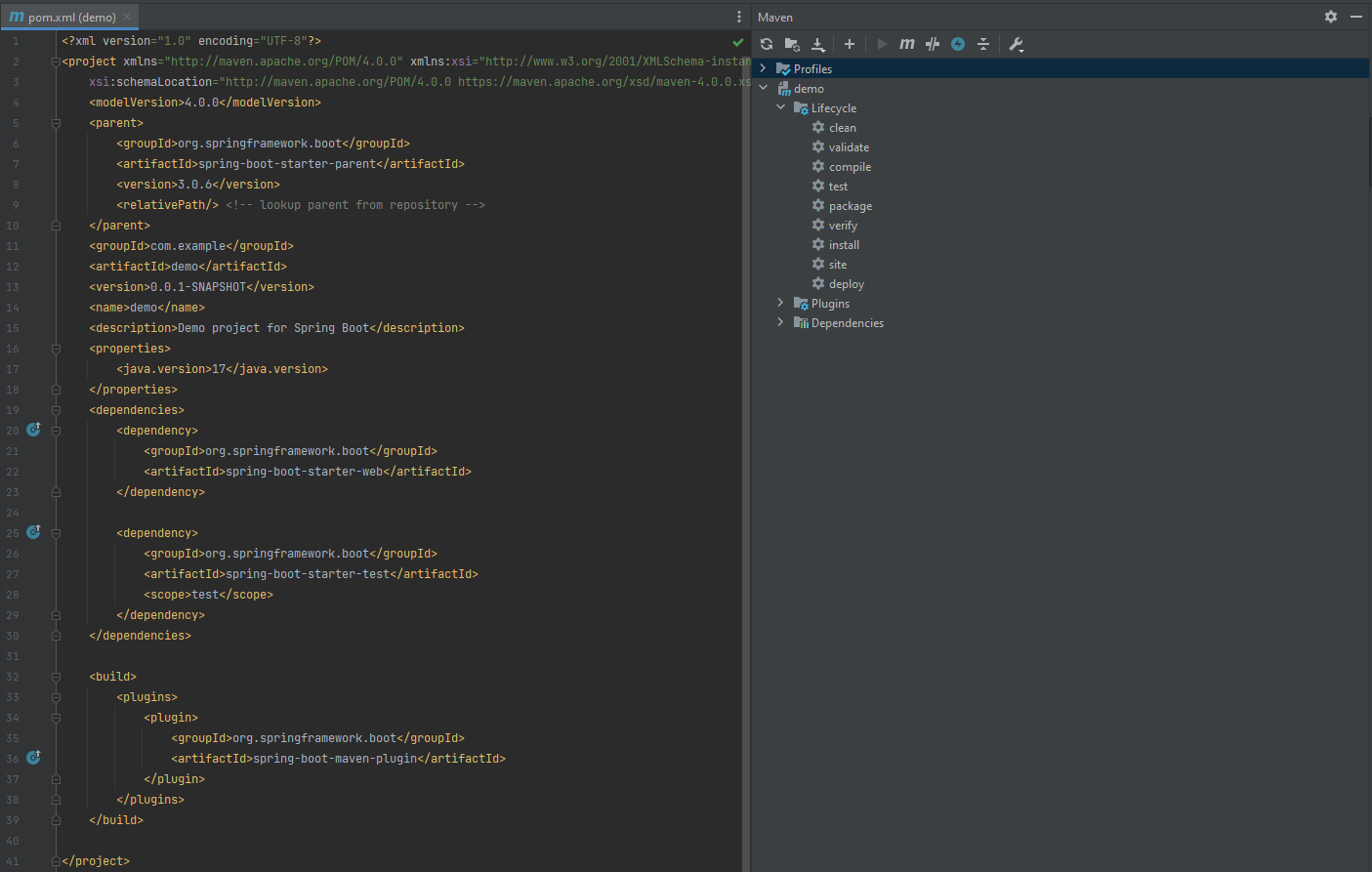
Java to wysokopoziomowy, obiektowy język programowania którego początek ustala się na rok 1991. Wtedy to firma o nazwie Sun z Patrickiem Naughtonem oraz Jamesem Goslingiem postanowili stworzyć prosty język, który można uruchamiany na różnych platformach i systemach[2]. Pierwsza wersja Javy, Java 1.0 ukazała się w 1996 roku i mimo upływającego czasu Java jest cały czas, aktualnym i szeroko wykorzystywanym językiem programowania. Jeszcze kilka lat temu Java prowadziła w większości rankingów badających popularność języków programowania, w tym momencie ze względu na szerokie zastosowanie w rankingach wyprzedziły ją JavaScript i Python. W kontekście możliwości uruchamiania raz napisanego kodu Java w różnych systemach należy powiedzieć że kod pisany przez programistę kompilowany jest do kodu bajtowego, ten z kolei wykonywany jest przez maszynę wirtualną Javy JVM. Maszyna ta przetwarza kod do instrukcji zrozumiałych przez system na którym się znajduję. Dzięki temu ten sam kodu napisany w Javie można odpalić na Windowsie, Linuxie, czy MacOS. W Javie duży nacisk postawiono na kompatybilność wsteczną to znaczy że mimo przejścia na nowszą wersję Javy programy napisane w starszej wersji również działają[2].

Platforma Spring Boot

Spring Boot to narzędzie, które opiera się na frameworku Spring. Twórcy Spring Boot skonfigurowali framework Spring do postaci, której potrzebuje większość użytkowników i udostępnili go jako otwartą platformę. Dzięki temu programiści nie muszą poświęcać ogromu czasu na konfiguracje i o wiele szybciej mogą przystąpić do tworzenia oprogramowania. Tworząc projekt oparty na Spring Boot programista może dociągać zależności do projektu i używać dostarczane przez nich funkcjonalności w swoich programach. Spring Boot oferuje zależności związane między innymi z tworzeniem serwerów webowych, baz danych, bezpieczeństwa, testowania aplikacji, czy usług chmurowych. Spring Boot jest nieustannie aktualizowany, poprawiany i podnoszone są jego wersje. W momencie pisania pracy najnowszą stabilną wersją jest wersja 3.0.6. Spring Boot stał się standardem, a jego znajomość jest podstawowym warunkiem koniecznym do tworzenia aplikacji webowych w języku Java.

Narzędzie Maven

Maven to potężne narzędzie wykorzystywane do automatyzowania budowy projektu w języku Java. To właśnie z wykorzystaniem Mavena programista może dodawać do projektu nowe zależności, usuwać je, lub zmieniać wersję z jakiej aktualnie korzysta program, wszystko to dokonuje się przez modyfikację pliku pom.xml, którego podstawowa konfiguracja widoczna jest na zrzucie ekranu poniżej. Oprócz budowania projektu i aktualizowania zależności Maven posiada wiele innych funkcjonalności, takich jak testowanie kodu, tworzenie plików wykonywalnych, czy tworzenie pakietów instalacyjnych. Funkcjonalności te widoczne są po prawej stronie umieszczonego poniżej zrzutu ekranu. Maven jest szeroko stosowany w środowisku programistów Java, a wpływ na to ma łatwość w korzystaniu i bogaty zestaw funkcjonalności.



// Podpis : Plik pom.xml i funkcjonalności Maven

JPA i Hibernate

JPA (ang. Java Persistence API) to specyfikacja, która opisuje jak powinna działać zamiana elementów wyciągniętych z encji w bazie danych i tworzenie na podstawie tych danych obiektu w programie. JPA jest jedynie specyfikacją i do działania potrzebuje implementacji. Hibernate jest jedną z takich implementacji, najpopularniejszą. Aby wykorzystać Hibernate w aplikacji opartej o Spring Boot wystarczy dodać do projektu zależność Spring Data JPA. Dzięki wykorzystywaniu Hibernate programista nie musi pisać skomplikowanych zapytać SQL. W zamian może korzystać z gotowych metod dostarczanych przez Hibernate, oczywiście mimo korzysania z gotowych rozwiązań w dalszym ciągu istnieje możliwość pisania własnych metod komunikujących się z bazą danych i używania ich w programie.

IntelliJ IDEA

Do pisania kodu Java, konfiguracji aplikacji Spring-Boot za pomocą Mavena, czy dodawania adnotacji Hibernate potrzebne jest wygodne środowisko programistyczne, takim środowiskiem jest IntelliJ IDEA. IntelliJ IDEA został stworzony przez firmę JetBrains dla programistów pracujących w języku Java oraz kilku innych językach programowania, takich jak Kotlin, Scala, czy Groovy. IDE oferuje szereg narzędzi umożliwiających szybkie i wydajne tworzenie, testowanie oraz debugowanie aplikacji.IntelliJ IDEA posiada dużą liczbę funkcjonalności, między innymi automatyczne uzupełnianie kodu, refactorowanie, system kontroli wersji (na przykład git) i wiele innych. IntelliJ IDEA jest w tym momencie jednym z najpopularniejszych IDE dla programistów Javy i jest szeroko stosowany w dużej liczbie projektów na całym świecie.

Angular

Angular jest łatwym w obsłudze frameworkiem często stosowany w aplikacjach webowych po stronie frontendu. Ułatwia on tworzenie wydajnych i zaawansowanych aplikacji typu SPA ( ang. Single Page Application). Framework został zaprojektowany przez firmę Google i zaprezentowany światu w 2014 roku, w dalszym ciągu jest on wspierany i rozwijany przez tą właśnie firmę. Framework ten jest zorientowany na komponenty dzięki czemu łatwo odnaleźć się w strukturze projektu. Podczas tworzenia aplikacji z wykorzystaniem frameworka Angular używa się HTML, CSS i języka TypeScript. TypeScript to statycznie typowany język programowania, będący rozszerzeniem języka JavaScript. Oferuje on funkcje takie jak dziedziczenie, interfejsy, klasy, moduły i wiele innych, które ułatwiają rozwijanie zaawansowanych aplikacji internetowych. TypeScript kompilowany jest do języka JavaScript, dzięki czemu ​​można używać go w środowisku przeglądarki internetowej.

Visual studio code

Frontend również potrzebuj wygodnego środowiska do tworzenia oprogramowania. Visual Studio Code to otwarty i bezpłatny edytor kodu źródłowego, opracowany przez Microsoft. Jest to popularne narzędzie, które zapewnia zaawansowane funkcje edycji, debugowania oraz kontroli wersji, umożliwiając programistom tworzenie aplikacji w sposób łatwy i wydajny. Dzięki obsłudze wielu języków programowania i systemów operacyjnych, Visual Studio Code jest wygodnym narzędziem dla programistów z różnych dziedzin.

Architektura REST

REST (ang. REpresentational State Transfer) to styl architektury stanowiący standard w komunikacji webowej, upraszczający komunikacje między aplikacjami internetowymi. Systemy zgodne z architekturą REST wyróżniają się bezstanowością oraz rozdzieleniem aplikacji klienckiej i serwera[4]. Rozdzielenie aplikacji klienckiej i serwera charakteryzuje się niezależnością jednej aplikacji od drugiej. Zmiany na serwerze nie wpływają na działania aplikacji klienckiej i odwrotnie. Dzięki temu aplikacje można rozwijać osobno, niezależnie od siebie. W architekturze REST bardzo ważnym elementem są endpointy, które wystawia serwer. Dowolny klient wysyłając zapytaniem powinien otrzymać taką samą odpowiedź od serwera. Ważne jest aby zwrócić uwagę na format przesyłanych danych i odpowiednio je obsłużyć po obu stronach. Bezstanowość natomiast objawia się tym że serwer nie musi nic wiedzieć o tym, w jakim stanie jest klient i odwrotnie. W ten sposób zarówno serwer, jak i klient mogą zrozumieć każdą otrzymaną wiadomość, nawet nie widząc poprzednich wiadomości[4].

MySQL

Oprócz aplikacji klienckiej i serwera sklep internetowy potrzebuje również bazy danych. W sklepie internetowym tworzonym w tej pracy wykorzystano bazę danych MySQL. Jak wskazuje nazwa zarządzanie bazą danych oparte jest na języku SQL. MySQL jest rozwiązaniem darmowym do wykorzystania w niekomercyjnych rozwiązaniach, firmy chcąc z niego korzystać muszą wykupić licencje. MySQL jest popularnym rozwiązaniem stosowanym przez największe aplikacje na świecie takie jak Facebook, Twitter czy YouTube.

## 2.2. Platforma abonamentowa – Shopify

Na rynku istnieje wiele ofert sklepów już stworzonych, gotowych do działania na których wystarczy umieścić swoje produkty i wybrać pasujące do sklepu rozwiązania z przygotowanych już możliwości na przykład dostawy, czy wyglądu sklepu. Sklep taki jest niejako wynajmowany za cenę kilkuset złotych miesięcznie i w przypadku zaprzestania płatności abonamentu usługodawca go zamyka. W zamian za miesięczny abonament rozwiązanie usługodawcy działa wraz z wszystkimi koniecznymi funkcjonalnościami takimi jak zapewnienie bezpieczeństwa, tworzenie kopii zapasowych danych, czy aktualizacji oprogramowania. Osoba prowadząca sklep musi zajmować się jedynie aktualizowaniem oferty i cen na sklepie, całą resztę obsługuje usługodawca oferujący platformę SaaS.

Istnieje wiele sklepów działających na platformach SaaS, najpopularniejsze w Polsce platformy to: Comarch, Shoper, Shoplo, Sky-Shop, Shopify[5]. Różnią się one ceną abonamentu, niektórymi funkcjonalnościami, czy dostępnymi integracjami.

W pracy wykorzystano platformę Shopify. Jest to platforma e-Commerce, które umożliwia sprzedaż produktów i usług online i zapewnia wszystkie narzędzia i funkcje niezbędne do tworzenia sklepu i jego zarządzania takie jak koszyki zakupowe, systemy płatności, integracje z firmami kurierskimi i wiele innych. Tak jak w przypadku wszystkich platform SaaS ​​nie trzeba martwić się tutaj o hosting, aktualizacje, zabezpieczenia sklepu internetowego i kilka aspektów, które trzeba wziąć pod uwagę tworząc sklep internetowy samodzielnie. Shopify to najpopularniejsza sklepowa platforma abonamentowa na świecie[5]. Posiada 3 możliwe pakiety użytkownika Basic, Shopify i Adavnced. Kosztują one odpowiednio 32, 92 i 399 dolarów miesięcznie, a różnią się przede wszystkim liczbą kont administratorów które w pakiet kolejno występują w liczbach 2,5 i 15 oraz raportami o sprzedaży jakie generuje dla nas sklep. W wersji Basic istnieje dostęp tylko do podstawowych raportów. Wszystkie 3 pakiety ograniczone są maksymalnie 1000 hurtowni z jakich korzysta sklep, co nie powinno stanowić problemu dla żadnego z zorientowanych na daną branżę sklepów. Platforma Shopify to platforma międzynarodowa, która obsługuje ponad milion sklepów e-Commerce na całym świecie[5].

//Sky-Shop

Jeśli chcesz sprzedawać w Polsce

//Shopify

Jeśli chcesz sprzedawać za granicę

# Tworzenie sklepu internetowego

W tym rozdziale przyjrzę się dwóm różnym podejściom do tworzenia sklepów internetowych. Pierwszym z nich będzie sklep stworzony z wykorzystaniem technologii Java Spring-Boot i Angular. Przybliżone zostaną kluczowe techniczne zagadnienia wykorzystywane w tym rozwiązaniu, oraz zaprezentowany zostanie efekt końcowy zarówno od strony technicznej jak i od strony użytkownika.

Drugim podejściem, które zostanie bliżej przedstawione, będzie sklep zbudowany na popularnej platformie Shopify. Dowiemy się, jak łatwo można stworzyć sklep online, korzystając z gotowych narzędzi tej platformy i jaki jest końcowy efekt.

## 3.1. Sklep tworzony z wykorzystaniem technologii programistycznych

W tym podrozdziale tego rozdziału skupię się na przedstawieniu sklepu internetowego, który został stworzony przy użyciu technologii Spring-Boot i Angular. Te dwie technologie stanowią popularne i potężne narzędzia do budowy skalowalnych i nowoczesnych aplikacji internetowych. Opiszę też stworzoną na potrzeby aplikacji bazę danych. Podczas tworzenia aplikacji sklepu internetowego korzystałem z lekcji umieszczonych w kursie Full Stack: Angular and Java Spring Boot E-Commerce Website.[6]

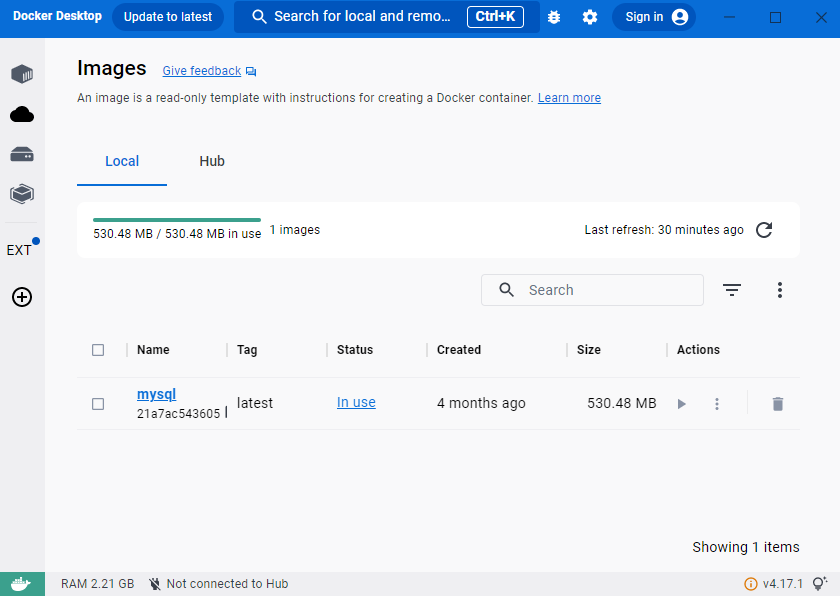
W pierwszej kolejności zajmę się wspomnianą wyżej bazą danych, następnie opiszę jak framework Spring-Boot wraz z dodawanymi zależnościami został wykorzystany do komunikacji z bazą danych i tworzenia innych funcjonalności po stronie backendu.

Przedstawię również frontendową stronę aplikacji, która została zaimplementowana przy użyciu technologii Angular. Skupie się na podstawowych elementach aplikacji klienckiej tworzonej z wykorzystaniem tej technologii, oraz na komunikacji między frontendem a backendem które realizowana jest za pomocą REST API.

Ostatni element opisu sklepu tworzonego w technologiach Java Spring-Boot i Angular to przybliżenie efektu końcowego, a więc funkcjonalności sklepu z których może korzystać użytkownik. Wszystkie funkcjonalności sklepu zostaną przedstawione w formie opisu i dołączonych zrzutów ekranu.

**Baza danych**

W celu stworzenia bazy danych dla aplikacji sklepu internetowego zainstalowałem Dockera na swoim systemie operacyjnym. Docker to platforma, która pozwala na łatwe tworzenie, uruchamianie i zarządzanie kontenerami. Po zainstalowaniu Dockera musiałem pobrać obraz MySQL z oficjalnego repozytorium Dockera. Obraz zawiera gotową do użycia konfigurację bazy danych MySQL. Po pobraniu obrazu, uruchomiłem kontener na podstawie tego obrazu. Podczas uruchamiania określiłem opcje takie jak porty, na których będzie działać baza danych i hasło do użytkownika root.



Gdy kontener MySQL działał już poprawnie, skonfigurowałem połączenie z bazą danych z poziomu narzędzia do zarządzania bazą danych MySQL Workbench. Do konfiguracji połączenia podałem adres IP lub nazwę hosta, port, nazwę użytkownika i hasło.

Następnie za pośrednictwem narzędzia MySQL Workbench utworzyłem nowy schemat bazy danych i 8 tabel.

Adress – tabela zawierająca dane adresowe wykorzystywane w tabeli zamówień, tabela zawiera informacje o kraju, stanie, mieście, ulicy, czy kodzie pocztowym

Country – tabela zawierająca informacje o kraju

State – tabela zawierająca informacje o konkretniejszej mniejszej niż kraj jednostce terytorialnej, w przypadku Polski będzie to województwo

Customer – tabela zawierająca informacje o klientach

Order\_item – tabela zawierająca informacje o konkretnym zamówionym produkcie, biorąca pod uwagę jego ilość i cenę

Orders – tabela której rekord zawiera w sobie co najmniej jeden zamówiony produkt, oprócz tego zawiera wszystkie istotne informacje o zamówieniu takie jak daty zamówienia, status, czy całkowita cena

Product – tabela zawierająca informacje o produkcie takie jak liczba jednostek w magazynie, opis, zdjęcie, data produkcji

Product\_category – tabela, która pomaga grupować produkty w kategorie

|  |
| --- |
| CREATE TABLE IF NOT EXISTS `full-stack-ecommerce`.`product` (  `id` BIGINT(20) NOT NULL AUTO\_INCREMENT,  `sku` VARCHAR(255) DEFAULT NULL,  `name` VARCHAR(255) DEFAULT NULL,  `description` VARCHAR(255) DEFAULT NULL,  `unit\_price` DECIMAL(13,2) DEFAULT NULL,  `image\_url` VARCHAR(255) DEFAULT NULL,  `active` BIT DEFAULT 1,  `units\_in\_stock` INT(11) DEFAULT NULL,  `date\_created` DATETIME(6) DEFAULT NULL,  `last\_updated` DATETIME(6) DEFAULT NULL,  `category\_id` BIGINT(20) NOT NULL,  PRIMARY KEY (`id`),  KEY `fk\_category` (`category\_id`),  CONSTRAINT `fk\_category` FOREIGN KEY (`category\_id`) REFERENCES `product\_category` (`id`)  ) |

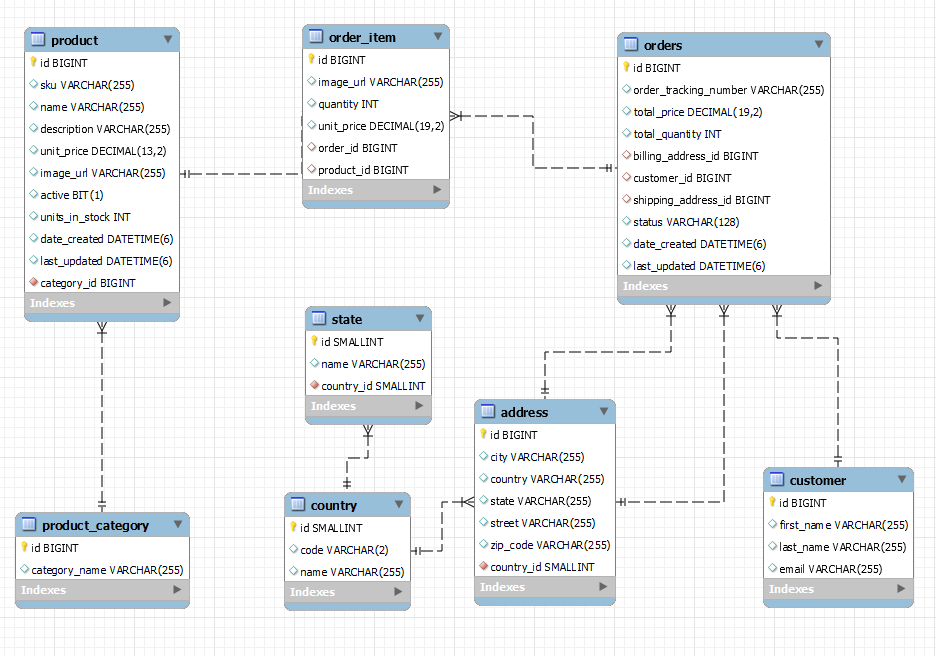
// Fragment kodu wykorzystany do tworzenia tabeli Product

Tabele Country, State, Product, Product\_category zostały wypełnione danymi. Dane przykładowych produktów uzyskałem z wspomnianego już kursu tworzenia sklepu internetowego[6]. Dane wprowadzone do tabeli Country i State zostały jednak zmodyfikowane tak, aby dało się obsługiwać zamówienia z Polski.

|  |
| --- |
| INSERT INTO product\_category(category\_name) VALUES ('Books');  INSERT INTO product\_category(category\_name) VALUES ('Coffee Mugs');  INSERT INTO product\_category(category\_name) VALUES ('Mouse Pads');  INSERT INTO product\_category(category\_name) VALUES ('Luggage Tags'); |

// Fragment kodu wprowadzający wartości do tabeli Product\_category

Ostateczną postać bazy danych można zaobserwować na poniższym diagramie.



//Diagram ERD bazy danych utworzonej na potrzeby sklepu internetowego

**Backend**

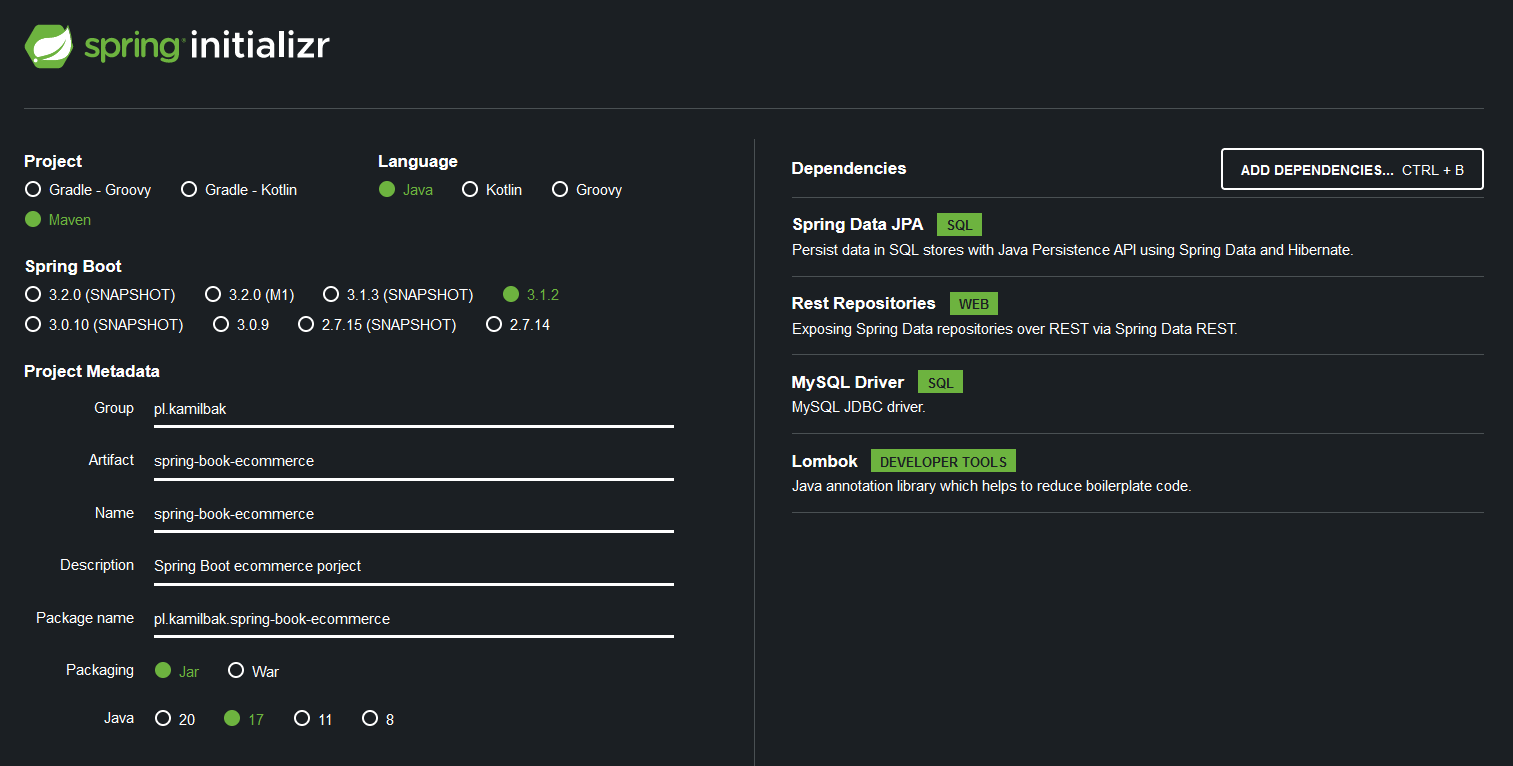
W celu rozpoczęcia pracy nad backendową stroną aplikacji stworzyłem nowy projekt korzystając z narzędzia spring initializr. Do swojego projektu wybrałem Jave w wersji 17 i Spring Boot w wersji 3.1.2. Do projektu dodałem 4 zależności Spirng Data JPA, Rest Repositiories, MySQL Driver i Lombok.

Spirng Data JPA – Spring Data JPA to rozszerzenie frameworku Spring, które ułatwia pracę z technologią Java Persistence API (JPA) do mapowania obiektowo-relacyjnego (ORM). JPA pozwala na zapisywanie i pobieranie danych z bazy danych, używając obiektów Java zamiast bezpośrednio operować na zapytaniach SQL.

Rest Repositiories – Rest Repositories to moduł Spring Data, który umożliwia automatyczne tworzenie interfejsów repozytoriów JPA jako punktów dostępowych do danych. Oznacza to, że po dodaniu odpowiednich adnotacji, Spring Boot automatycznie udostępni standardowe operacje CRUD (Create, Read, Update, Delete) dla Twoich encji w ramach komunikacji REST.

MySQL Driver – MySQL Driver to zależność, która dostarcza niezbędne klasy i narzędzia do komunikacji z bazą danych MySQL. Dzięki tej zależności, projekt Spring Boot będzie w stanie nawiązać połączenie z bazą danych MySQL i wykonywać operacje związane z odczytem i zapisem danych.

Lombok – Lombok to narzędzie wspomagające rozwijanie aplikacji w Javie. Dzięki Lombokowi możesz znacznie skrócić i uprościć kod poprzez dodawanie adnotacji do Twoich klas. Lombok automatycznie generuje za Ciebie niektóre standardowe metody, takie jak gettery, settery, konstruktory i wiele innych.



// Zrzut ekranu z wybranymi ustawieniami początkowymi projektu i dodanymi zależnościami

Encje

Encje to klasy w projekcie, które wykorzystują technologię Java Persistence API (JPA) do mapowania obiektowo-relacyjnego (ORM), aby pozwolić na łatwe zapisywanie i pobieranie danych z bazy.

Poniżej przedstawiono jedną z takich klas i opisano najważniejsze fragmenty kodu.

|  |
| --- |
| @Entity  @Table(name="country")  @Getter  @Setter  public class Country {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  @Column(name="id")  private int id;  @Column(name="code")  private String code;  @Column(name="name")  private String name;  @OneToMany(mappedBy = "country")  @JsonIgnore  private List<State> states;  } |

// kod klasy Country

Kod zawiera kilka adnotacji, które warunkują działanie klasy.

@Entity to adnotacja oznaczająca, że klasa "Country" jest encją JPA, która będzie mapowana do tabeli w bazie danych.

@Table(name="country") toadnotacja określająca nazwę tabeli w bazie danych, do której będzie mapowana ta encja. W tym przypadku tabela nazywa się "country".

@Getter i @Setter to adnotacje Lombok, które tworzą gettery i setery dal każdego z pól klasy, dzięki temu znacznie zmniejsza się rozmiar pisanego kodu.

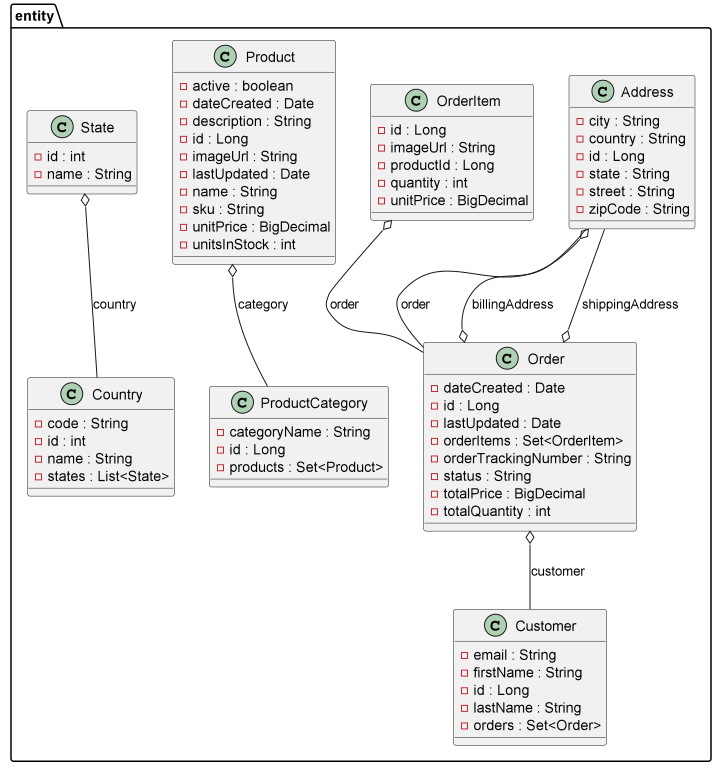
@Id to adnotacja wskazująca, że pole "id" jest kluczem głównym w tabeli "country".

@GeneratedValue(strategy = GenerationType.IDENTITY) ta adnotacja wskazuje na sposób przypisywanie id do nowych obiektów klasy, powyższa opcja to pobieranie id z obiektu bazy danych, który generuje id automatycznie podczas tworzenia nowych rekordów.

@Column(name="\*") to adnotacja która przypisuje pole w klasie do konkretnej kolumny w tabeli bazy danych.

@OneToMany(mappedBy = "country") to adnotacja, która definiuje relację jeden-do-wielu. Oznacza to, że jeden kraj może mieć wiele stanów. W klasie State znajduje się adnotacja @ManyToOne co pozwala na połączenie ze sobą tych dwóch encji

W podobny sposób utworzono 7 pozostałych klas encji, każda dla konkretnej encji z bazy danych.



Repozytoria JPA

Repozytoria JPA są kluczowym elementem frameworka Spring Data JPA. Są to interfejsy, które definiują zestaw metod dostępowych do danych. Te metody pozwalają na wykonywanie różnych operacji na danych bez konieczności pisania konkretnych zapytań SQL. Repozytoria pełnią rolę tzw. warstwy dostępu do danych (Data Access Layer).

Poniżej przedstawiono jedno z utworzonych w projekcie repozytoriów ProductRepository.

|  |
| --- |
| public interface ProductRepository extends JpaRepository<Product, Long> {  Page<Product> findByCategoryId(@Param("id") Long id, Pageable pageable);  Page<Product> findByNameContaining(@Param("name") String name, Pageable pageable);  } |

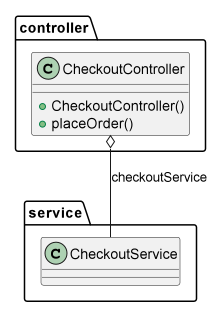
// kod klasy ProductRepository

Każde z utworzonych w projekcie repozytoriów rozszerza interfejs JpaRepository podając klasę Entity której będą dotyczyć zapytania i typ jakiego jest pole id, w powyższym przypadku jest to typ Long. Oprócz standardowych metody dostarczanych przez interfejs JpaRepository istnieje możliwość dodawania kolejnych metod zachowując konwencję nazewnictwa. W powyższym przykładzie metoda findByCategoryId() zwróci wszystkie produkty dla danej kategorii. Spring Boot analizując nazwę metody sam stworzy adekwatne zapytanie SQL i pozwoli na korzystanie z metody pod endpointem api/products/search/findByCategoryId?id= . Druga metoda findByNameContaining() pozwala na znalezienie produktów których nazwa zawiera konkretny ciąg znaków. Korzystanie z repozytoriów JPA znacznie ułatwia tworzenie zapytań do bazy, jeśli jednak istnieje potrzeba napisania niestandardowego zapytania, nieobsługiwanego przez JPA wystarczy zastosować adnotację @Query i napisać własne zapytanie SQL, z którego będzie korzystać wybrana metoda interfejsu.

Kontroler i serwis obsługujący przyjmowane zamówień

W celu obsłużenia procesu przyjmowania zamówień stworzono kontroler CheckoutController i serwis CheckoutService. Klasa CheckoutController obsługuje żądania związane z procesem składania zamówienia w aplikacji, przetwarza je i zwraca odpowiedź. Aby złożyć zamówienie aplikacja kliencka musi odwołać się do metody wystawionej pod endpointem /api/checkout/purchase i przekazać dane dotyczące zamówienia w formacie JSON. Po stronie backendu dane te zostały zebrane w klasie Purchase. Do przetworzenia żądania wykorzystywana jest metoda placeOrder z serwisu CheckoutService , zapisuje ona zamówienie w bazie danych i zwraca numer śledzenia zamówienia.

|  |
| --- |
| @RestController  @RequestMapping("/api/checkout")  public class CheckoutController {  private CheckoutService checkoutService;  public CheckoutController(CheckoutService checkoutService) {  this.checkoutService = checkoutService;  }  @PostMapping("/purchase")  public PurchaseResponse placeOrder(@RequestBody Purchase purchase) {  PurchaseResponse purchaseResponse = checkoutService.placeOrder(purchase);  return purchaseResponse;  }  } |



**Frontend**

Frontend aplikacji e-commerce został stworzony z użyciem framoweorka Angular. Skupia się na zapewnieniu użytkownikom wydajnego i przyjemnego doświadczenia zakupów online. Aplikacja komunikuje się z backendem opartym na Spring Boot.

Tak jak większość projektów tworzonych z wykorzystaniem Angular, moja aplikacja składa się z mniejszych komponentów, które są wykorzystywane w odpowiednich częściach programu. Wspomniane komponenty wymieniono poniżej.

**Cart Details** – Ten komponent wyświetla szczegóły koszyka zakupowego, w tym listę produktów, ich ilości oraz łączny koszt zamówienia.

**Cart Status** – Komponent ten prezentuje użytkownikowi podsumowanie koszyka w widoczny sposób na każdej stronie. Wyświetla liczbę pozycji w koszyku oraz łączną kwotę.

**Checkout** – Komponent Checkout umożliwia użytkownikowi finalizację zamówienia. Pozwala na wprowadzenie danych kontaktowych, adresu dostawy oraz informacji płatniczych.

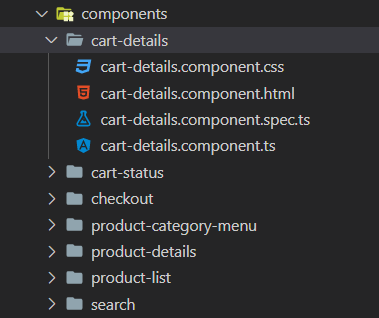
**Product Category Menu** – Komponent odpowiedzialny za wyświetlanie menu kategorii produktów, które ułatwiają nawigację po stronie.

**Product Details** – Komponent wyświetla szczegóły dotyczące wybranego produktu, takie jak zdjęcie, opis, cena itp.

**Product List**– Komponent ten wyświetla listę produktów w określonej kategorii lub ogólnie. Umożliwia filtrowanie i sortowanie produktów.

**Search**– Komponent umożliwia użytkownikowi wyszukiwanie produktów na podstawie wprowadzonej nazwy.

Każdy z komponentów składa się z 4 plików. Pliku zawierającego kod html, stanowiącego wygląd komponentu, pliku type-script odpowiadającego za logikę działania danego komponentu i przekazywanie i pobieranie z niego danych, plu definiującego style css i pliku używanego do pisania testów.



Przemieszczanie się pomiędzy komponentami w widoku odbywa się za pośrednictwem routingu. W kontekście aplikacji Angular, routing umożliwia określenie, jakie komponenty powinny być wyświetlane w zależności od wprowadzonej przez użytkownika ścieżki URL. W ścieżce do komponentu można przekazać również parametry takie jak :id, czy :keyword, które pozwolą na załadowanie odpowiednich danych do komponentu. Należy zabezpieczyć też aplikację na wypadek podania niepoprawnej ścieżki, lub pustej ścieżki, opisywana aplikacja w obu przypadkach przekierowuje użytkownika do komponentu wyświetlającego wszystkie produkty.

|  |
| --- |
| const routes : Routes = [  {path: 'checkout', component: CheckoutComponent},  {path: 'cart-details', component: CartDetailsComponent},  {path: 'products/:id', component: ProductDetailsComponent},  {path: 'search/:keyword', component: ProductListComponent},  {path: 'category/:id', component: ProductListComponent},  {path: 'category', component: ProductListComponent},  {path: 'products', component: ProductListComponent},  {path: '', redirectTo: '/products', pathMatch: 'full'},  {path: '\*\*', redirectTo: '/products', pathMatch: 'full'}  ]; |

// Routing aplikacji

Aplikacja definiuje klasy takie jak CartItem, Order, Product, czy Address, które odzwierciedlają dane odbierane, lub przesyłane do backendu i określają postać tych danych. W przypadku klas wymienionych wyżej dane dotyczą odpowiednio produktu w koszyku, zamówienia, produktu w sklepie i adresu. Do tworzenia klas wykorzystano charakterystyczny dla języka TypeScript konstruktor, który tworzy pola w klasie i przypisuje do nich wartości, dzięki temu znacznie skraca się ilość pisanego kodu.

|  |
| --- |
| export class Address {  constructor(public street: string,  public city: string,  public state: string,  public country: string,  public zipCode: string) { }  } |

// kod klasy Address

Część logiki programu, oraz obsługę komunikacji z backendem, przeniesiono do serwisów. W aplikacji utworozno między innymi takie serwisy.

**ProductService** – Serwis ten odpowiada za pobieranie danych produktów z backendu oraz dostarczanie ich do komponentów, które wyświetlają listę produktów, szczegóły produktów, itp. Serwis umożliwia również wyszukiwanie produktów po nazwie i po kategorii.

**CheckoutService** – Serwis odpowiedzialny za przetwarzanie zamówień i przekazywanie danych związanych z klientem, płatnością i dostawą na backend.

**CartService** – Ten serwis zarządza zawartością koszyka, w tym dodawaniem, usuwaniem i aktualizacją produktów w koszyku.

|  |
| --- |
| getProductList(categoryId: number): Observable<Product[]> {  const url = `${this.baseUrl}/search/findByCategoryId?id=${categoryId}`;  return this.getProducts(url);  }  private getProducts(url: string): Observable<Product[]> {  return this.httpClient.get<GetResponseProducts>(url).pipe(  map(response => response.\_embedded.products)  );  } |

// fragment kodu z serwisu **ProductService zwracający produkty danej kategorii**

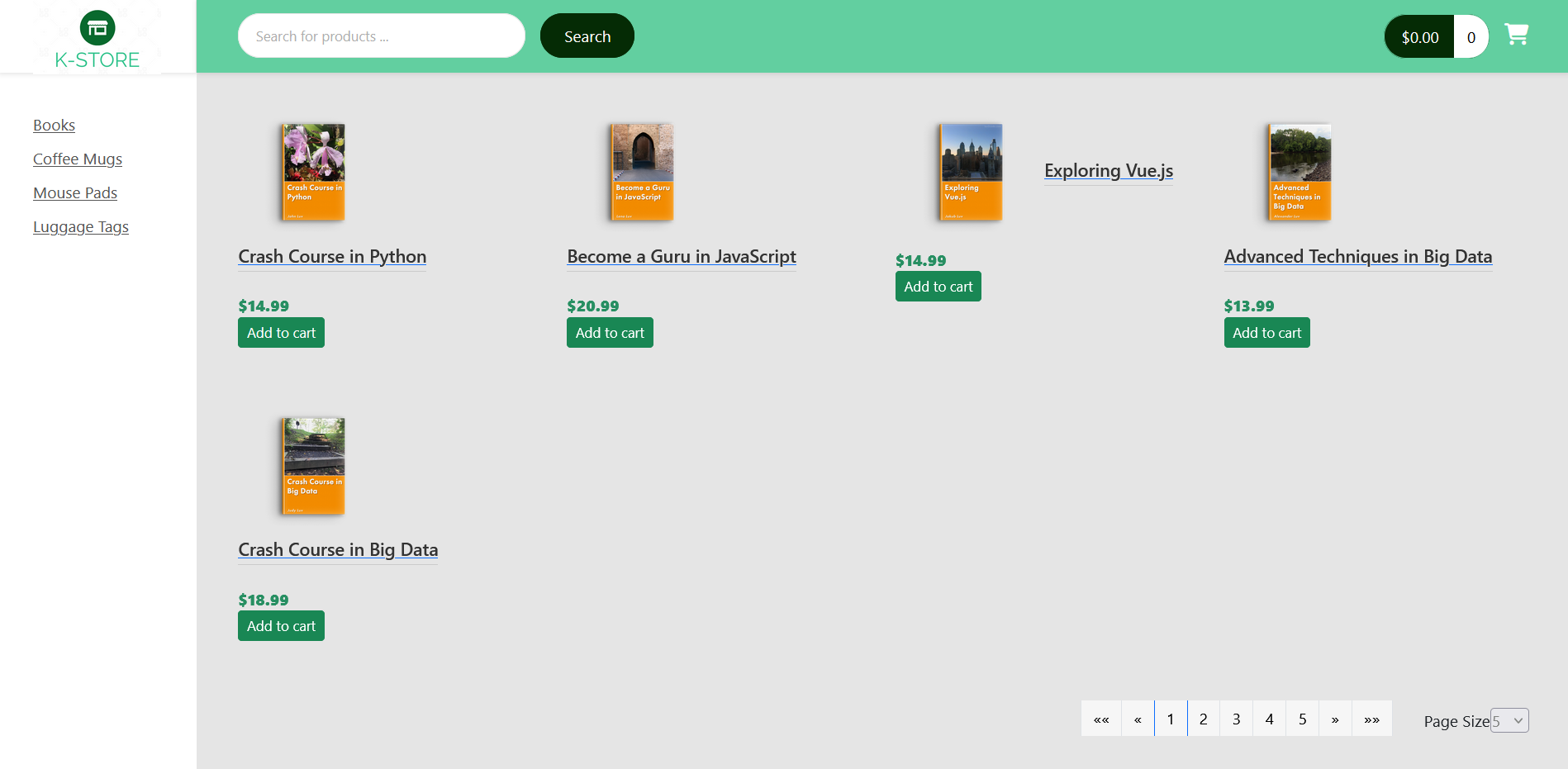
W aplikacji zaimplementowany jest również walidator, który dba o poprawność danych wprowadzanych przez użytkownika do formularzy, takich jak dane kontaktowe w procesie finalizacji zamówienia. Poniżej przedstawiono fragment kodu, który sprawdza czy podane w formularzu imię nie jest puste, nie jest jedynie spacją, lub czy nie jest krótsze niż dwa znaki.

|  |
| --- |
| <div \*ngIf="firstName?.errors?.['required'] || firstName?.errors?.['notOnlyWhitespace']">  First Name is required  </div>  <div \*ngIf="firstName?.errors?.['minlength']">  First Name must be at least 2 characters long  </div> |

// fragment kodu html wykorzystujący walidację

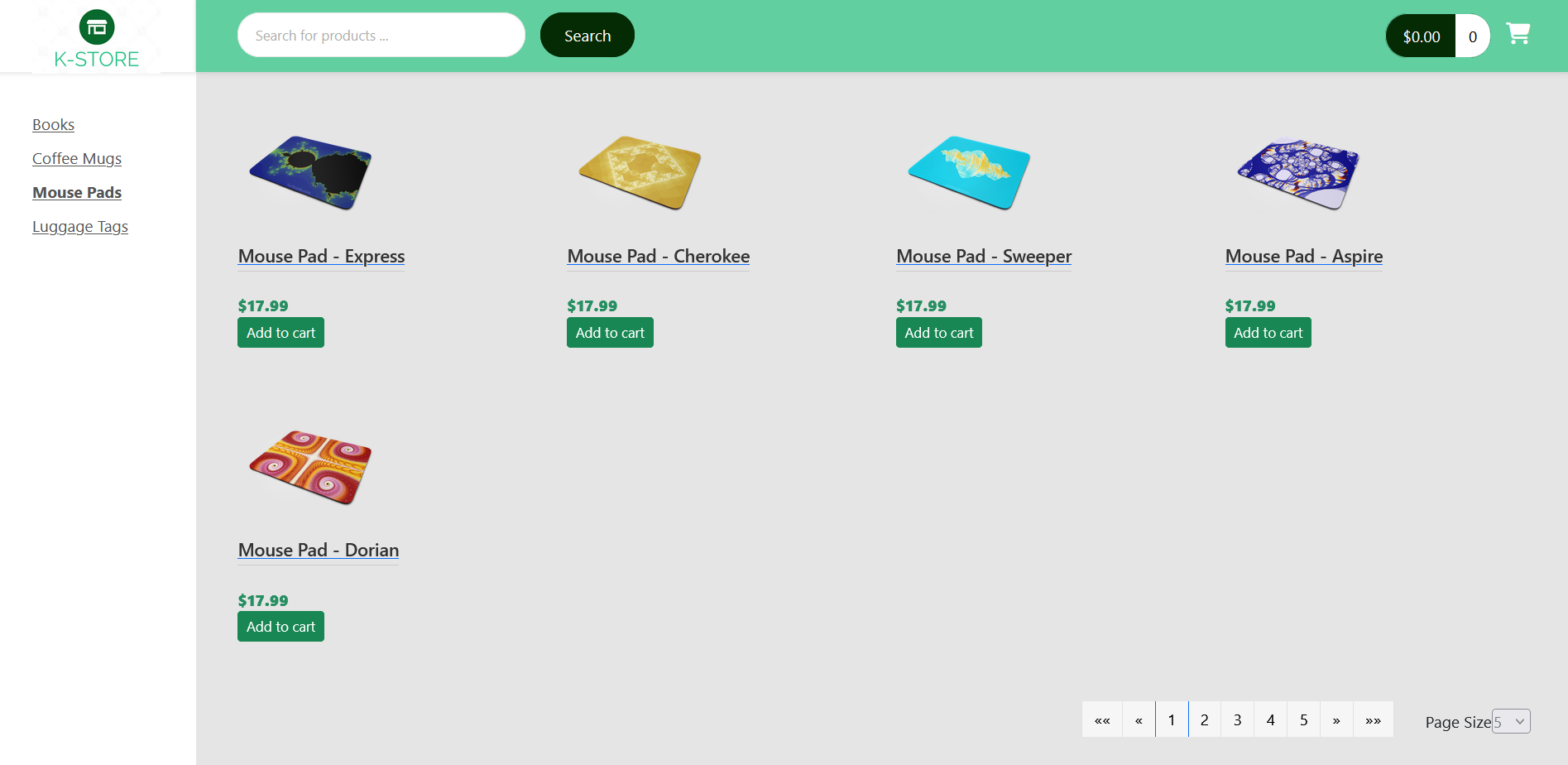
**Efekt końcowy**

Poniżej przedstawiono zrzuty ekranu z działania aplikacji, które przybliżają sposób jej funkcjonowania i pozwalają odczuć wrażenia użytkownika sklepu.



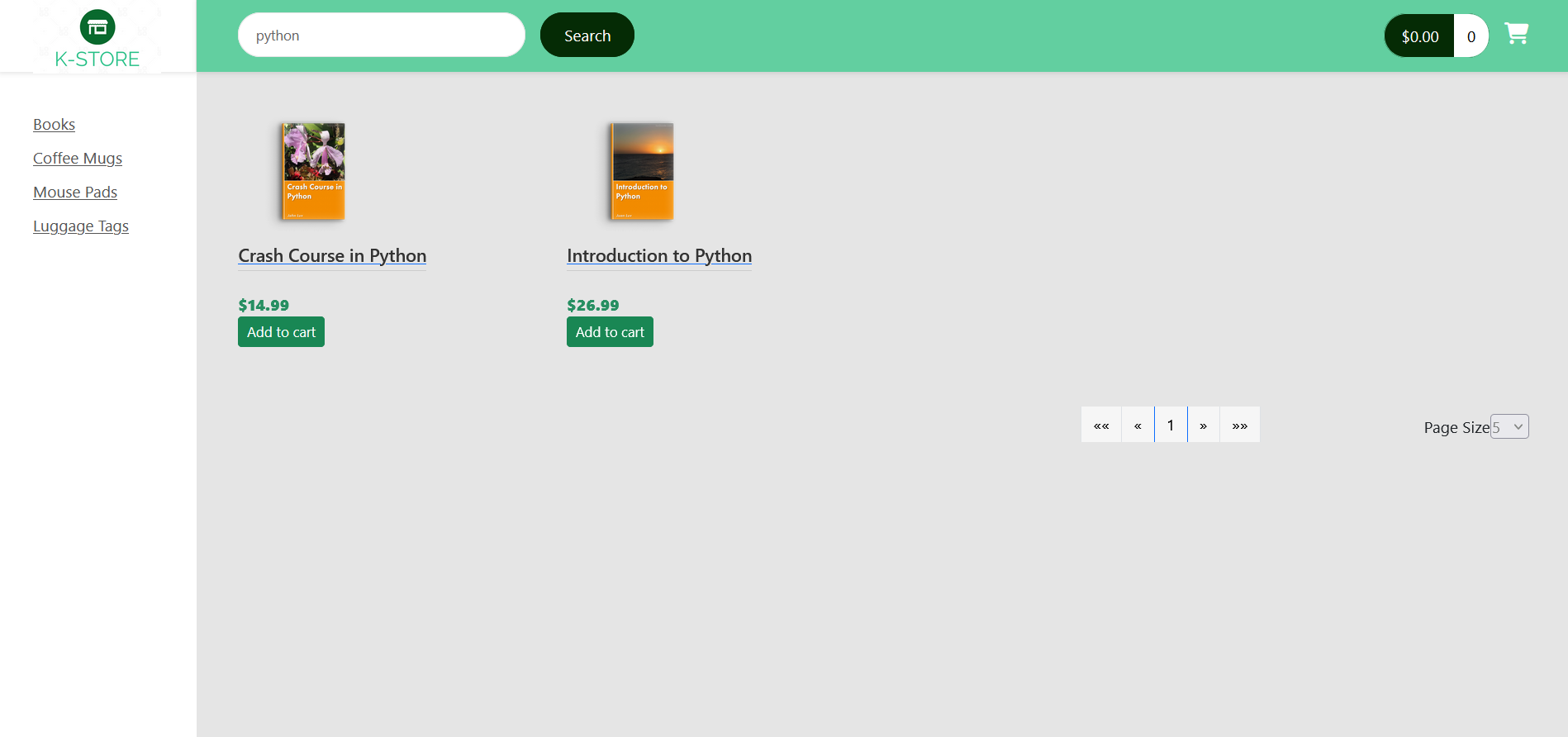
// Widok produktów z kategorii książki

Użytkownik wchodzący na stronę sklepu widzi stronę z dostępnymi produktami. Są to produkty pierwszej kategorii książki, po lewej widzimy możliwość zmiany kategorii wyszukiwania



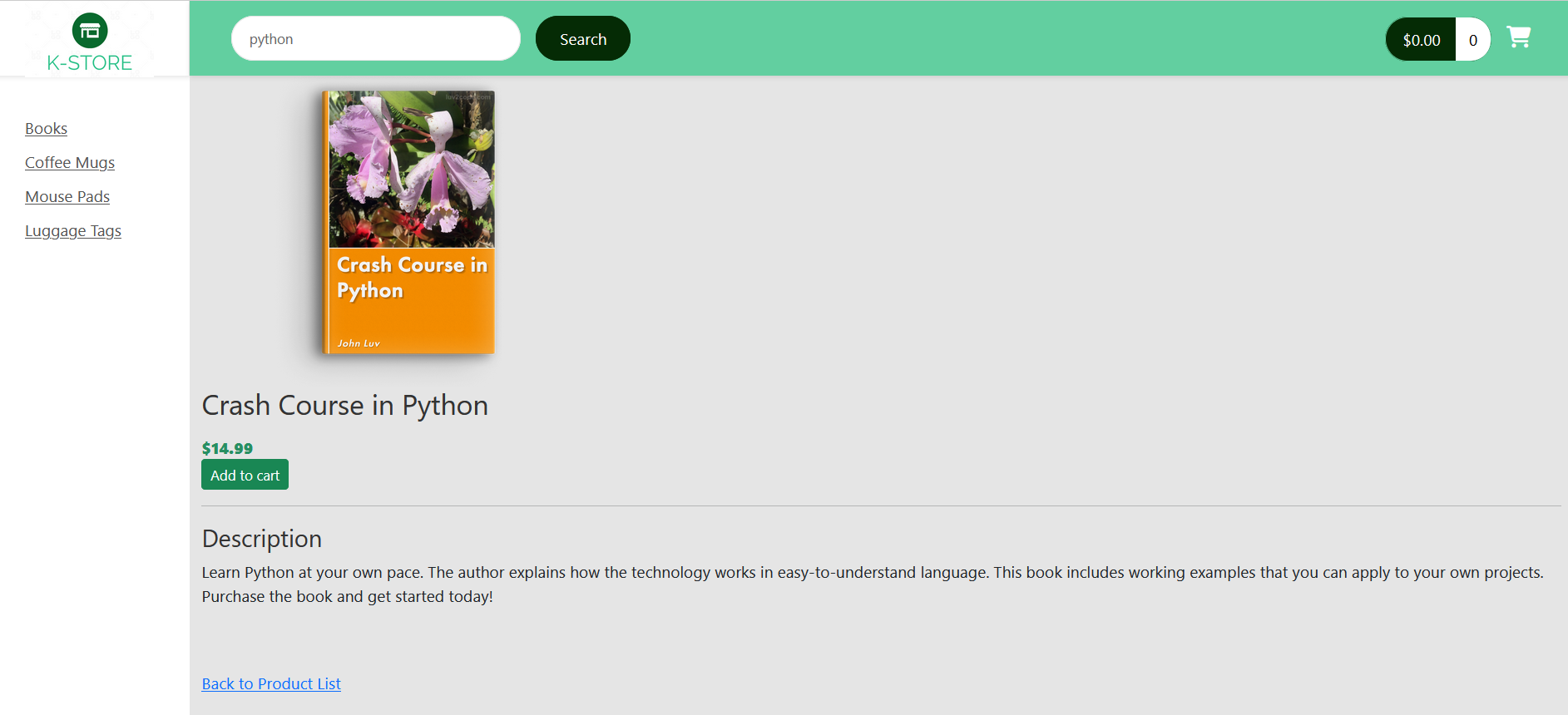
// Widok produktów z kategorii podkładki na myszki

Użytkownik może również wyszukać interesujący go produkt po nazwie. Załóżmy więc że chce on kupić książkę o języku Python. Wpisuje więc odpowiednią frazę w okno wyszukiwania i otrzymuję listę produktów. Jeżeli w sklepie nie ma produktów pasujących do frazy wyszukiwania użytkownik zostanie o tym poinformowany poprzez odpowiedni komunikat.



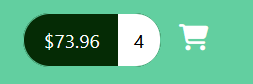
// Wyszukiwanie produktów po nazwie

Jeżeli użytkownik chce zobaczyć więcej szczegółów dotyczących produktu wystarczy że kliknie w jego nazwę lub zdjęcie. Wtedy ukaże mu się widok szczegółów produktu.



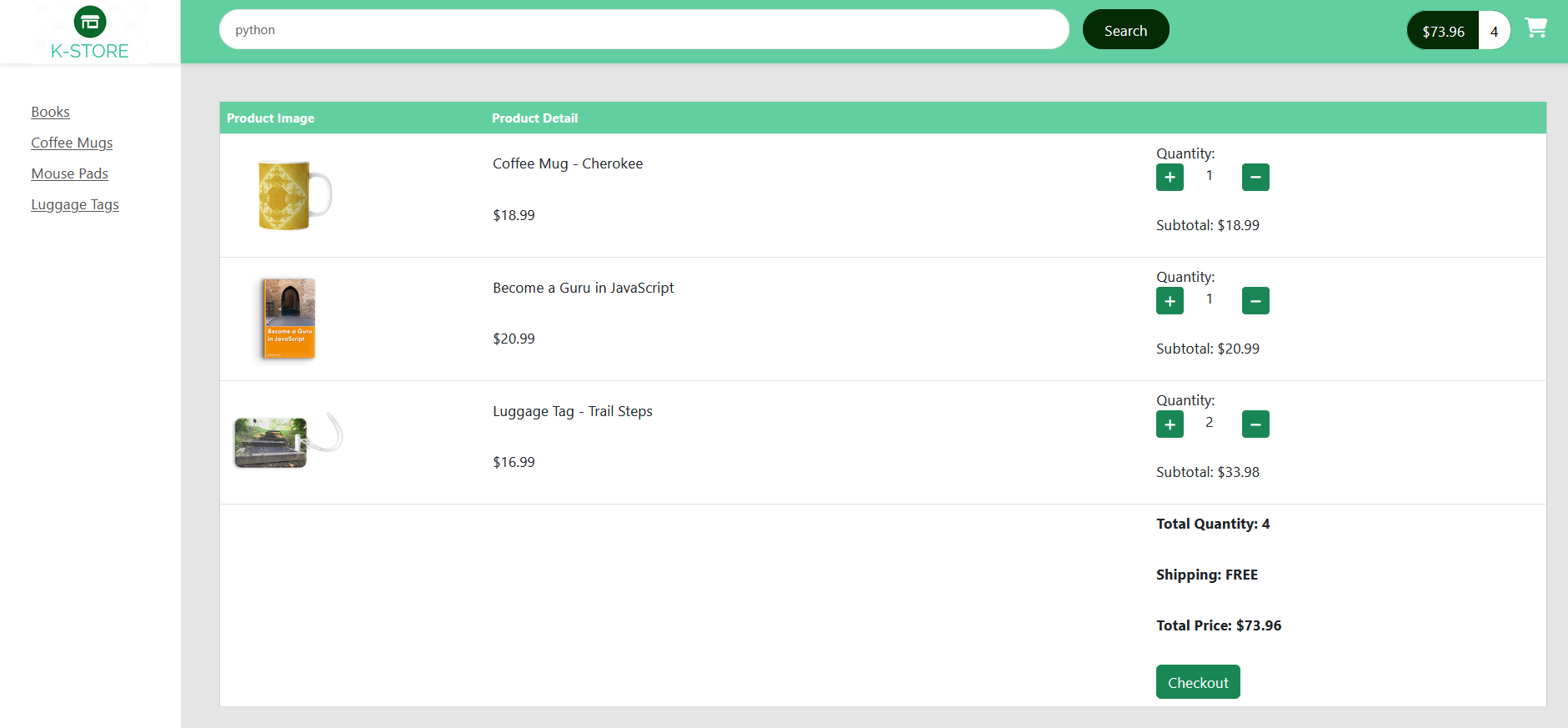
// widok szczegółów produktu

Użytkownik zarówno z widoku szczegółów produktu, jak i z widoku przeglądania może dodać produkt do koszyka. Informacje o liczbie produktów w koszyku i aktualnej cenie za wszystkie wybrane produkty jest aktualizowana i widoczna w prawym górnym rogu widoku.



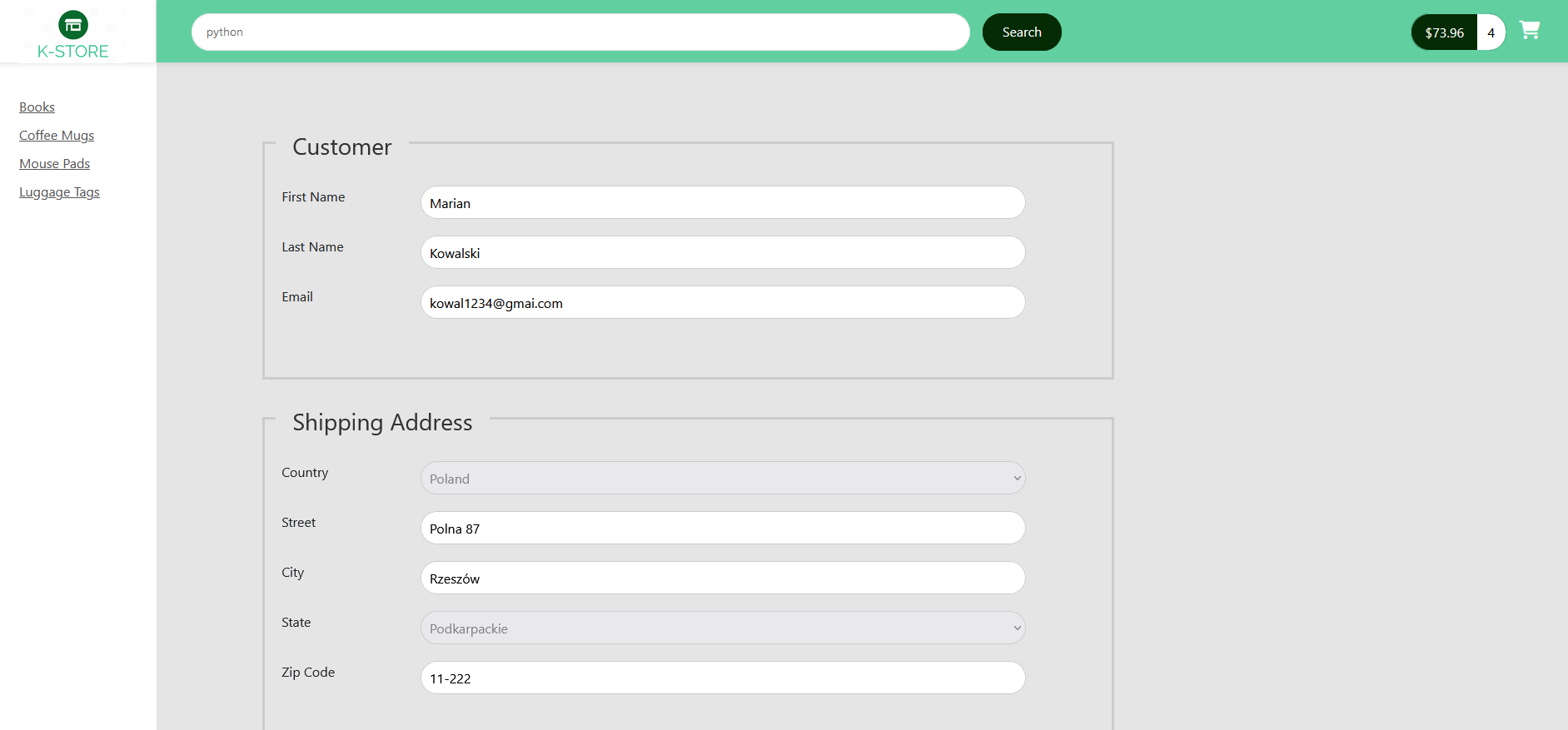
// informacje o produktach w koszyku

Jeżeli użytkownik chce już przejść do finalizacji zakupów to klika w ikonkę koszyka, lub w informacje o produktach w koszyku i aplikacja przeniesie go do kolejnego widoku. W tym widoku użytkownik widzi wszystkie wybrane produkty, a korzystając z dostępnych przycisków może jeszcze zmienić ich ilość lub całkowicie usunąć z koszyka.



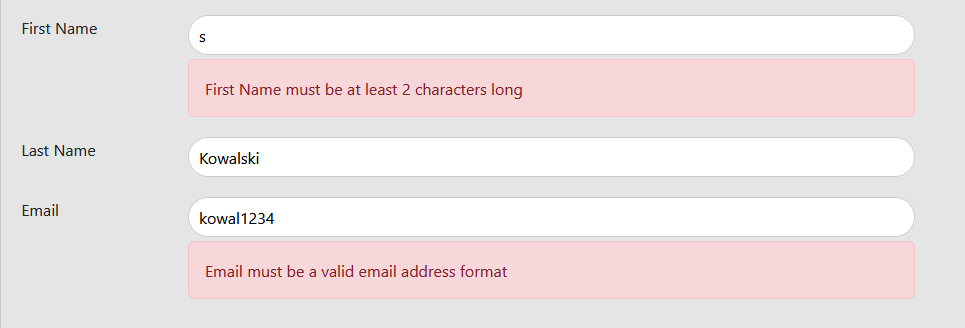
// widok koszyka z produktami

Jeżeli użytkownik jest już zdecydowany co do ilości produktów może kliknąć zielony przycisk „Checkout” i przejść do podsumowania zakupów. Wtedy aplikacja przeniesie użytkownika do kolejnego widoku. W tym widoku użytkownik wprowadza dane dotyczące osoby kupującej, adresu dostawy, adresu płatności i danych do płatności.



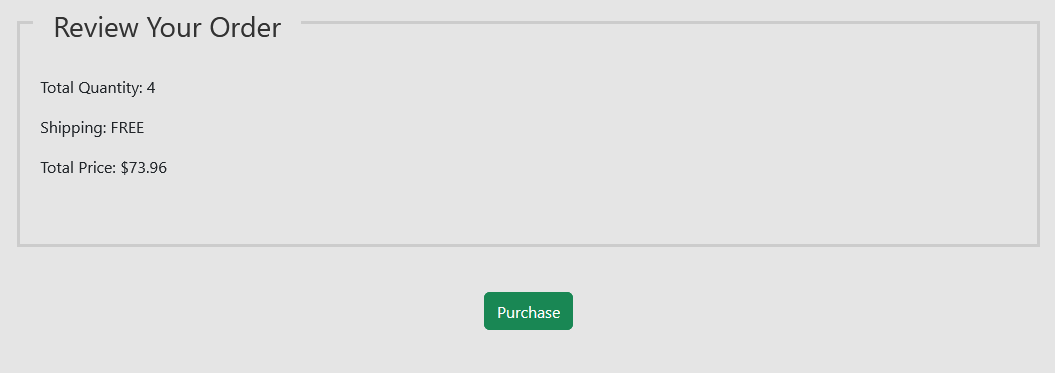
// Wprowadzanie danych kupującego

Kiedy aplikacja wykryje brak danych w miejscu, gdzie są one konieczne, albo błędne dane nie pozwoli ona na zarejestrowanie zakupu, a użytkownik zostanie poinformowany o błędzie i otrzyma wskazówki do jego naprawy.



// Informacje o nieprawidłowo wprowadzonych danych

Jeżeli użytkownik wprowadzi wszystkie dane poprawnie to aplikacja po kliknięciu przycisku „Purchase” zapisze informacje o zamówieniu w bazie danych.



// podsumowanie zamówienia

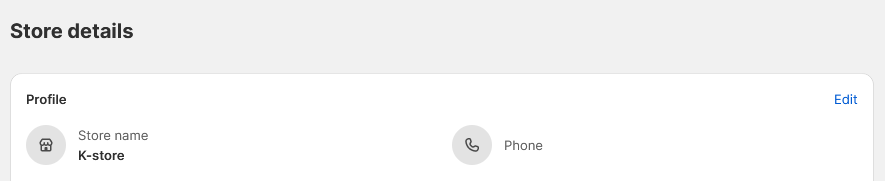
Po zarejestrowaniu zamówienia w bazie użytkownik otrzyma powiadomienie że wszystko przebiegło pomyślnie i zamówienie zostało złożone.

## 3.2. Stworzenie sklepu internetowego z wykorzystaniem gotowego rozwiązania

W tym podrozdziale skupię się na przedstawieniu tworzenia sklepu internetowego, przy użyciu platformy Shopify. Proces tworzenia sklpeu jest intuicyjny, a w całości prac pomagał mi film instruktażowy stworzony przez osobę z dużym doświadczeniem w tworzeniu sklpepó na tej platformie [7]

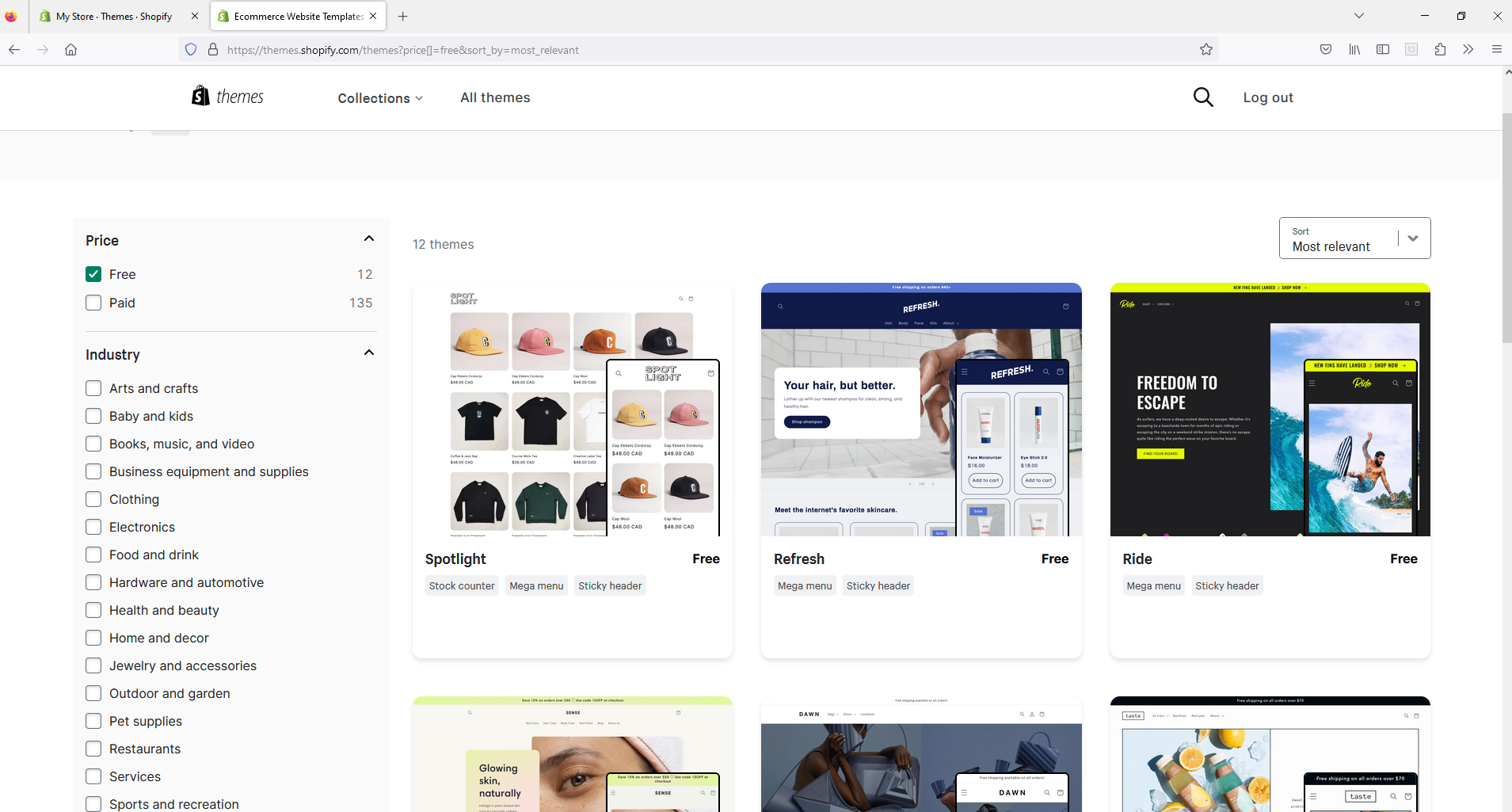
Sklep stworzyłem z wykorzystaniem darmowego okresu próbnego. Platforma shopify daje możliwość przetestowania jej usług poprzez stworzenie próbnego, darmowego sklepu i sprawdzenie funkcjonalności sklepu.

Na oficjalnej stronie [8] założyłem konto i zacząłem okres próbny. Po wypełnieniu wstęnej ankiety platforma przeniosła mnie do edytora sklepu, a tam pierwszą rzeczą jaką zrobiłem by wybór nazwy sklepu. Wybrałem nazwe taką samą jak w przypadku sklepu opisywanego w poprzednim podrozdziale.



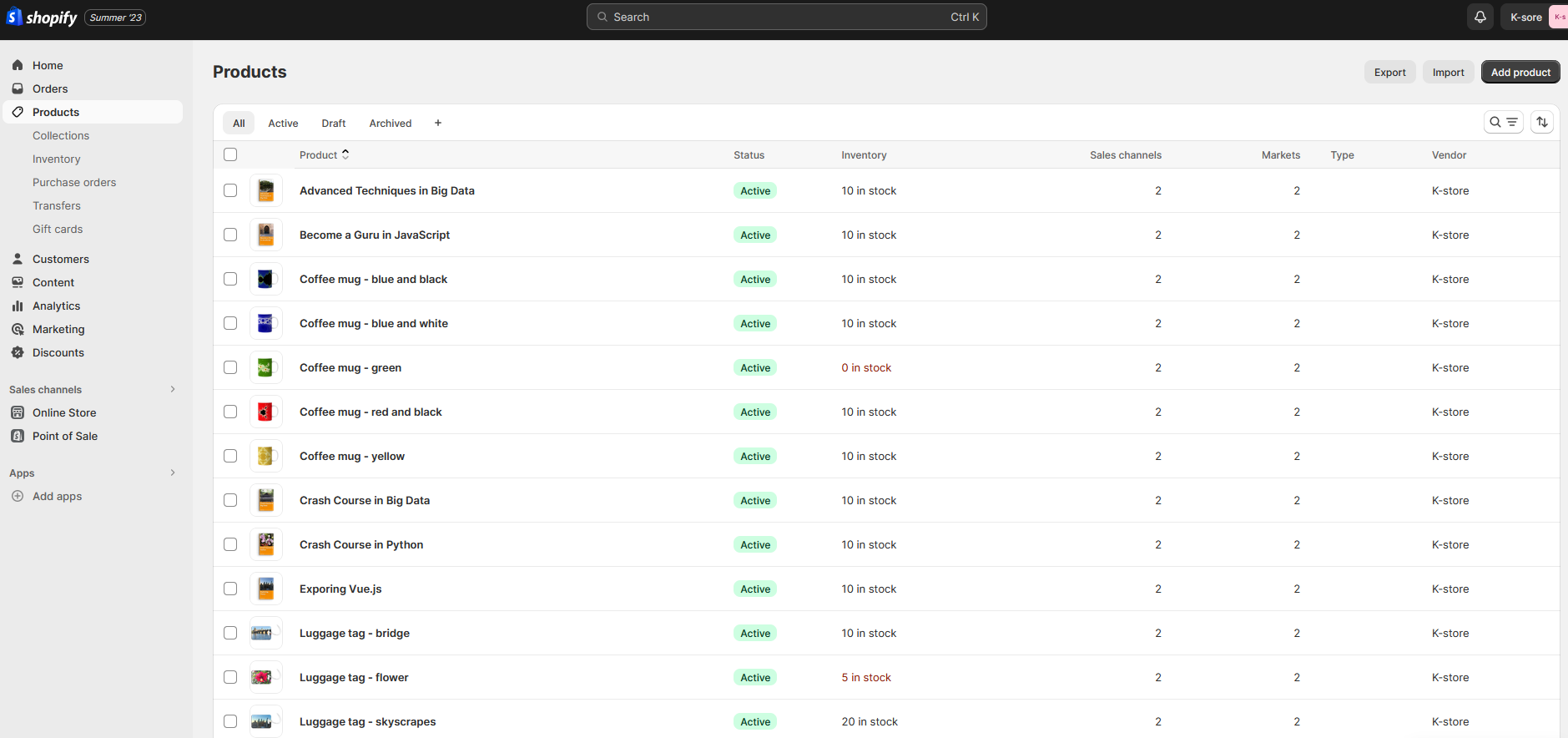
// Ustawianie nazwy sklepu

Kolejnym krokiem nył wybór szablonu. Shopify zapewnia szeroki wybór gotowych szablonów sklepów. Te szablony są projektowane z myślą o różnych branżach i stylach, więc warto przejrzeć dostępne opcje i wybrać ten, który najlepiej pasuje do marki. Możesz filtrować szablony według kategorii, popularności, czy funkcji, które oferują. W momencie tworzenia sklepu platforma oferowała 12 bezpłatnych i 135 płatnych szablonów.



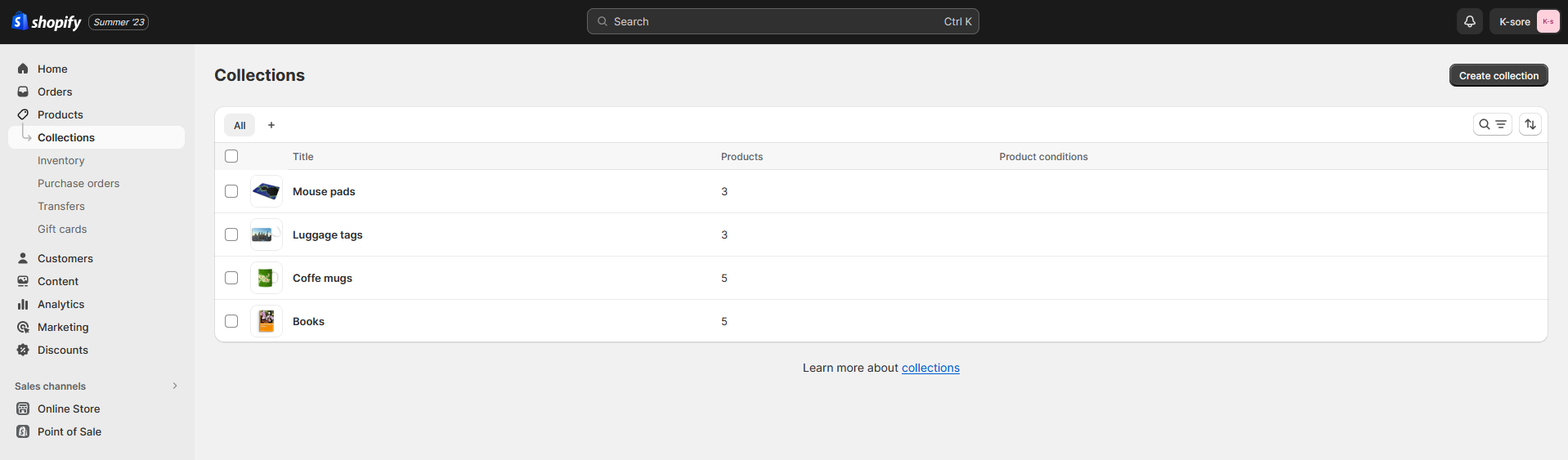
// Okno wyboru szablonów

Po wybraniu szablonu przeszedłem do dodawania produktów do sklepu. Proces dodawania produktów do sklepu jest prosty i intuicyjny. W panelu administracyjnym w sekcji "Products" jest widoczny przycisk "Add product". Po kliknięciu przycisku należy wprowadzić nazwę produktu, opis, cenę oraz dostępność w magazynie. Możesz również dodać zdjęcia produktu. Shopify pozwala na zarządzanie wieloma aspektami produktów, takimi jak różne warianty (np. rozmiary, kolory), co ułatwia klientom znalezienie odpowiednich produktów.



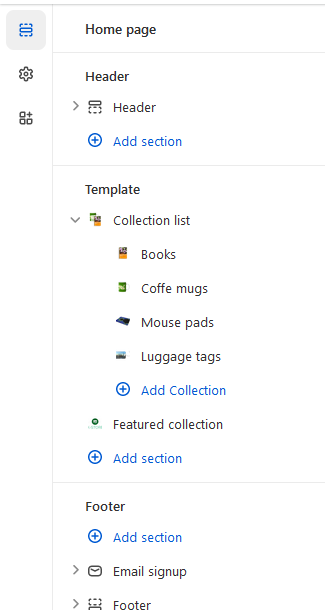
// dodane produkty w sekcji Products

Po dodaniu produktów do mojego sklepu skorzystałem z opcji tworzenie kolekcji. Podobnie jak w aplikacji Spring Boot, Angular stworzyłem cztery kategorie produktów i umieściłem w nich ręcznie dodane wcześniej produkty.



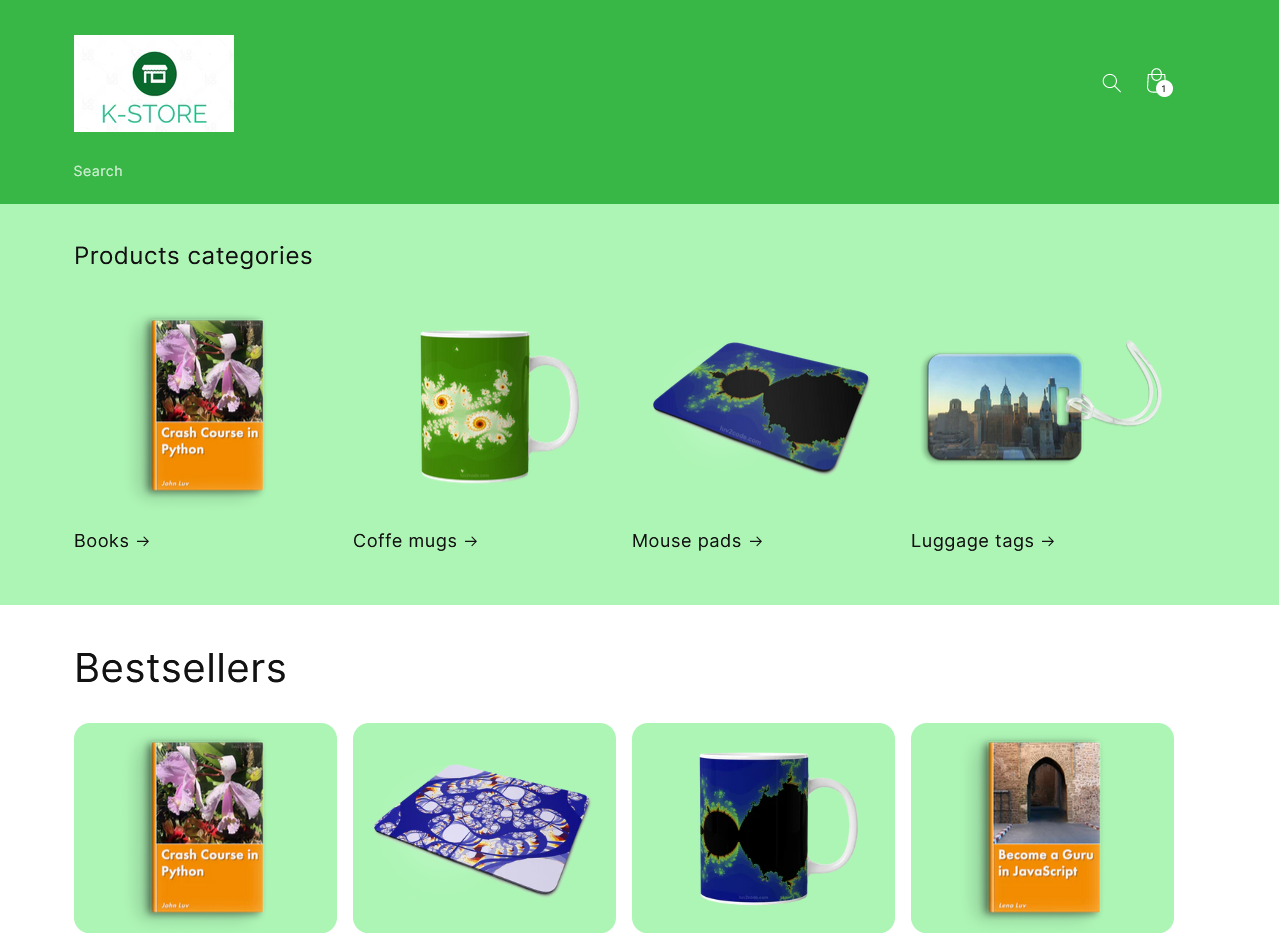
// kategorie produktów

Po wybraniu szablonu i dodaniu produktów przeszedłem do personalizowania wyglądu sklepu . W panelu administracyjnym Shopify jest dostęp do narzędzi, które pozwalają na łatwe dostosowanie wyglądu sklepu. Można dodać swoje logo i zmienić kolorystykę tak, aby pasowała do identyfikacji wizualnej marki. Dodatkowo, można wybrać odpowiednie czcionki, które będą stosowane na stronie. Dzięki funkcji "Drag-and-drop" (przeciągnij i upuść) można łatwo zmieniać układ strony, dodawać sekcje i edytować ich zawartość. Sekcje te mogą przedstawiać różne kolekcje, pojedyncze produkty, udzielać informacji o firme, albo przekierowywać na zewnętrzne strony. W tworzonym sklepie starałem się utworzyć rozwiązanie podobne do utworzonej wcześniej aplikacji ecommerce Spring Boot, Angular.

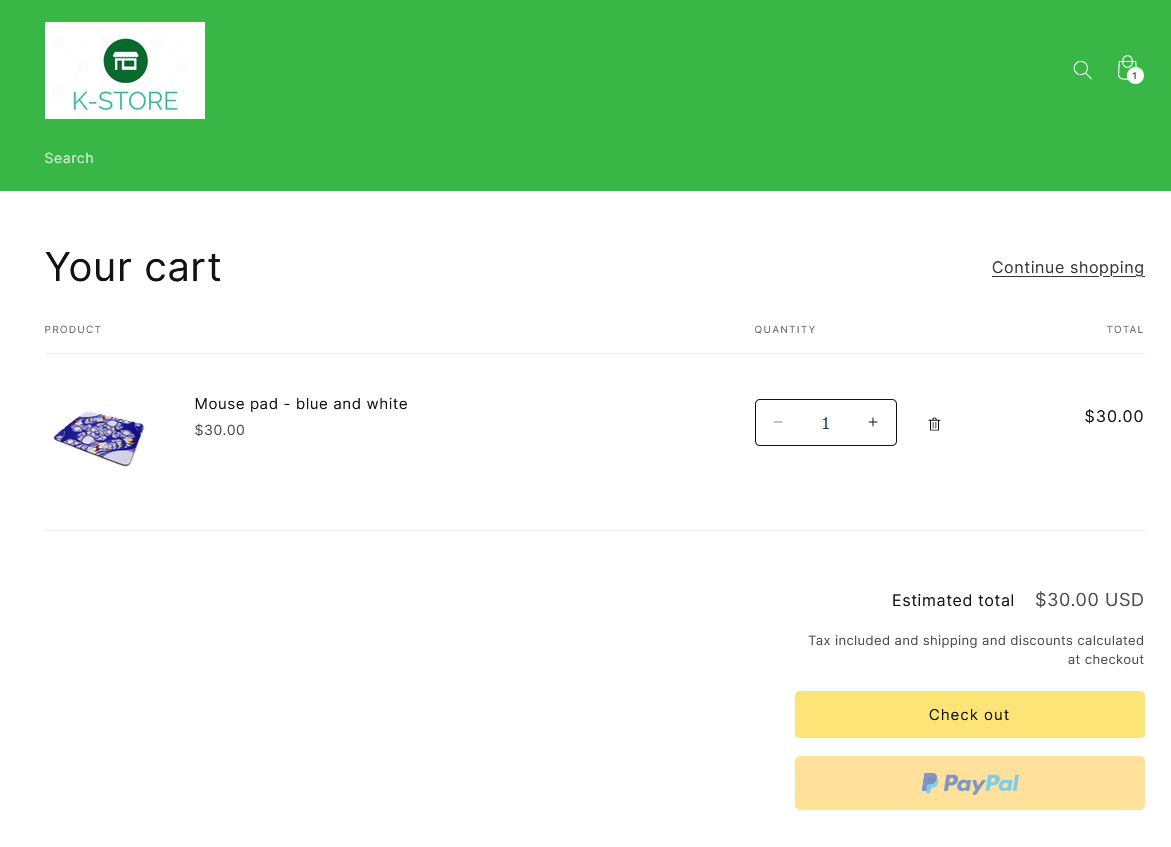


// Menu dostępne w sekcji personalizacji sklepu

Ostatecznie w sklepie utworzonym na platformie Shopify można wybrać jedną z czterech kategorii produktów, zobaczyć szczegóły każdego z produktów w sklepie, wyszukać produkt po nazwie i dodać go do koszyka. W koszyku można zmienić liczbę umieszczonych w nich produktów, albo przejść do sekcji podsumowania zamówienia i płatności w której kupujący podaje dane niezbędne do złożenia zamówienia.



// Widok wyszukiwania produktów



// widok koszyka

# Porównanie obu podejść

Asdasd

- dostęp do bazy danych rozwiązania, podglądanie danych przez zapytania SQL

## 4.1. Koszty

Adasd

## 4.2. Elastyczność

Qwdwqd

## 4.4. Liczba korzystających

Adsad

xcxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

## 4.5. Możliwość zmian

Zmiany w wyglądzie – html

Dodanie funkcjonalności niestandardowej

4.6. Pomoc w biznesie

<https://www.youtube.com/watch?v=6Da36Z9gNuU>

Jakieś tam policy, ustawianie rybku na dany kraj, dostawa, marketing też daje dużo

### 4.7 Koszty prowadzenia

4.8 dostęp do bazy danych

4.9. utrata danych po zakończeniu współpracy

4.10. koszty i czas tworzenia sklepu

4.11. aspkety ważne dla klienta (wygląd, wygoda korzystania)

4.12. płatności o dsotawy

4.13. Wprowadzeni niestandardowych funkcjonalności

4.14. wsparcie rozwiązania

4.15 grono użytkowników

Itp…

# Metody marketingowe stosowane w sklepach internetowych

Adsasda

<https://www.youtube.com/watch?v=vgz1zeNc7kU&t=164s>

<https://thenewlook.pl/jak-zalozyc-sklep-internetowy/>

1. Podsumowanie

Opis wyników porównań

# Zakończenie

Adasdasd

# Literatura

[1] <https://thenewlook.pl/e-commerce-co-to-jest/>

[2] <https://javastart.pl/baza-wiedzy/wprowadzenie/historia-javy>

[3] <https://angular.io/docs>

[4] <https://www.codecademy.com/article/what-is-rest>

[5] <https://thenewlook.pl/platformy-saas/>

[6] <https://www.udemy.com/course/full-stack-angular-spring-boot-tutorial/>

[7] <https://www.youtube.com/watch?v=6Da36Z9gNuU>

[8] <https://www.shopify.com/>

**Sekcja TO DO**

- wymyśl co można porównać

- Opisz porównania od strony shopify najpierw ( bo masz 3 dni trial)

- opisz porównania od strony sklepu tworzonego ręcznie

- podsumuj swoje porównania w osobny rozdziale podsumowania

- przerób wygląd dokumentu na WSiZowy

- Dopisz wstęp i zakończenie (chyba że podsumowanie i zakończenei w jendym)

- Dodaj książki z google scholar do dokumentacji

//https://scholar.google.com/

LUKNIJ NA TO !!

<https://bluemedia.pl/baza-wiedzy/blog/ecommerce/najpopularniejsze-platformy-ecommerce-w-polsce>

<https://www.youtube.com/watch?v=vgz1zeNc7kU&t=164s>

<https://thenewlook.pl/jak-zalozyc-sklep-internetowy/>