

WYDZIAŁ FINANSÓW I BANKOWOŚCI

Aplikacja do zarządzania magazynem

PROJEKT DYPLOMOWY

Poznań 2022

DANE PARTNERÓW

A1. Dane Promotora

Imię i nazwisko	Mariusz Nogala
Stopień / Tytuł naukowy	Doktor
Data i podpis	

A2. Dane członków Zespołu projektu

Imię i nazwisko	Bartosz Gwiazdowski
Kierunek studiów	Informatyka
Tryb studiów	Niestacjonarny
Data i podpis	

Imię i nazwisko	Rafał Graś
Kierunek studiów	Informatyka
Tryb studiów	Niestacjonarny
Podpis	

Imię i nazwisko	Adrian Bąk
Kierunek studiów	Informatyka
Tryb studiów	Niestacjonarny
Data i podpis	

Spis treści

ZAŁOŻENIA PROJEKTU.....	4
B1. Opis projektu.....	4
B2. Zadania w projekcie.....	7
REALIZACJA	8
C1. Opracowanie projektu	8
C2. Efekty realizacji projektu	17
C3. Użyteczność projektu.....	18
C4. Autoewaluacja zespołu projektowego	19
C5. Wykorzystane materiały i bibliografia związana z realizacją projektu	21
C6. Spis załączników.....	21

ZAŁOŻENIA PROJEKTU

B1. Opis projektu

1. Uzasadnienie wyboru tematu

Wraz z rozwojem przedsiębiorstwa, wzrostem produkcji rosną stany magazynowe produkowanych dóbr, wraz z tym wielkie wyzwania czekają dział logistyki. Magazynem należy zarządzać w taki sposób by realizował wszystkie zamówienia, potrzebne w danym momencie i by przynosił największy możliwy zysk, generując jak najmniejsze koszty. Kluczem do dalszego rozwoju w tej dziedzinie jest sprawne zarządzanie procesem dystrybucji, jak i poszerzanie asortymentu. Sam proces jest skomplikowany i wymaga od przedsiębiorców przemyślanych działań, wielkie wsparcie w tej kwestii może zapewnić aplikacja, wspierająca firmę w procesie zarządzania magazynem. Zautomatyzowanie magazynu pozwoli efektywniej nim zarządzać, szybciej wykonywać zadania, usprawnić proces dystrybucji oraz zminimalizować ryzyko błędów, z których bezpośrednio mogą wynikać straty. Rozwój technologii pozwala już, aby magazyny mogły być w większym stopniu zarządzane z pomocą komputerów. Duże przedsiębiorstwa korzystają w tych celach najczęściej z systemów ERP, a małe firmy zostają w tej kwestii w tyle i nie chcą wydawać dużych sum na wprowadzenie systemu o zakresie jakiego nie potrzebują i w celu zarządzania magazynem korzystają najczęściej z oprogramowania, które nie jest stworzone w tym celu. Powstanie aplikacji dedykowanej, taniej, nieskomplikowanej, prostej w obsłudze i stworzonej z myślą wsparcia małych przedsiębiorstw w codziennych zadaniach związanych z zarządzaniem magazynem i łańcuchem dostaw może rozpocząć proces informatyzacji firmy i otworzenia się jej na korzystanie ze wsparcia jakie dedykowane aplikacje mogą zaoferować i przyczynić się bezpośrednio do dalszego rozwoju przedsiębiorstwa, nie ponosząc przy tym zbyt dużych kosztów.

2. Problem badawczy

Główny problem badawczy przyjął postać następującego pytania „Jak skutecznie usprawnić pracę magazynu i zredukować czas, koszty i ryzyko pomyłki przy jego

zarządzaniu”. W pewnym momencie rozwoju przedsiębiorstwa, takie pytania zaczynają przybierać coraz większy priorytet, ponieważ od tego zależy później maksymalizacja dochodów i zdolność do ciągłego powiększania bazy klientów, nie tracąc przy tym tych dotycznych. Produkt musi być dostarczony w odpowiednim czasie oraz całość tego procesu należy dokładnie dokumentować, w przypadku zautomatyzowania magazynu, zmniejszamy ryzyko błędu ludzkiego do minimalnego poziomu, co niesie ze sobą redukcję kosztów i znacząco zwiększa wydajność magazynu. W przypadku zarządzania wieloma magazynami błęd w procesie przenosin towaru miedzy poszczególnymi magazynami może okazać się tragiczny w skutkach i nieść ze sobą ogromne koszty. Zwłaszcza w takiej sytuacji, od stopnia zautomatyzowania zależy wydajność magazynu, co wpływa na pozycję samego przedsiębiorstwa, gdyż dopiero gdy magazyn funkcjonuje sprawnie, firma zyskuje możliwość dalszego, szybszego rozwoju.

3. Cel główny i cele szczegółowe projektu

Za główny cel projektu postawiono znalezienie sposobu na sprawniejsze i jak najbardziej zautomatyzowane zarządzanie magazynem.

By zrealizować wyżej postawiony cel główny wykonano poniższe zadania i cele szczegółowe:

- stworzono aplikację wspierającą proces zarządzania magazynem i automatyzującą go w pewnym stopniu;
- wykonano badania polegające na przeprowadzeniu wywiadu w dziale logistyki w firmie UTAL;
- przeprowadzono wywiady z osobami odpowiedzialnymi za funkcjonowanie magazynu w małych i średnich przedsiębiorstwach;
- przeanalizowano wzory procesów logistycznych w przedsiębiorstwach;
- zbadano modele i struktury budowania magazynu oraz jego podział, z naciskiem na te, na podstawie których, najczęściej tworzy się magazyn;
- przeanalizowano literaturę poruszającą dany problem;
- przeprowadzono wywiad ze studentami WSL;
- wykonano analizę przeprowadzonych badań i informacji.

4. Zakres podmiotowy, przedmiotowy, czasowy i przestrzenny

Zakres podmiotowy:

- firma „UTAL”, mająca swoją siedzibę w Gruszzynie, w województwie wielkopolskim, w powiecie poznańskim, na terenie gminy Swarzędz i osoby zarządzające magazynem i zajmujące się jego funkcjonowaniem;
- osoby odpowiedzialne za procesy magazynowe w małych i średnich przedsiębiorstwach na terenie województwa wielkopolskiego tj. magazynierzy, logistycy, spedytorzy;
- studenci WSL w Poznaniu.

Zakres przedmiotowy:

- struktura magazynu i procesy przy jego zarządzaniu i funkcjonowaniu w firmie „UTAL”;
- procesy i procedury zachodzące podczas funkcjonowania magazynu w małych i średnich przedsiębiorstwach;
- problemy i trudności w procesach magazynowych;
- wyniki analizy modelów magazynowych;

Zakres czasowy:

- projekt powstał od marca 2022 do stycznia 2023 roku.

Zakres przestrzenny:

- firma „UTAL” znajdująca się w miejscowości Gruszczyn, w województwie wielkopolskim, w powiecie poznańskim, na terenie gminy Swarzędz;
- małe i średnie przedsiębiorstwa na terenie województwa wielkopolskiego.

5. Metody i techniki badawcze:

- obserwacja;
- analiza;
- wywiad;

B2. Zadania w projekcie

Cele szczegółowe projektu	Zadania w projekcie oraz termin rozpoczęcia i zakończenia realizacji zadania	Osoby zaangażowane w realizację zadania
Cel 1: Stworzenie I rozwój aplikacji do zarządzania magazynem	Zadanie 1: 08.2022 – 08.2022 Napisanie założeń, wybór narzędzi i technologii	1. Rafał Graś 2. Adrian Bąk 3. Bartosz Gwiazdowski
	Zadanie 2: 08.2022 – 09.2022 Stworzenie bazy danych	1. Adrian Bąk
	Zadanie 3: 08.2022-12.2022 Stworzenie aplikacji wg wcześniej założonych planów i technologii	1. Adrian Bąk 2. Rafał Graś 3. 4.
<hr/>		
Cel 2: Przeprowadzanie wywiadu w dziale logistyki firmy “UTAL” oraz w małych i średnich	Zadanie 1: 11.2022 Przeprowadzenie wywiadu w dziale logistyki w firmie “UTAL”	1. Bartosz Gwiazdowski 2. 3. 4.



przedsiębiorstwach, i rozmowa ze studentami WSL w Poznaniu.	Zadanie 2: 11.2022 Przeprowadzenie wywiadów w małych i średnich przedsiębiorstwach w województwie wielkopolskim	1. Bartosz Gwiazdowski 2. 3. 4.
	Zadanie 3: 11.2022 Przeprowadzenie wywiadu ze studentami WSL w Poznaniu	1. Bartosz Gwiazdowski 2. 3. 4.
Cel 3: Przeprowadzenie analizy zebranych danych oraz literatury	Zadanie 1: 05.2022 – 12.2022 Zebranie i analiza literatury dotyczącej tematu pracy	1. Bartosz Gwiazdowski 2. Rafał Graś 3. Adrian Bąk
	Zadanie 2: 05.2022 – 12.2022 Analiza zebranych danych	1. Bartosz Gwiazdowski 2. Rafał Graś 3. Adrian Bąk

REALIZACJA

C1. Opracowanie projektu

1. Założenia teoretyczne

Sprawnie funkcjonujący magazyn jest podstawą dobrze działającego przedsiębiorstwa, zwłaszcza gdy zajmuje się ono produkcją. Podział magazynów oraz jego zawartości powinien być dla pracownika jasny, przejrzysty i intuicyjny. Planując miejsca, które będą przeznaczone do składowania zapasów, należy być świadomym tego, że od nich będzie

zależeć bezpieczeństwo towaru oraz efektywność w procesie magazynowania, przyjęcia, składowania, przechowywania oraz wydania produktu. Planując całe przedsięwzięcie, należy zacząć od zaznajomienia się z gotowymi już modelami oraz podziałem magazynów, w zależności od branych pod uwagę kryteriów.

W przypadku podziału w oparciu na typ oraz konstrukcję obiektu, wyszczególnia się magazyny:

- otwarte, które będą odpowiednie, przy składowaniu towaru odpornego, przede wszystkim, na warunki atmosferyczne, mogą mieć one postać placu, bądź zbiornika. Do ich najważniejszych zalet należą koszty oraz prostota, są one zazwyczaj najtańsze w eksploatacji, a poziom skomplikowania przy jego powstawaniu jest bardzo mały;
- półotwarte, są podobne do magazynów otwartych, lecz posiadają najczęściej zadaszenie, na przykład wiaty. Mogą być przeznaczone do składowania towaru, który ma już nieco mniejszą odporność na deszcz, czy słońce, na przykład surowa stal;
- zamknięte, które są przeznaczone do składowania produktów, dla których przechowywanie na otwartej przestrzeni może okazać się zbyt szkodliwe, takich jak chociażby sprzęty elektroniczne, artykuły RTV i AGD. Najczęściej przybierają postać hali magazynowej, są najdroższe i najbardziej skomplikowane w procesie powstawania, lecz są najbezpieczniejsze dla towaru, który jest zabezpieczony nie tylko przed warunkami atmosferycznymi, ale także przed chociażby kradzieżą. Magazyny otwarte również dzielimy, a kryterium według którego dokonywany jest wysokość. Są to magazyny:
 - niskiego składowania – do 4.2m;
 - średniego składowania – od 4.2m do 7.2m;
 - wysokiego składowania – od 7.2m.

Magazyny dzieli się także ze względu rodzaju produktów w nich przechowywanych, są to magazyny:

- składające jednostki ładunkowe, towar w nich jest składowany głównie na paletach, w pojemnikach, czy kartonach. Produkty, które składujemy w nich to chociażby artykuły RTV i AGD;

- składające towary sypkie, najczęściej oddzielone od siebie, leżące luzem w specjalnych zbiornikach, na przykład silosy w przypadku zboża;
- składające gazy i ciecze, magazynowane w specjalistycznych już zbiornikach, przeznaczonych i wyprodukowanych w tym właśnie celu, jak na przykład butla na propan butan;
- specjalne, które są przeznaczone do składowania materiałów niebezpiecznych, bądź wymagających określonych warunków przy przechowywaniu, są to na przykład zbiorniki na materiały łatwopalne, czy zbiorniki ciśnieniowe.

W przypadku gdy kryterium podziału to pełiona funkcja bądź miejsce składowania, które to kryteria rzadko są podzielane przy tworzeniu oprogramowania do zarządzania magazynem, to wyróżnić można magazyny:

- surowców, istniejące najczęściej w przedsiębiorstwach zajmujących się produkcją. Służą one do przechowania komponentów i towarów, z których powstają dobra, produkowane przez firmę;
- półfabrykatów, gdzie przechowywane są wyroby nieskończone, które są przeznaczone do dalszej obróbki, bądź sprzedaży;
- odpadów, najczęściej przeznaczonych do recyklingu, czasami do sprzedaży;
- gotowych wyrobów, gdzie najczęściej znajdują się zrobione już produkty przeznaczone do sprzedaży;
- opakowań, gdzie przechowywane są zapasy opakowań;
- techniczne, w których zazwyczaj składowane są urządzenia, bądź części zamienne;
- przeładunkowe, mające za zadanie usprawnić i skrócić łańcuch dostaw;
- produkcyjne, w których składa się towary i dobra, pomiędzy poszczególnymi etapami produkcji;
- handlowe, znajdujące się blisko punktu sprzedaży;
- dystrybucyjne, pełniące funkcje wysyłkowe i dostawcze, których działanie w swojej książce Ireneusz Fechner opisał następująco: „Przeładunek kompletacyjny (cross docking) polega na przemieszczaniu dostawy przez łańcuch dostaw bez składowania w ogniwach pośrednich pomiędzy dostawcą i odbiorcą. Przeładunek kompletacyjny może być stosowany w sytuacji, gdy pomiędzy dostawcą i jego odbiorcami występuje ogniwko pośrednie,

np. Magazyn dystrybucyjny. W rozwiązaniu klasycznym Magazyn dystrybucyjny jest pośrednikiem gromadzącym zapasy od dostawcy i przekazującym je odbiorcom skompletowane zgodnie ze składanymi przez nich zamówieniami”.

Ostatnim kryterium, według którego często dzielimy magazyny, jest stopień ich zautomatyzowania, który często zależy od ich wielkości i poziomu rozwoju przedsiębiorstwa. Wyróżniamy magazyny:

- niezmechanizowane, w których wszystkie procesy odbywają siłą ludzkich rąk, większość operacji odbywa się przy pełnym udziale człowieka;
- zmechanizowane, w przypadku których większość działań i procesów, podobnie jak przy niezmechanizowanych odbywa się przy znacznym udziale człowieka, lecz który jest wspierany przez urządzenia i maszyny, na przykład wózki widłowe;
- zautomatyzowane, w których większość, bądź wszystkie operacje odbywają się automatycznie, bez ingerencji człowieka.

W obecnych czasach przedsiębiorcy zdają sobie sprawę, z tego jak ważne jest efektywne zarządzanie magazynem. Sprawne zarządzanie logistyką jest często kluczowe w konkurowaniu z najlepszymi. Przechowywanie towaru oraz produkowanych dóbr to przede wszystkim koszty. Rozwój magazynu i procesów logistycznych polega na redukcji tych kosztów, wszystkie rozwiązania, rozwój technologiczny, nowe rozwiązania na magazynie, mają przede wszystkim jeden cel, redukcję kosztów jakie ten magazyn generuje. Kiedy magazyn będzie działał szybciej, koszty jego pracy i utrzymania zmaleją, kiedy na magazynie zmniejszy się zapotrzebowanie na ludzi, koszty zmaleją, im bardziej automatyczny będzie magazyn, tym mniej odczuwalne będą trudności związane z jego pracą, jak i koszty, które będzie generował. Właśnie dlatego największe firmy dążą do jego ciągłego rozwoju i maksymalnego zautomatyzowania. Bez dobrze rozwiniętych procesów magazynowych, rozwój firmy zacznie w pewnym momencie spowalniać, a na pewnym etapie całkowicie się zatrzyma.

Zautomatyzowany magazyn wysokiego składowania jest tym do czego dążą najbardziej rozwinięte przedsiębiorstwa, wskazuje on na wysokie możliwości produkcyjne oraz

finansowe przedsiębiorstwa. Magazyn może wiele mówić o stopniu zaawansowania technicznego i technologicznego przedsiębiorstwa, moment w którym magazyn zaczyna wymagać automatyzacji, to również chwila, w której w dział logistyki, zaczyna ingerować dział IT.

Wcześniej wspomniane połączenie kryteriów miejsca składowania oraz pełnionej funkcji, w kwestii podziału magazynów, wynika w sporej, o ile nie w całkowitej mierze, z powodu ingerencji informatyki i technologii komputerowej w procesy logistyczne. W wielu systemach ERP, jak i oddzielnych aplikacjach do zarządzania magazynem, jest to jednym kryterium, które znacznie ułatwia posługiwanie się systemem oraz sprawia, że jest on bardziej intuicyjny dla osób, które będą się nim posługiwać. Systemy te dodatkowo często tworzone są w taki sposób, by osoby odpowiedzialne za jego administrowanie, a nie biorące udziału w jego tworzeniu, były w stanie edytować jego funkcjonalności i dostosować go do potrzeb przedsiębiorstwa. Zwłaszcza w systemach ERP, sekcja odpowiedzialna za magazyn jest dostosowana do zmieniania jej funkcjonalności, tak by dostosować ją do tego, jak funkcjonuje przedsiębiorstwo.

Obecnie ciężko sobie wyobrazić funkcjonowanie, zwłaszcza dużych przedsiębiorstw, bez znaczącego udziału komputerów i systemów wspomagających działanie firmy, ich udział jest nieoceniony niemalże w każdej dziedzinie życia przedsiębiorstwa. W kwestiach zarządzania gospodarką magazynową najważniejszym aspektem jest skrócenie czasu procesów, zachodzących podczas działania magazynu, jak i zmniejszenie liczby pracowników odpowiedzialnych za jego funkcjonowanie. W obu tych kwestiach najważniejszym skutkiem jest zmniejszenie kosztów. Systemy do zarządzania magazynem pozwalają na to, poprzez takie funkcje jak:

- generowanie faktur, jak i innych dokumentów, takich jak na przykład WZ, PZ;
- generowanie raportów na podstawie danych w systemie;
- zapisywanie w historii wszystkich dokumentów i procesów;
- możliwość zmian w stanach magazynowych;
- ustalenie stanów minimalnych danego towaru;
- szybkie wyszukiwanie i filtrowanie dóbr, gotowych produktów, półfabrykatów, materiałów, jak i dokumentów;

- pomoc w inwentaryzacji;
- gotowe szablony dokumentów;
- lokalizacje poszczególnych towarów;
- szybkie przyjmowanie i wysyłka towaru.

To tylko jedne z niewielu udogodnień, które może dostarczyć dobrze działający system magazynowy. Systemy te dodatkowo wymiernie wpływają na poprawę jakości całego procesu logistycznego, na którego ocenę wpływają następujące kryteria:

- jakość, którą najprościej opisać jako zestaw cech, od których zależy poziom zadowolenia z realizacji zamówienia;
- warunki dostaw, najbardziej wymierne z kryteriów oceny łańcucha dostaw, gdzie najbardziej na nie wpływa jakość i terminowość dostawy;
- czas całego cyklu;
- straty.

Dwa ostatnie kryteria są tymi od których przede wszystkim zależy ocena wydajności całego procesu, przedsiębiorstwa muszą dążyć do tego by straty były jak najmniejsze. Przy najkrótszym czasie procesu i w tym najbardziej systemy do zarządzania magazynem mogą wesprzeć osoby odpowiedzialne za procesy magazynowe. Minimalizują one ryzyko błędu ludzkiego poprzez chociażby wcześniej wspomniane szablony dokumentów ale też poprzez ciągłe monitorowanie procesów w łańcuchu dostaw. Mniej oczywistym atutem w tym przypadku, może być funkcja ustalenia stanów minimalnych, która przy ciągłych i dynamicznych zmianach w procesie dystrybucji, chroni przedsiębiorstwo przed sytuacją, w której okaże się, że brakuje towaru. Najczęściej działa ona w taki sposób, że kiedy ilość danego towaru zbliży się, bądź spadnie poniżej stanu minimalnego, to bez ingerencji pracownika magazynu, zostanie wysłane powiadomienie, w formie na przykład maila, do handlowca, który powinien wtedy zadbać o zamówienie ubywającego towaru.

Bardzo istotnym udogodnieniem, które powinna zaoferować aplikacja do zarządzania magazynem jest bezpieczeństwo, które powinno się rozumieć w tej kwestii jako zabezpieczenie danych przed ich utratą, bądź wyciekiem na zewnątrz. Kopie zapasowe powinny być robione na tyle często, na ile to możliwe, a poszczególni użytkownicy systemu powinni otrzymać od administratora tylko uprawnienia konieczne, zależne od wykonywanej

funkcji oraz stanowiska. W przedsiębiorstwach często między administratorem systemu w firmie, a danymi użytkownikami znajduje się ktoś jeszcze, może być to kierownik magazynu, bądź dyrektor danego działu. Często to ta właśnie osoba jest odpowiedzialna za przyznawanie uprawnień do danych funkcji, bądź widoków do danych sekcji, co zapobiega sytuacji, w której nieuprawniony pracownik, posiada dostęp do mniej, bądź bardziej wrażliwych danych. Pozostając w kwestii bezpieczeństwa danych i kopii zapasowych, należy prowadzić taką politykę zarządzania systemem, aby w przypadku problemów z działaniem aplikacji, bądź całkowitym przestoju w jej funkcjonowaniu, awarii lub innej kwestii, uniemożliwiającej działanie systemu, dane nie zostały utracone, a przedsiębiorstwo było w stanie dalej działać tak samo, bądź bez większych problemów, czy przestojów. W przypadku chwilowej utraty zasilania, bądź innej, krótkoterminowej awarii, mając tu na myśli takiej, której czas trwania nie przekroczy doby, najczęściej przywrócenie kopii zapasowej i sprawne uzupełnienie danych o procesy, które były realizowane danego dnia, bądź towar, który był przyjęty, wydany czy transportowany, jest wystarczający i skutki awarii będą znikome. Kiedy awaria może zająć więcej czasu, należy mieć na dany obrót wypadków, przygotowany schemat działania. Najczęściej polega to na tym, że wszystkie procesy będą przeprowadzane w formie papierowej, wydania wewnętrzne notowane na liście, specjalnie na taką chwilę przygotowanej, przyjęcia i wydania prowadzone tylko w tradycyjnej, papierowej formie, by po przywróceniu funkcjonowania systemu, posegregowane jasno i przejrzyste dane wprowadzić do systemu. Należy jednak w miarę możliwości takich sytuacji unikać, ponieważ jest to zawsze czasochłonny proces i wraz z ilością czasu przestoju, rośnie ryzyko błędu ludzkiego przy wprowadzaniu danych. Proces uzupełniania tych właśnie danych również powinien mieć jasną, określona politykę, tak aby nie powstał chaos w systemie.

Przy takim obrocie spraw bardzo często największe problemy pojawiają się w kwestii ciągłego uzupełniania zapasów. Partnerzy handlowi, dostawcy, czy kontrahenci, w zależności o charakterystyki przedsiębiorstwa, często są zaopatrywani w raporty, dane o sprzedaży i informacje, często konieczne przy planowaniu krótkoterminowych prognoz zapotrzebowania na zapasy. Cały łańcuch dostaw opiera się o często ciągle aktualizowane dane, które przesyłane są zazwyczaj w górę tego łańcucha. Przy przestoju, bądź awarii, niezwykle istotne jest by nie spowodowało to przerwania łańcucha. Zawsze należy mieć

przygotowaną strategię na takie wypadki, ponieważ problem z jednym elementem, potrafi przerwać nawet cały łańcuch, co potrafi wygenerować ogromne koszty i narazić przedsiębiorstwo na straty nie tylko finansowe ale i wizerunkowe.

Łańcuch dostaw funkcjonuje trochę jak organizm, problem z jednym jego elementem, potrafi sprawić, że cały przestanie działać tak, jak powinien. To wszystko sprawia, że im bardziej zautomatyzowany jest magazyn, tym większą świadomość należy mieć w trakcie jego użytkowania, mając na uwadze jak wrażliwy jest cały system logistyczny.

To wszystko nie zmienia jednak faktu, że sprawny system oferuje nieporównywalnie większą ilość zalet, gdy przy świadomym korzystaniu z niego, ryzyko jest znikome i zdecydowanie mniejsze, niż w przypadku ryzyka błędu ludzkiego.

Reasumując, działający system magazynowy powinien być intuicyjny i prosty w użytkowaniu, powinien porządkować procesy i towary, tak by jego funkcje były proste w użyciu, a towary łatwe do znalezienia. Powinien realnie wpływać na czas trwania tych procesów, jak i przyspieszyć zadania które są codziennością w działającym magazynie, równocześnie redukując koszty związane z jego pracą.

2. Opis sytuacji faktycznej

Aplikacja jest skierowana głównie dla małych przedsiębiorstw, które często pracę swojego magazynu opierają na programach niekoniecznie stworzonych w tym celu, takich jak chociażby Excel. Może być ona pierwszym ułatwieniem, jak i również pierwszym krokiem przy rozwoju magazynu. Aplikacja ma na celu usystematyzować codzienne zadania pracowników i ułatwić wgląd w działanie magazynu.

W najmniejszych firmach może działać ona lokalnie, przy użyciu serwera Apache, jednak istnieje możliwość zainstalowania jej na zewnętrznym serwerze, dostęp do niej jest możliwy poprzez użycie przeglądarki internetowej. Została ona napisana w języku PHP przy użyciu framework'a Laravel.

Aplikacja posiada sześć modułów:

- magazyny, odpowiada on za dodawanie i zarządzanie miejscami do magazynowania;

- produkty, gdzie można dodać, zarządzać produktami, ich stanem, jak i określić miejsce ich przechowywania.
- kontrahenci, miejsce w którym można dokumentować oraz zarządzać bazą zainteresowanych;
- przyjęcia towaru (PZ), moduł w którym znajdują się, w postaci listy, dokumenty PZ oraz szablon do tworzenia tych dokumentów;
- wydania towaru (WZ), podobnie jak w przypadku PZ, dotyczący jednak dokumentów wydania towaru;
- archiwum PZ/WZ, do którego trafiają dokumenty przyjęcia i wydania towaru, po wcześniej określonym czasie od ich stworzenia.

Aplikacja posiada podział na użytkowników, w zależności od pełnionej w przedsiębiorstwie roli (magazyn, biuro, etc.), oraz administratora. Administrator posiada wszystkie dostępne uprawnienia oraz może je nadawać poszczególnym użytkownikom, zarządzać nimi, czy tworzyć nowych. Pozostali użytkownicy posiadają ograniczone uprawnienia, tylko te, które są wymagane do ich codziennej pracy, jak i ukryty widok wrażliwych danych, chociażby takich jak wgląd w uprawnienia innych użytkowników systemu.

Tabele baz danych są tworzone przy pomocy migracji i proces ich tworzenia, jak i całość działania aplikacji odbywa się w kodzie źródłowym.

Aplikacja jest bardzo intuicyjna i prosta w obsłudze, co sprawia, że nowy użytkownik szybko nauczy się z niej korzystać, w czym pomaga także przyjazny interface.

3. Badania własne / opis metod, technik i narzędzi badawczych / aparatura / oprogramowanie

W celu stworzenia aplikacji i napisania pracy sporo czasu poświęcono na analizę literatury dotyczącej tematu zarządzania magazynem, budowania łańcucha dostaw, oraz działania istniejących już aplikacji magazynowych, czy całych systemów ERP. W tym celu

przeczytano książki polskich autorów znanych w temacie zarządzania magazynem i łańcuchem dostaw, jak i artykuły naukowe w tym temacie.

Po analizie literatury i zdobyciu wstępnej wiedzy, przystąpiono do przygotowania, a następnie przeprowadzenia badań, w postaci wywiadu, w dziale logistyki w firmie „UTAL”, w małych i średnich przedsiębiorstwach na terenie województwa wielkopolskiego, oraz ze studentami WSL w Poznaniu. Równolegle dobierano technologię, wybierając narzędzia wykorzystane do projektu, a następnie tworząc już wstępne wersje aplikacji w języku PHP, przy wykorzystaniu framework'a Laravel. Po podsumowaniu badań i wyciągnięciu z nich wniosków podjęto decyzję o stworzeniu aplikacji skierowanej do tych najmniejszych przedsiębiorstw, takich w których nie funkcjonują jeszcze żadne systemy do zarządzania magazynem.

C2. Efekty realizacji projektu

W trakcie realizacji projektu i przeprowadzania analizy literatury oraz danych z przeprowadzonych badań, zauważono, że duże przedsiębiorstwa, przy zarządzaniu magazynem, wykorzystują narzędzia jakie oferują systemy ERP. Odrębne aplikacje magazynowe nie są wśród nich popularne, za to te najmniejsze przedsiębiorstwa nie korzystają najczęściej z pomocy systemów, czy poszczególnych aplikacji stworzonych docelowo do zarządzania przestrzenią magazynową. W tym celu najczęściej używają programów, takich jak chociażby Excel, stąd właśnie najmniejsze firmy stanowią potencjalnych odbiorców aplikacji, stworzonej podczas realizacji projektu. Kierując się tymi wnioskami, założono iż aplikacja powinna być intuicyjna, łatwa w obsłudze i oferująca proste narzędzia, które lepiej spełnią poszczególne zadania w małym przedsiębiorstwie, oraz będą bardziej przejrzyste i ukierunkowane na zarządzanie magazynem, niż programy z których często te przedsiębiorstwa korzystają do tej pory.

Aplikacja posiada także intuicyjny, przejrzysty i przyjazny dla użytkownika interface, dzięki któremu nawigacja po niej oraz korzystanie z poszczególnych funkcji oraz narzędzi jakie oferuje, będzie łatwe i jasne. Pomoże ona uporządkować magazyn, jak i doprowadzi do tego, że działania w procesie magazynowania staną się szybsze i tańsze. Praca związana z

działaniem magazynu będzie bardziej usystematyzowana, a ryzyko błędu, niedociągnięcia, czy braków w stanach będzie mniejsze.

Myśląc o najmniejszych przedsiębiorstwach, które często nie chcą wydawać dużych pieniędzy na systemy wspierające pracę, aplikacja stworzona w ramach pracy dyplomowej, powinna okazać się dla nich dobrą alternatywą. Działa ona docelowo na serwerze lokalnym, jednak istnieje możliwość zainstalowania jej na serwerze zewnętrznym. Posiada konta użytkowników, w zależności od stanowiska i od funkcji jaką pełni dany pracownik na magazynie. Pozwala tworzyć miejsca magazynowe, jak i przypisywać do nich towar, materiał, czy produkowane dobra, w raz z ich stanem.

Te funkcje zostały przedstawione pracownikom w małych przedsiębiorstwach oraz ich właścicielom, którzy stwierdzili, że chętnie skorzystali by z takiego rozwiązania w swoich firmach i znacznie ułatwiłoby im to pracę.

W kwestii wyboru form prowadzenia badań oraz zgłębiania wiedzy, zdecydowano się na literaturę w postaci książek i artykułów naukowych, oraz rozmowie ze studentami WSL w Poznaniu w celu zebrania wiedzy w temacie, następnie prowadzenia wywiadów w firmie „Utal”, korzystającej z systemu ERP, oraz w małych i średnich przedsiębiorstwach na terenie województwa wielkopolskiego. Analizując zebraną wiedzę i badania zdecydowano o wyborze małych firm jako docelowego odbiorcy aplikacji, uznając że może się ona przyczynić do dalszego rozwoju tychże przedsiębiorstw.

C3. Użyteczność projektu

Projekt powstał z myślą o zaspokojeniu potrzeb małych przedsiębiorstw, takich które nie korzystają z żadnego systemu wspierającego pracę magazynu. Te właśnie firmy najczęściej przy zarządzaniu magazynem używają programów i aplikacji, które najczęściej nie są dedykowane i stworzone w takim celu. Często nie zdają sobie sprawy z tego jak aplikacja dedykowana wesprze procesy i uporządkuje pracę na magazynie, co przyniesie wymierne skutki dla przedsiębiorstwa oraz podniesie komfort pracy, nie inwestując dużych pieniędzy w cały zintegrowany system. Aplikacja stworzona na potrzebę pracy inżynierskiej jest odpowiedzią na zapotrzebowanie tych najmniejszych firm. Aplikacja jest elastyczna w

użyciu, w przypadku firmy, która ma jednego pracownika zajmującego się kwestiami magazynowania. Może być on jedynym użytkownikiem oraz zarządcą. Administratorem również może zostać przedsiębiorca, bądź osoba odpowiedzialna za zarządzanie, na przykład kierownik, który będzie przyznawał uprawnienia w aplikacji. Jest to zalecane zwłaszcza w przypadku, gdy za działanie magazynu odpowiada więcej niż jeden pracownik. Aplikacja przede wszystkim porządkuje pracę magazynu, daje możliwość wygodnego oglądu w stany towarów oraz produkowanych dóbr, czy półfabrykatów. Oferuje wyższy poziom kontroli nad pracą magazynu i konkretnymi procesami, które w nim zachodzą. Aplikacja jest prosta w użyciu, posiada przyjazny interface, a jej działanie jest bardzo intuicyjne, co sprawia, że nie ma potrzeby prowadzenia dodatkowych szkoleń, w związku z jej wdrożeniem. Jest też bardzo elastyczna, nie ma jednego, konkretnego modelu, który by mówił jak jej użyć, można to dostosować do potrzeb przedsiębiorstwa oraz własnej wygody.

C4. Autoewaluacja zespołu projektowego

Bartosz Gwiazdowski

W trakcie realizacji projektu zdobyłem przede wszystkim umiejętności miękkie, w głównej mierze to ja pełniłem rolę konsultanta pomiędzy potencjalnymi odbiorcami systemu, a programistami. Były to dla mnie nowe doświadczenia, ponieważ wcześniej nie pracowałem bezpośrednio z potencjalnym klientem. Sądzę, że dzięki temu zdobyłem umiejętności, których wcześniej nie posiadałem, takie jak zdolność zadawania właściwych i konkretnych pytań, czy wyciąganie trafnych wniosków i rozpoznawanie potrzeb rozmówcy. Oprócz tego, do moich zadań należało zebranie materiałów i źródeł wiedzy, potrzebnych do wykonania projektu. Podczas tego zadania rozwinałem umiejętność szukania odpowiednich i potrzebnych w danym momencie fragmentów, czy cytatów, jak i wyciągania z nich wniosków i umiejętnego użycia w swojej pracy. Myślę, że dobrze sprostałem swoim zadaniom w okresie pracy nad projektem dyplomowym, jak i reszta zespołu, pomimo tego, że w trakcie pisania pracy inżynierskiej, skład partnerów uległ zmianie, mianowicie zmniejszył się o jedną osobę, którą musielibyśmy wykreślić z powodu braku wkładu swojej pracy w trakcie tworzenia projektu. Sprawiło to, że czas pisania pracy uległ wydłużeniu, jak i pewne plany dotyczące tego jak projekt ma finalnie wyglądać uległy zmianie, jednak jako zespół, w uboższym już składzie personalnym, zaczeliśmy współpracować ściślej, jak i ciężej, co sprawiło że poradziliśmy sobie z nową koncepcją, jak i dodatkowymi obowiązkami i projekt finalnie powstał.

Rafał Graś

Podczas realizacji projektu odpowiedzialny byłem za stworzenie dokumentacji technicznej aplikacji oraz za konsultacje dotyczące wymaganych funkcjonalności. Nasz projekt z założenia nie jest przystosowany do wdrożenia w konkretnej firmie a stanowi bazę którą dość łatwo można dostosować pod konkretne wymagania klienta. Jednak staraliśmy się opisać program w taki sposób aby potencjalny klient mógł łatwo znaleźć funkcje które będą go interesowały. Nauczyłem się przede wszystkim opisywania często skomplikowanego kodu tak aby był zrozumiały dla osób nie mających styczności z programowaniem posiadający podstawową wiedzę w zakresie informatyki. Wyzwaniem podczas realizacji projektu okazało się znaleźć funkcjonalności które są niezbędne dla funkcjonowaniu magazynu gdyż żaden z członków zespołu nie był bezpośrednio związany z zarządzaniem magazynem. Jednak na podstawie wywiadu przeprowadzonego przez Bartosza udało nam się zatrzymać konieczne funkcje potrzebne w prawie każdym systemie zarządzania magazynem. Dowiedzieliśmy się, że bardzo często potrzebna jest możliwości generowania PZ (przyjęcie z zewnętrz) i WZ (wydanie na zewnętrz). Mimo pewnych problemów przy realizacji projektu jestem zadowolony z jego końcowego rezultatu.

Adrian Bąk

W okresie trwania realizacji projektu rozwinałem przede wszystkim swoje umiejętności z zakresu programowania w języku PHP z wykorzystaniem framework'a Laravel. Byłem w głównej mierze odpowiedzialny za cały backend, frontend oraz bazę danych aplikacji projektowej. Nie było to łatwe zadanie. Było to zadanie czasochłonne, pracochłonne oraz wymagające ciągłego poszukiwania nowych, nieznanych mi rozwiązań, ale dzięki temu nauczyłem się nowych rzeczy oraz poszerzyłem swoje umiejętności. Największym problem stanowiła dla mnie baza danych, którą miał się zająć jeden z partnerów, natomiast ze względu na marginalne zainteresowanie z strony jego osoby projektem, został podczas narady zespołu wykreślony z naszych szeregów. Baza danych działa, natomiast nie tak jak bym chciał, z czego nie jestem do końca zadowolony. Jedną z barier na początku ustalania koncepcji aplikacji był fakt, że żaden z członków zespołu nigdy nie miał styczności z pracą stricte związaną z magazynem bądź podobną aplikacją do jego zarządzania. Tutaj podziękowania należą się Bartoszowi za zebranie informacji i trafne ich zinterpretowanie po odbytych rozmowach z potencjalnymi klientami oraz Rafałowi za wyszukiwanie wskazówek do zaprogramowania poszczególnych funkcjonalności oraz skrupulatne tworzenie dokumentacji aplikacji. Po napisaniu tej aplikacji mam odczucie, że następnym razem zrobiłbym to lepiej, lepiej zoptymalizował kod, uprościł go, lepiej zaprojektował bazę danych. Mimo wszystko finalny projekt wygląda dobrze, działa, a co najważniejsze w każdej chwili można nanieść poprawki i dodać brakujące funkcjonalności. Dzięki współpracy z partnerami z zespołu udało się nakreślić koncepcję projektu oraz do bazowej wersji aplikacji

dodać niezbędne funkcjonalności. Myślę, że sprostałem oczekiwaniom zespołu, a ostateczna wersja projektu została przez nich zaakceptowana.

C5. Wykorzystane materiały i bibliografia związana z realizacją projektu

1. Czesław Skowronek, *Logistyka w przedsiębiorstwie.* , Polskie Wydawnictwo Ekonomiczne, Warszawa 2003.
2. Ireneusz Fechner, *Zarządzanie łańcuchem dostaw.* , Wyższa Szkoła Logistyki, Poznań 2007..
3. Remigiusz Kozłowski (red.), *Podstawowe zagadnienia współczesnej logistyki.* , Wolters Kluwer Polska, Kraków 2009.
4. Aleksander Niemczyk, *Zarządzanie magazynem.* , Wyższa Szkoła Logistyki, Poznań 2010.
5. Elżbieta Gołembcka (red.), *Kompendium wiedzy o logistyce.* , Wydawnictwo Naukowe PWN, Warszawa 1999.
6. Anna Czubała, *Dystrybucja produktów.* , Polskie Wydawnictwo Ekonomiczne, Warszawa 2001.
7. Paweł Kamiński, *Laravel. Kurs Video. Poziom Pierwszy. Programowanie aplikacji w PHP.* , Videopoint Grupa Helion, Gliwice 2018.
8. *Laravel Documentation – Laravel – The PHP Framework For Web Artisans.*, Laravel LLC 2020, online: <https://laravel.com/docs/8.x/readme>.
9. *PHP: PHP Manual.* ,PHP Documentation Group, online:
<https://www.php.net/manual/en/>.
10. *Zaprogramuj Życie, Jak dodawać dane do bazy? Seedery w Laravel.* ,2021 ,online:
<https://www.youtube.com/watch?v=dG1rppLxk0M>.

C6. Spis załączników

ZAŁĄCZNIK 1

DO PROJEKTU DYPLOMOWEGO

Dokumentacja techniczna aplikacji zarządzania magazynem

Informacje ogólne

Aplikacja internetowa umożliwia zarządzanie asortymentem w poszczególnych magazynach. Użytkownik ma możliwość dodania produktu oraz określenia jego miejsca przechowywania. Docelowym odbiorcą naszego systemu mają być niewielkie działy logistyczne w małych firmach, które chcą usystematyzować pracę personelu i mieć przejrzysty wgląd w działanie ów działu.

System składa się z następujących elementów:

1. **Serwera lokalnego** – komputera z zainstalowanym systemem Microsoft Windows w wersji Windows 10 lub 11 (występuje także możliwość zainstalowania aplikacji na zewnętrznym serwerze, dzięki któremu aplikacja stanie się ogólnodostępna). Na serwerze lokalnym pracują aplikacje:
 - XAMPP Control Panel – do zarządzania bazą danych, serwerem Apache oraz jako interpreter języka PHP
 - Composer – do zarządzania pakietami dla języka PHP (aplikacja wiersza poleceń)
 - Laravel – PHP Framework do aplikacji internetowych napisany w języku PHP
 - GitHub Desktop – kontrola wersji Git, klonowanie repozytorium z serwisu GitHub
2. **Aplikacji „Magazyny”** stanowiącej zasadniczy element systemu, napisanej w języku PHP przy użyciu Framework PHP Laravel.

Dostęp do aplikacji możliwy jest poprzez przeglądarkę internetową z poziomu serwera lokalnego lub po wdrożeniu aplikacji na serwer zewnętrzny z poziomu Internetu.

Sposób instalacji na serwerze lokalnym

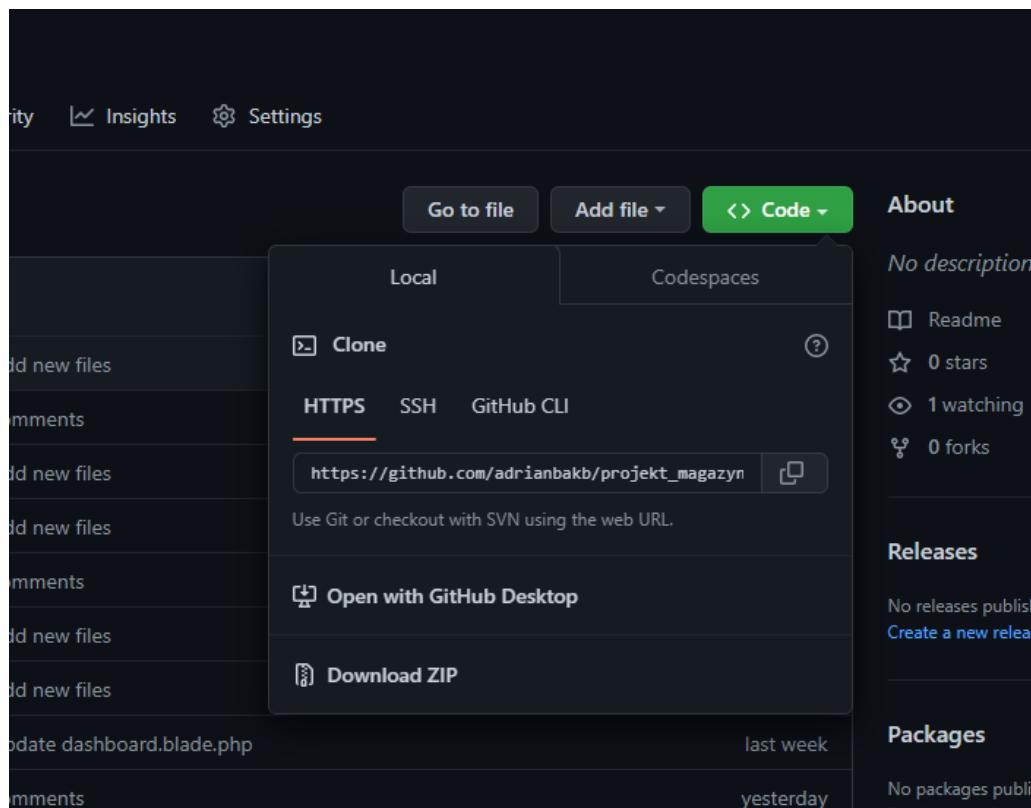
Aplikacja została zainstalowana na hostingowy serwis internetowy przeznaczony do projektów programistycznych wykorzystujący system kontroli wersji Git – Git Hub. Repozytorium aplikacji znajduje się pod wskazanym poniżej linkiem do serwisu Git Hub.

https://github.com/adrianbakb/projekt_magazyn.git

Importowanie plików aplikacji

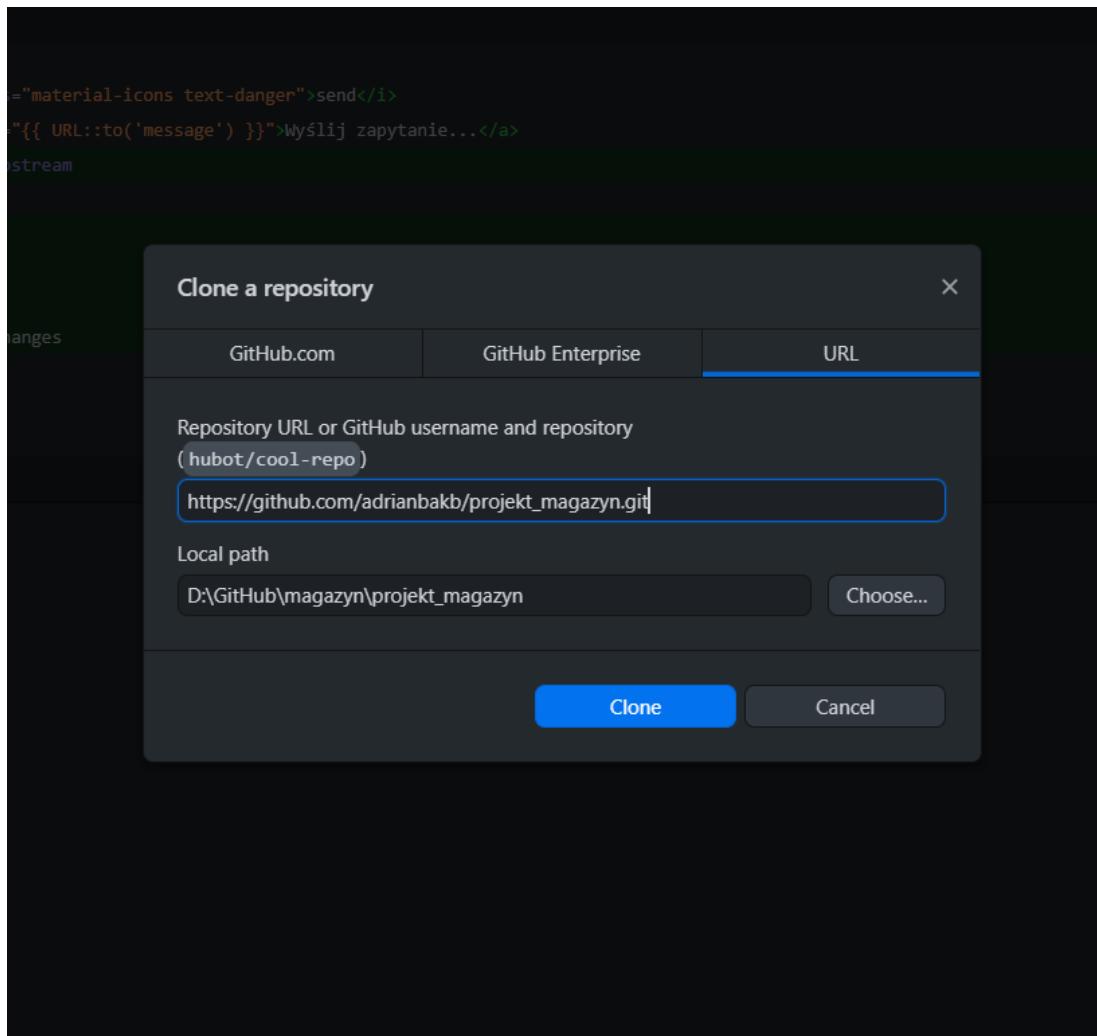
Występują dwa sposoby zainstalowania aplikacji na dysk lokalny:

1. Pobranie aplikacji jako archiwum ZIP oraz rozpakowanie go w docelowym miejscu
 - Download ZIP



Rysunek 1 - pobieranie archiwum ZIP

2. Przy pomocy aplikacji GitHub Desktop zimportowanie (sklonowanie) repozytorium aplikacji na dysk lokalny używając wcześniej skopiowany link do repozytorium

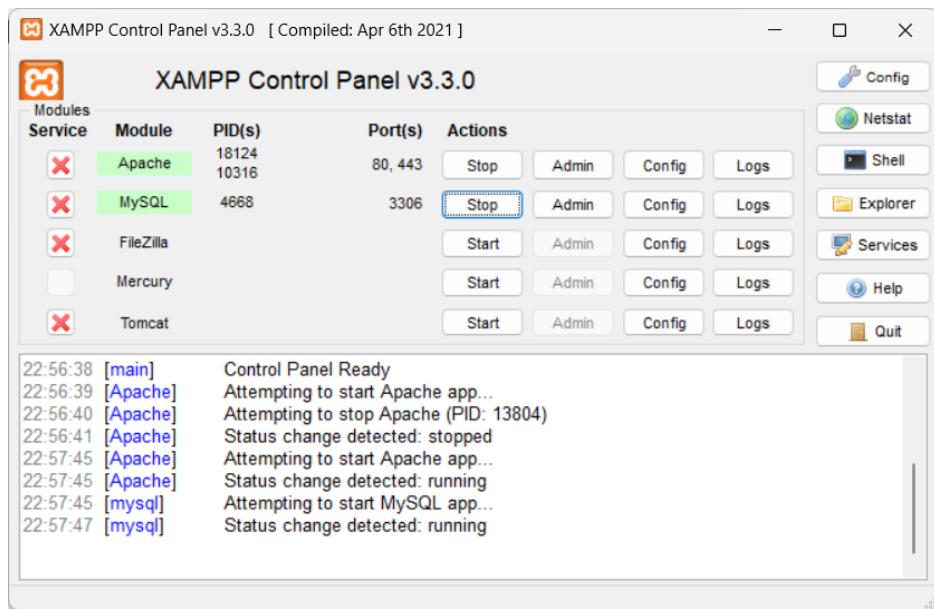


Rysunek 2 - klonowanie repozytorium GitHub Desktop

Uruchomienie środowiska aplikacji

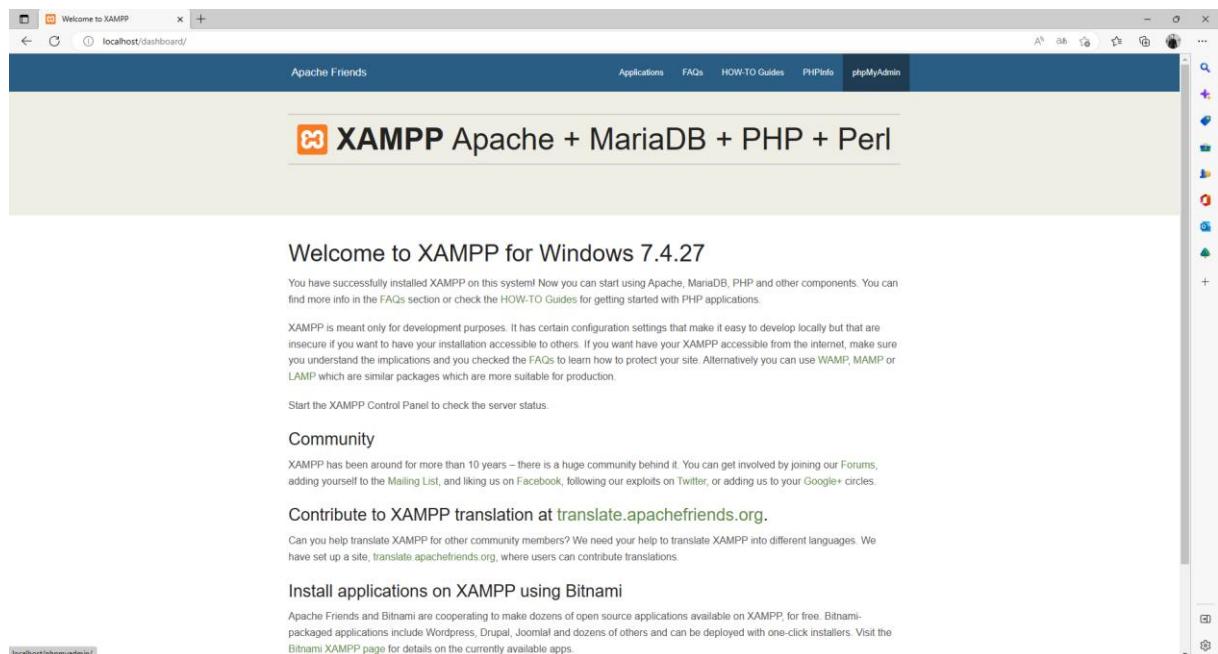
Aby móc rozpoczęć korzystanie z aplikacji należy rozpocząć od przygotowania odpowiedniego środowiska dla jej prawidłowego działania.

1. Pierwszym punktem jest uruchomienie aplikacji XAMPP Control Panel i wystartowanie serwera Apache oraz bazy danych MySQL.



Rysunek 3 -Apache i MySQL - Xampp

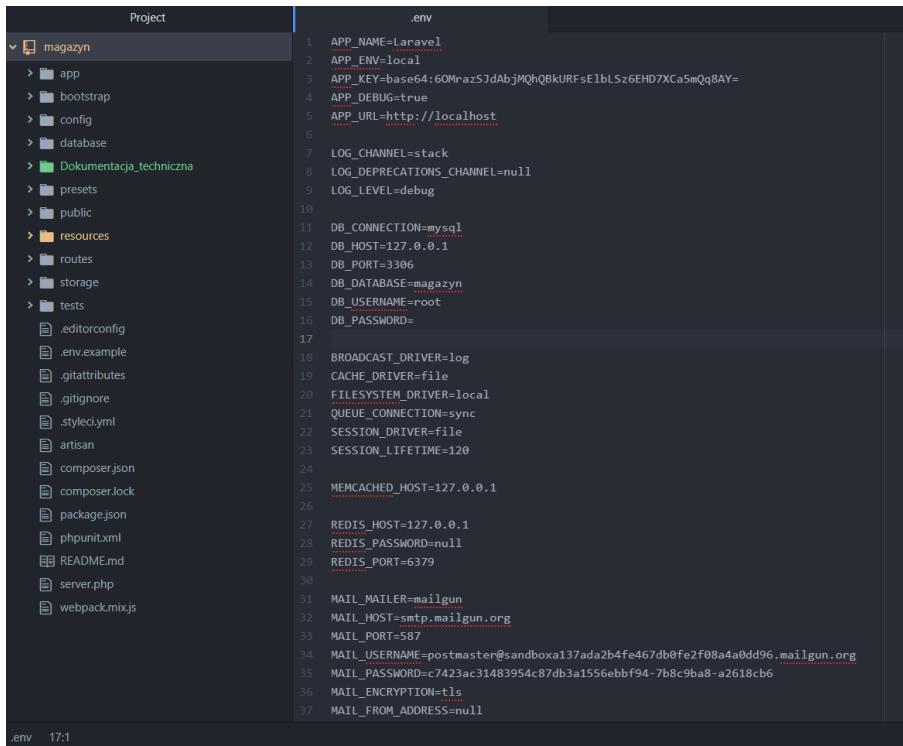
Przy pomocy przeglądarki wchodzimy na lokalną stronę do zarządzania serwerem XAMPP. Wpisując ‘localhost’ pojawi się następująca strona:



Rysunek 4 – localhost – phpMyAdmin

Należy wejść w phpMyAdmin oraz utworzyć nową pustą bazę danych.

2. W plikach konfiguracyjnych aplikacji zapisane są parametry związane z konfiguracją serwera bazy danych.



```

Project .env
└── magazyn
    ├── app
    ├── bootstrap
    ├── config
    ├── database
    ├── Dokumentacja_techniczna
    ├── presets
    ├── public
    ├── resources
    ├── routes
    ├── storage
    ├── tests
    ├── .editorconfig
    ├── .env.example
    ├── .gitattributes
    ├── .gitignore
    ├── .styleci.yml
    ├── artisan
    ├── composer.json
    ├── composer.lock
    ├── package.json
    ├── phpunit.xml
    ├── README.md
    ├── server.php
    └── webpack.mix.js

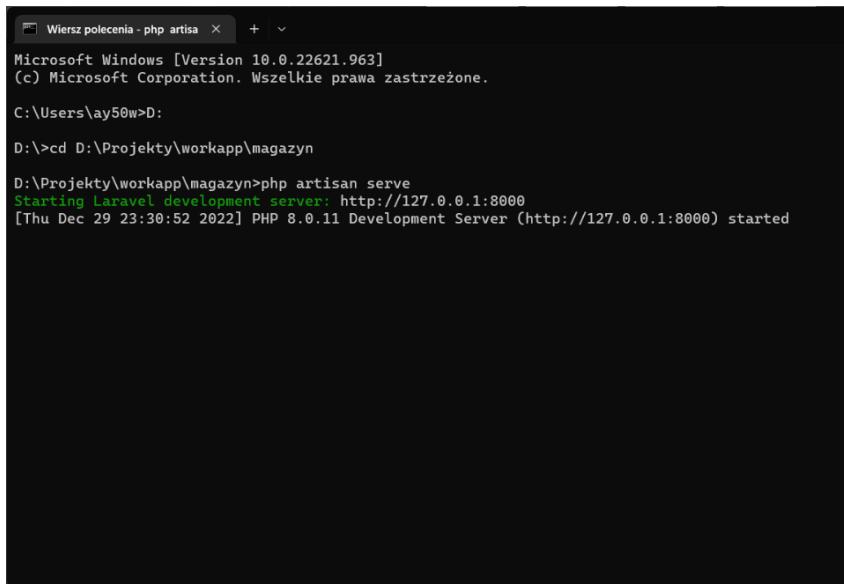
.env 17:1
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:60Mrz5JdAbjMQhQBkURFsElbLsz6HD7XCa5mQq8AY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=magazyn
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DRIVER=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
30
31 MAIL_MAILER=mailgun
32 MAIL_HOST=smtp.mailgun.org
33 MAIL_PORT=587
34 MAIL_USERNAME=postmaster@sandboxa137ada2b4fe467db0fe2f08a4a0dd96.mailgun.org
35 MAIL_PASSWORD=c7423ac31483954c87db3a1556ebbf94-7b8c9ba8-a2618cb6
36 MAIL_ENCRYPTION=tls
37 MAIL_FROM_ADDRESS=null

```

Rysunek 5 - konfiguracja pliku. Env

Należy stworzyć plik. env (przekopiować zawartość pliku. env.example) oraz skonfigurować go pod własne wymagania. W tym pliku konfigurujemy między innymi połaczenie z utworzoną już bazą danych w phpMyAdmin.

3. Następnie należy uruchomić przy pomocy wiersza poleceń serwer Artisan. W tym celu uruchamiamy wiersz poleceń, przechodzimy do folderu z aplikacją i wpisujemy polecenie ‘php artisan serve’. Aplikację można umieścić w folderze ‘htdocs’ znajdującym się w głównym katalogu Xampp. Wtedy obsługujemy wiersz poleceń znajdujący się w Control Panel Xampp.



```
Wiersz poleceń - php artisan × + ↻

Microsoft Windows [Version 10.0.22621.963]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\ay50w>D:

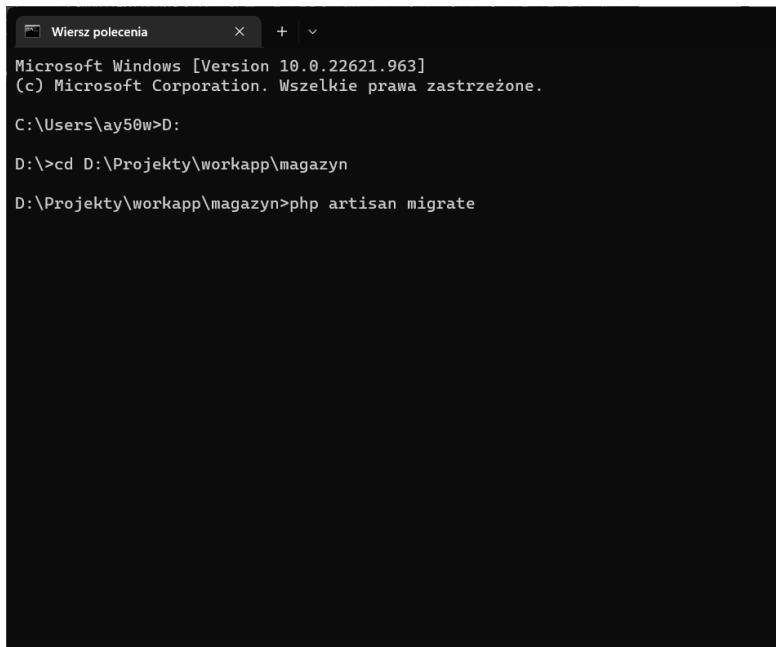
D:>cd D:\Projekty\workapp\magazyn

D:\Projekty\workapp\magazyn>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Thu Dec 29 23:30:52 2022] PHP 8.0.11 Development Server (http://127.0.0.1:8000) started
```

Rysunek 6 - wiersz poleceń ‘php artisan serve’

W tym momencie można już dostać się na stronę aplikacji. Lokalny adres aplikacji to: 127.0.0.1:8000.

4. Aby aplikacja była w pełni funkcjonalna należy jeszcze dokonać migracji bazy danych poprzez wpisanie polecenia w nowym oknie wiersza poleceń ‘php artisan migrate’. Dzięki temu struktura bazy danych zapisana w plikach aplikacji zostanie zimportowana do utworzonej wcześniej bazy danych.



```
Wiersz polecenia Microsoft Windows [Version 10.0.22621.963]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\ay50w>D:
D:\>cd D:\Projekty\workapp\magazyn
D:\Projekty\workapp\magazyn>php artisan migrate
```

Rysunek 7 - wiersz poleceń 'php artisan migrate'

Kod aplikacji zawiera również domyślne dane do zalogowania dla pierwszego użytkownika:

login: 'admin@admin.pl'

hasło: 'admin@admin.pl'

Aby te dane znalazły się w bazie danych automatycznie, należy po dokonaniu migracji zastosować kolejne polecenie: 'php artisan db:seed' lub 'php artisan migrate: fresh --seed'. Nie jest to jednak zabieg, który trzeba wykonać. Dodawanie użytkownika może odbyć się poprzez formularz rejestracji po uruchomieniu aplikacji.

W tym momencie można już przejść do uruchomienia aplikacji.

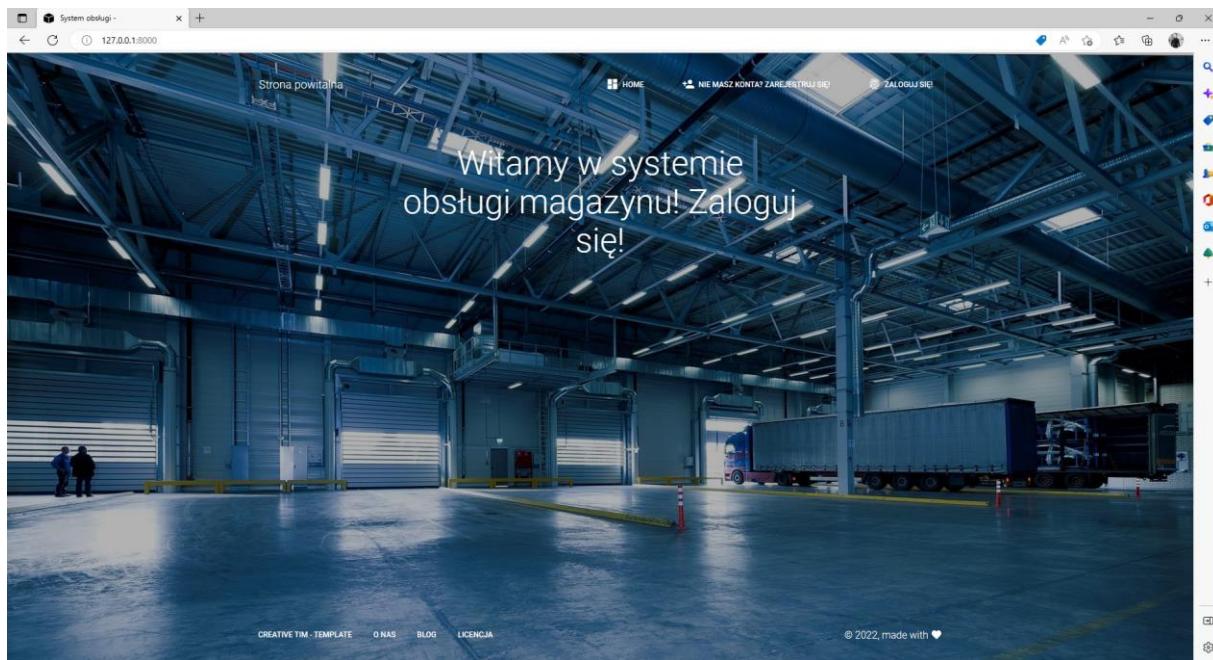
Pierwsze uruchomienie aplikacji

Przy użyciu przeglądarki internetowej i wpisaniu domyślnego adresu hosta: **127.0.0.1:8000** można zacząć korzystać z naszego systemu.

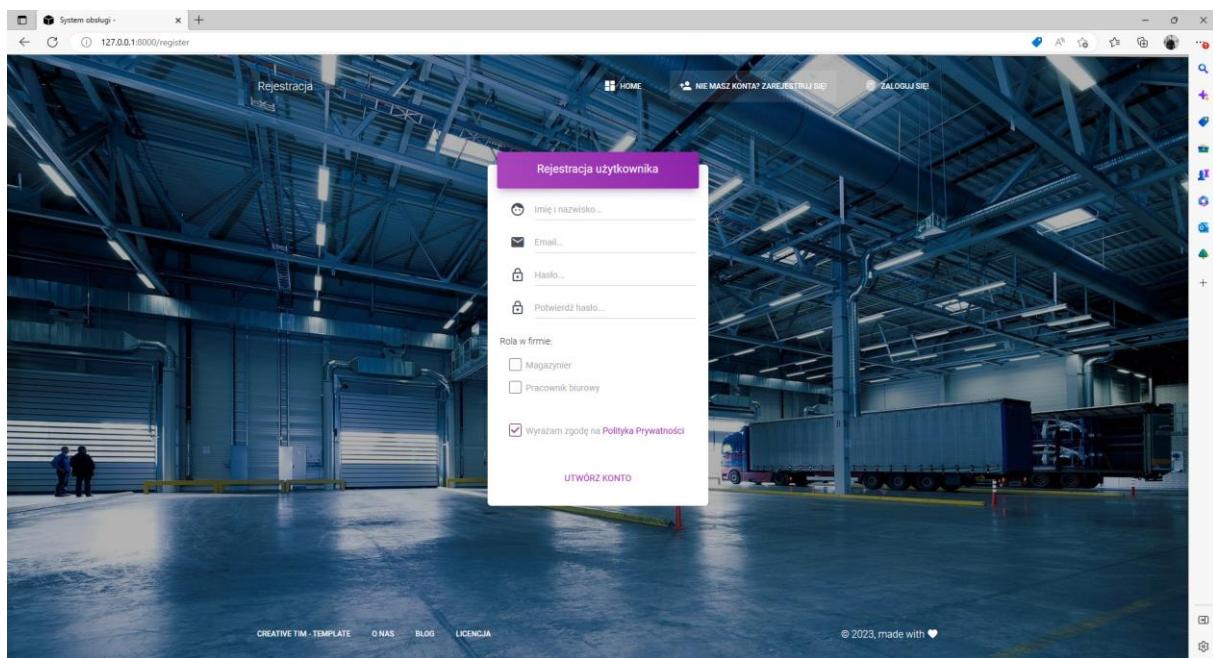
Strona powitalna

Do wyboru na stronie powitalnej są dostępne dwie opcje:

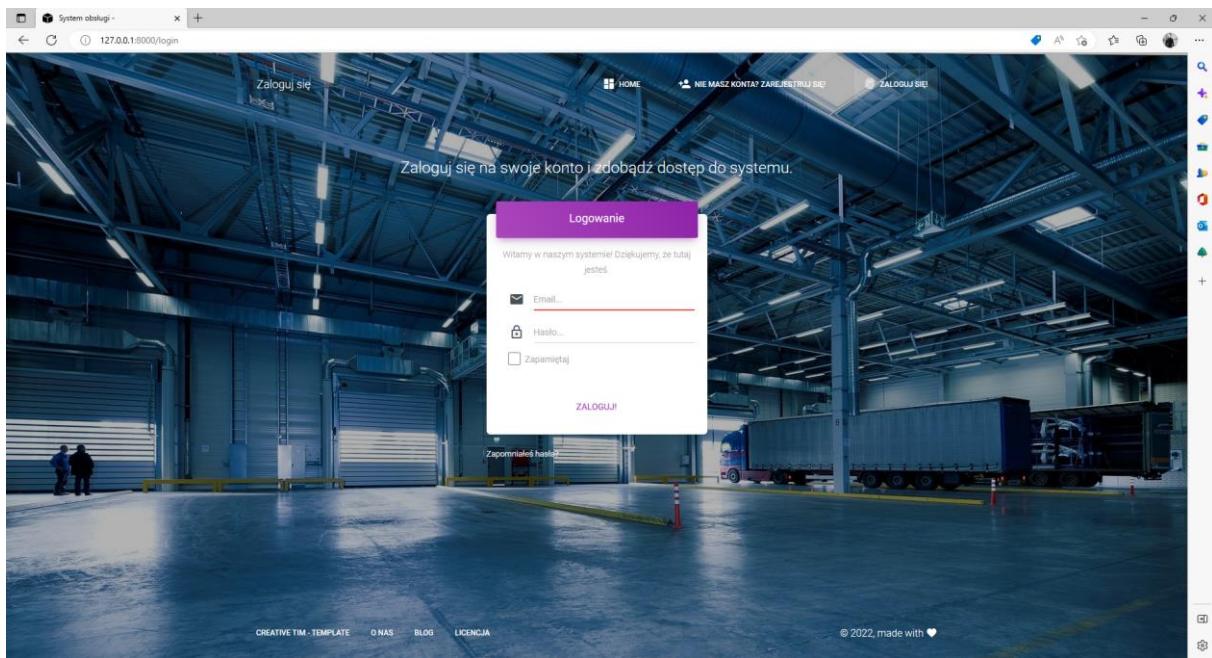
- Zaloguj się
- Zarejestruj się



Rysunek 8 - system obsługi magazynu - strona powitalna



Rysunek 9 - system obsługi magazynu - formularz rejestracji

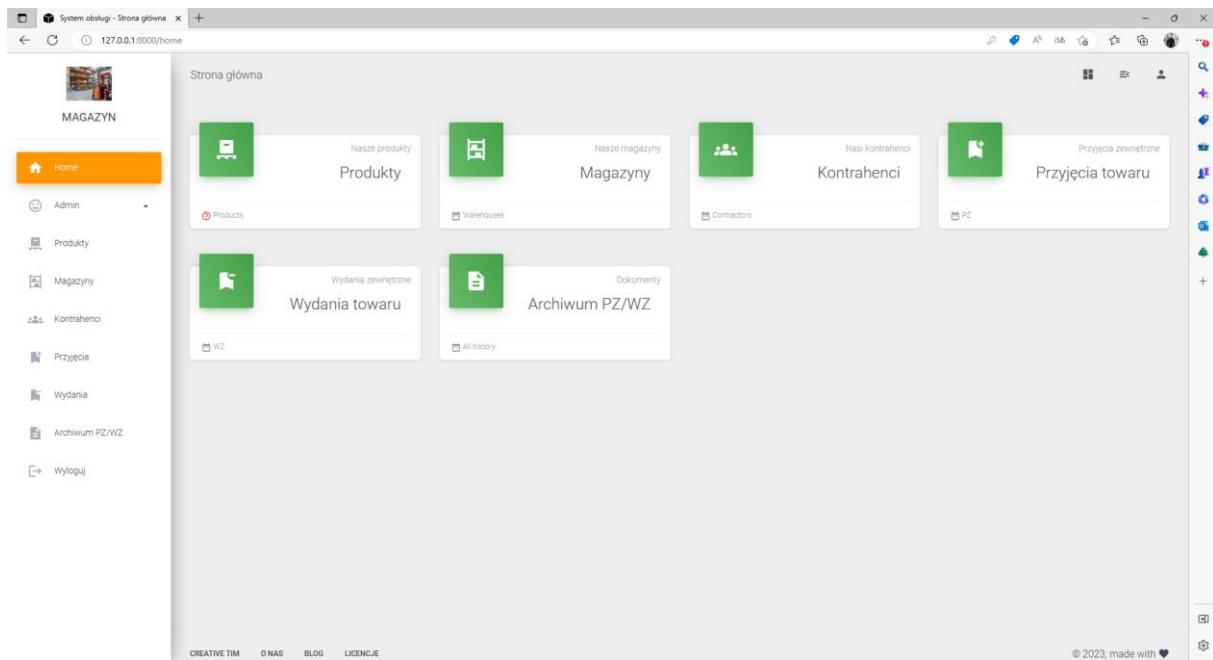


Rysunek 10 - system obsługi magazynu -formularz logowania

Strona główna po zalogowaniu

W tym miejscu użytkownik ma do wyboru dwa następujące moduły:

- Moduł 'Produkty'
- Moduł 'Magazyny'
- Moduł 'Kontrahenci'
- Moduł 'Przyjęcia'
- Moduł 'Wydania'
- Moduł 'Archiwum PZ/WZ'



Rysunek 11 - strona główna

Moduł

‘Produkty’

Ten moduł odpowiada za dodawanie produktów do bazy danych oraz określenie jego miejsca przechowywania (magazynu). Pierwsza strona modułu zawiera listę wszystkich produktów oraz docelowo ma pokazać użytkownika, który dodał daną pozycję.

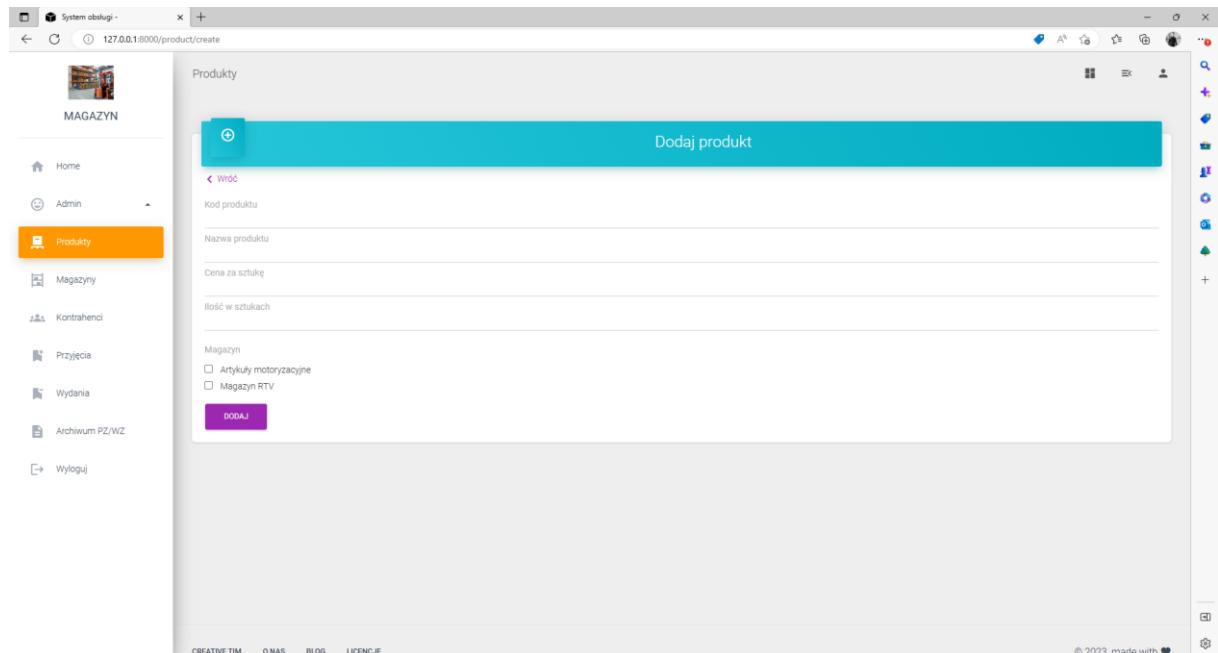
#	Kod produktu	Nazwa	Cena za szt.	Ilość	Magazyn	Dodane przez:	Zarządzaj
1	01	Plyn hamulcowy 0.3l	15 PLN	20 szt.	Artykuły motoryzacyjne (kod magazynu: 01)	użytkownik	Usuń produkt Edytuj dodany produkt
2	02	Plyn do spryskiwaczy zimowy	20 PLN	15 szt.	Artykuły motoryzacyjne (kod magazynu: 01)	użytkownik	Usuń produkt Edytuj dodany produkt
3	03	Sony TV 55"	2600 PLN	10 szt.	Magazyn RTV (kod magazynu: 02)	użytkownik	Usuń produkt Edytuj dodany produkt

Rysunek 12 - moduł ‘produkty’ – lista

W tym miejscu użytkownik ma możliwość zarządzania produktami. Opcji jest kilka:

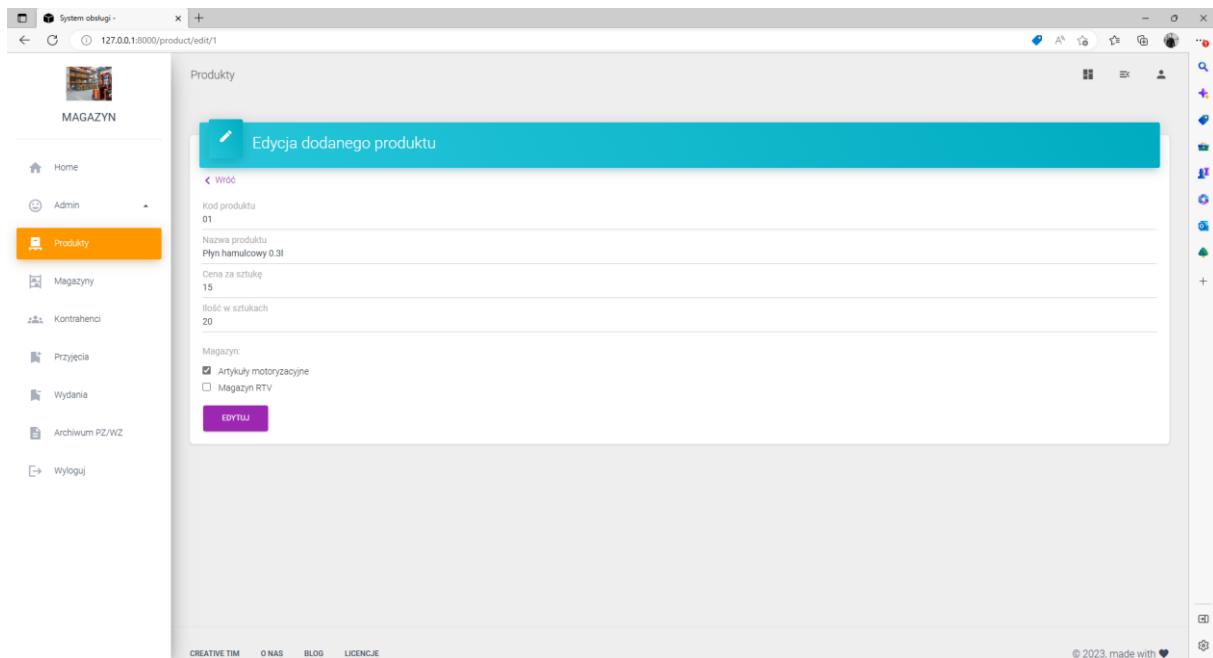
- Dodawanie nowego produktu
- Edytowanie dodanego produktu
- Usuwanie produktu

Po kliknięciu w Dodaj nowy produkt pojawi się formularz, który należy wypełnić.



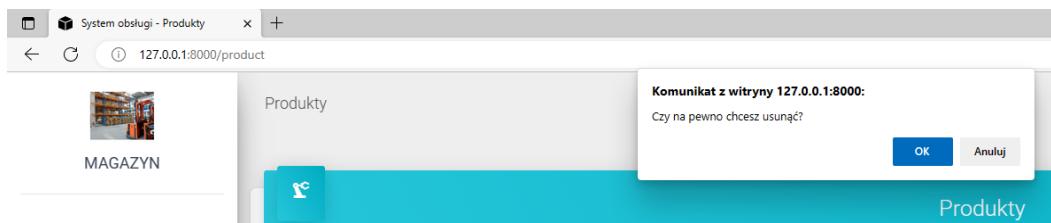
Rysunek 13- ‘moduł produkty’ - dodaj produkt

Z kolei kliknięcie w Edytuj dodany produkt pozwoli użytkownikowi edytować wprowadzone dane. Po kliknięciu w Edytuj, wyświetli się stosowny komunikat z witryny.



Rysunek 14- moduł 'produkty' - edytuj produkt

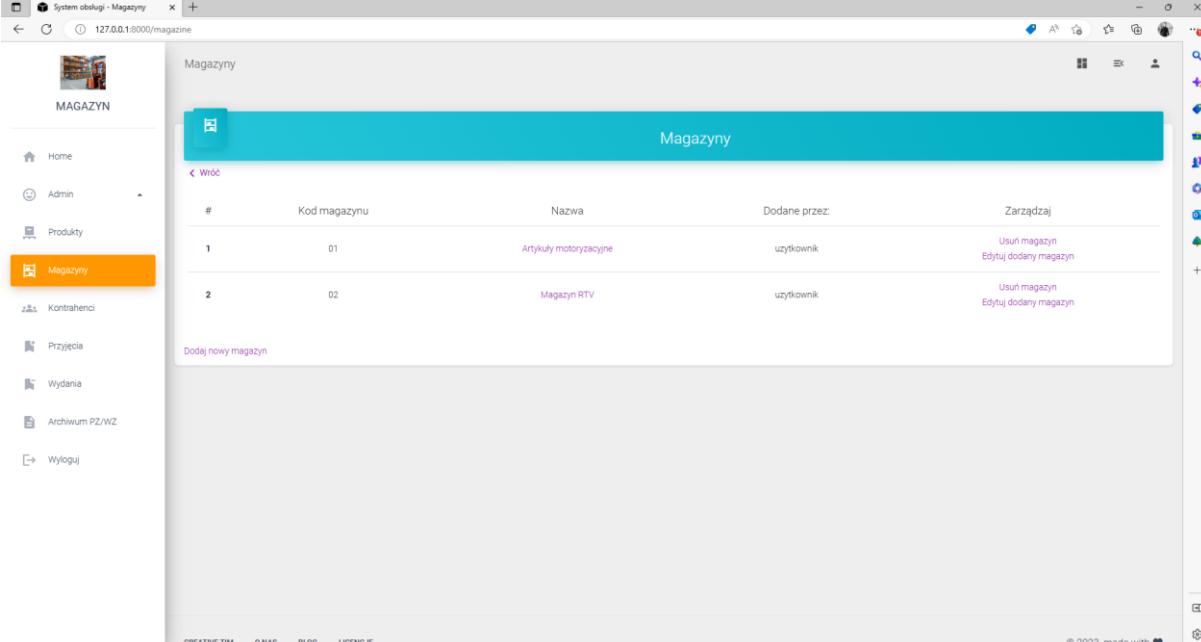
Kliknięcie w Usuń produkt spowoduje pojawienie się poniższego komunikatu z witryny:



Rysunek 15- moduł 'produkty' - usuń produkt

Moduł 'Magazyny'

Moduł odpowiada za dodawanie nowych miejsc przechowywania (magazynów, działów itd.). Pierwsza strona modułu zawiera listę wszystkich dostępnych (dodanych) miejsc oraz docelowo ma pokazać użytkownika, który dodał daną pozycję.

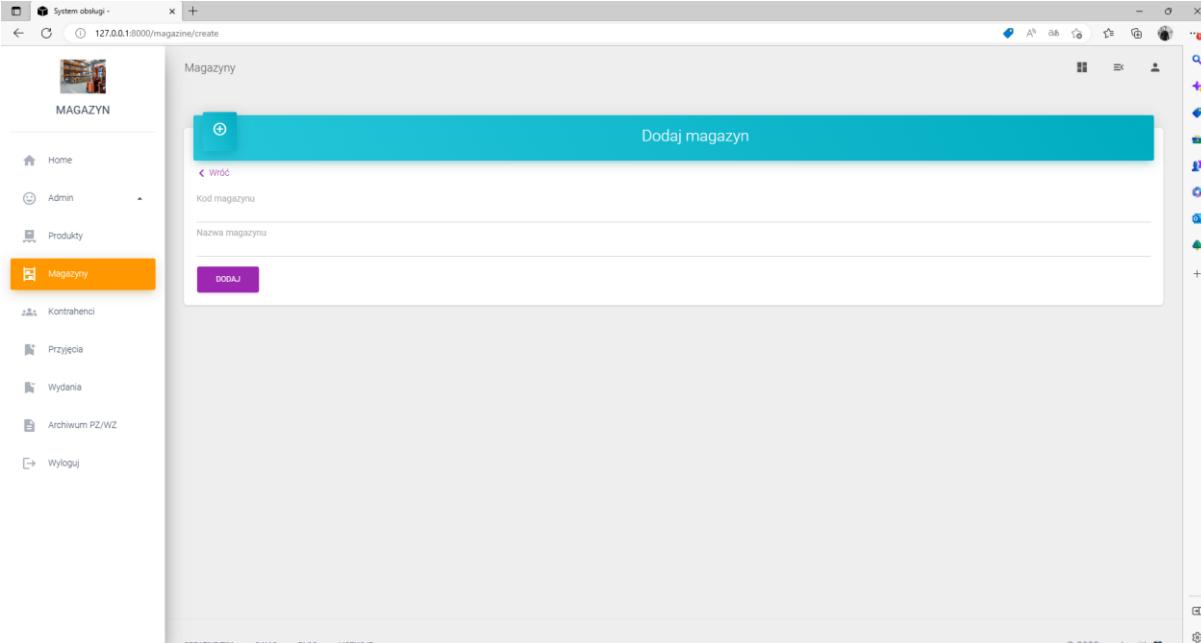


#	Kod magazynu	Nazwa	Dodane przez:	Zarządzaj
1	01	Artykuły motoryzacyjne	Użytkownik	Usuń magazyn Edytuj dodany magazyn
2	02	Magazyn RTV	Użytkownik	Usuń magazyn Edytuj dodany magazyn

Rysunek 16- moduł 'magazyn' - lista

Funkcjonalności dostępne na tej stronie niczym nie odbiegają od modułu 'Produkty'.

Schemat użytkowania jest taki sam.

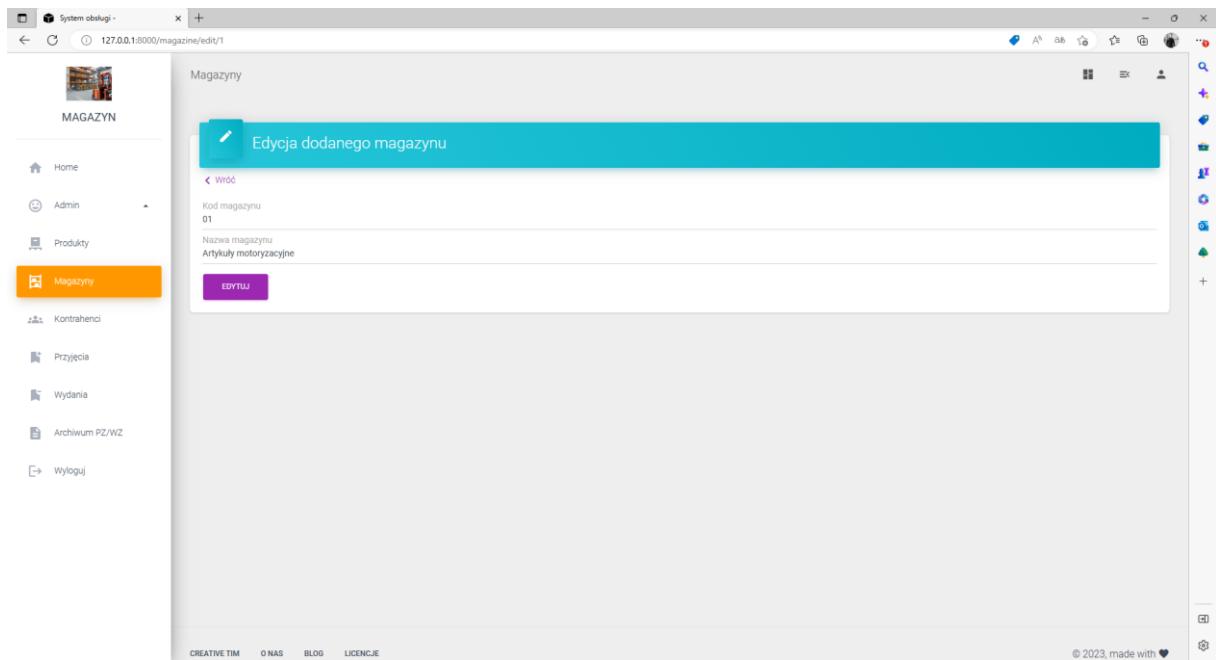


Dodaj magazyn

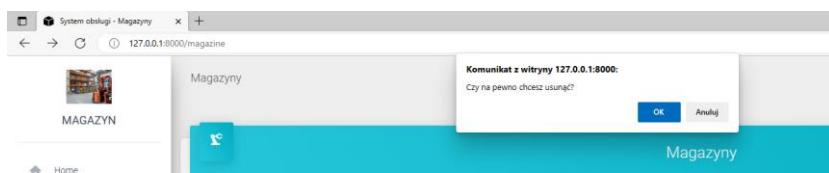
Nazwa magazynu

DODAJ

Rysunek 17- moduł 'magazyn' - dodaj magazyn



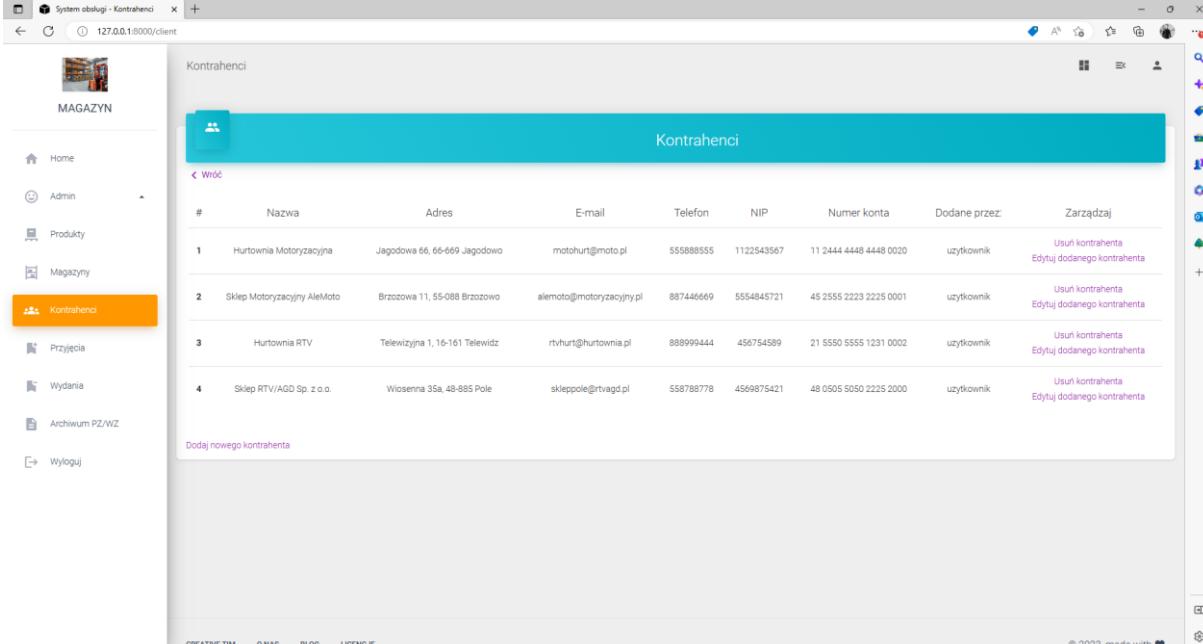
Rysunek 18 - moduł 'magazyny' - edytuj magazyn



Rysunek 19 - moduł 'magazyny' - usuń magazyn

Moduł 'Kontrahenci'

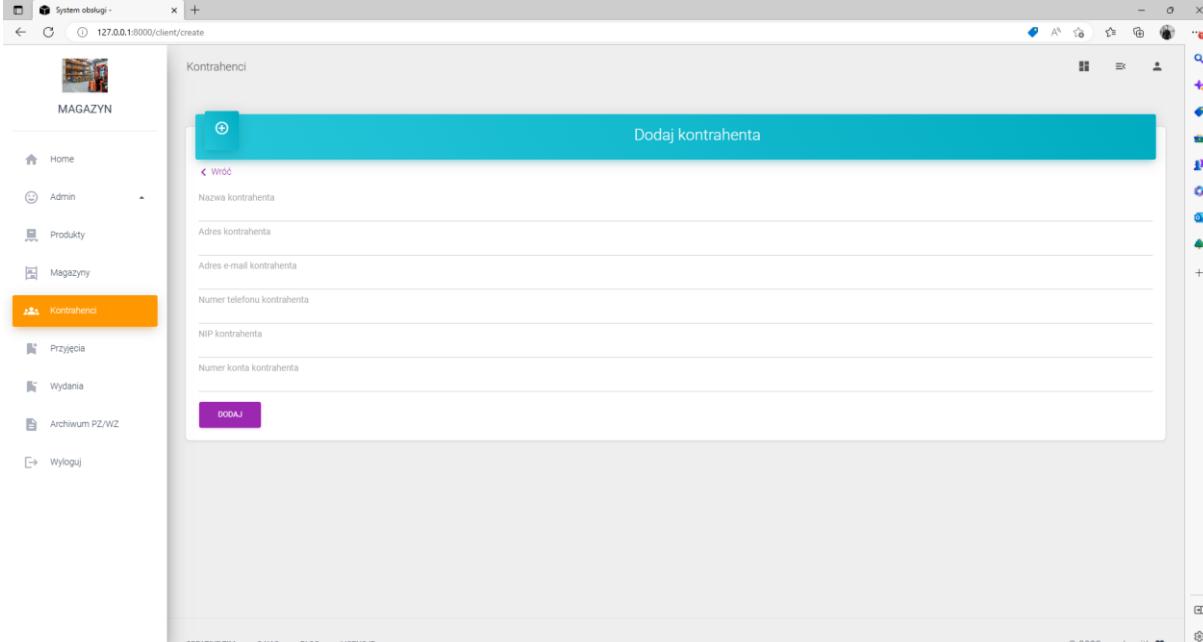
Moduł odpowiada za dodawanie nowych kontrahentów/zainteresowanych. Pierwsza strona modułu zawiera listę wszystkich dodanych podmiotów oraz docelowo ma pokazać użytkownika, który dodał daną pozycję.



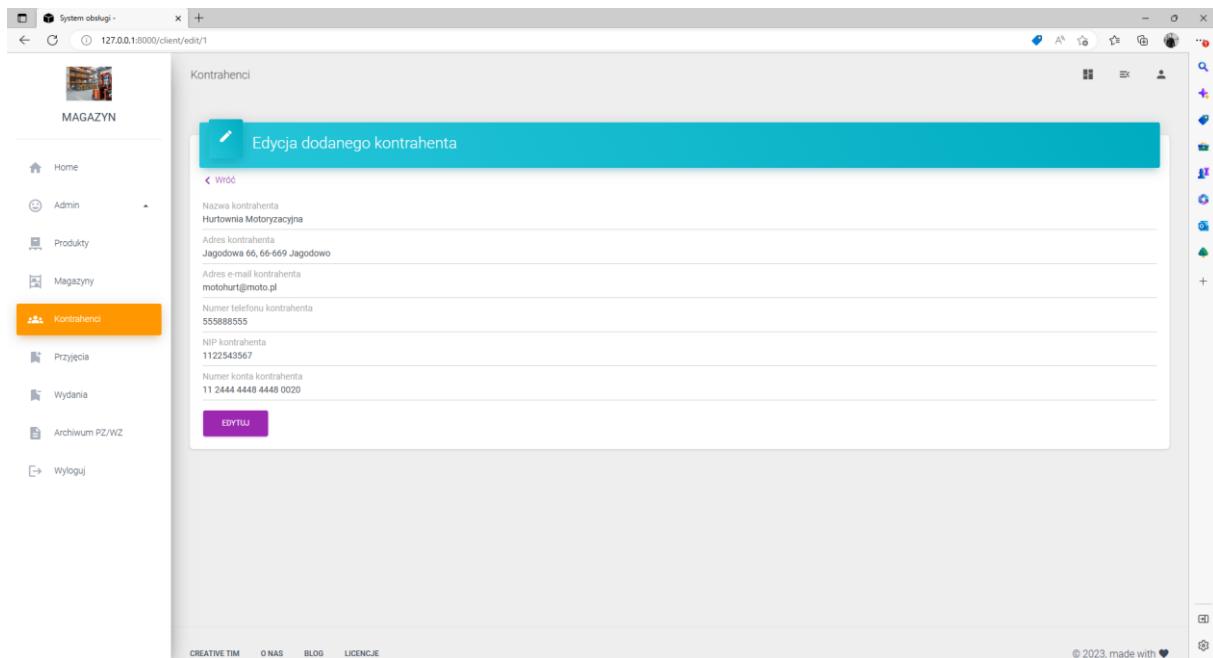
#	Nazwa	Adres	E-mail	Telefon	NIP	Numer konta	Dodane przez	Zarządzaj
1	Hurtownia Motoryzacyjna	Jagodowa 66, 66-669 Jagodowo	motoshurt@moto.pl	555888555	1122543567	11 2444 4448 4448 0020	użytkownik	Usun kontrahenta Edytuj dodanego kontrahenta
2	Sklep Motoryzacyjny AleMoto	Brzozowa 11, 55-088 Brzozowo	alemoto@motoryzacyjny.pl	887446669	5554845721	45 2555 2223 2225 0001	użytkownik	Usun kontrahenta Edytuj dodanego kontrahenta
3	Hurtownia RTV	Telewizyjna 1, 16-161 Telewidz	rthurt@hurtownia.pl	888999444	456754589	21 5550 5555 1231 0002	użytkownik	Usun kontrahenta Edytuj dodanego kontrahenta
4	Sklep RTV/AGD Sp. z o.o.	Wiłosenna 35a, 48-885 Pole	skleppole@rtvagd.pl	558788778	4569875421	48 0505 5050 2225 2000	użytkownik	Usun kontrahenta Edytuj dodanego kontrahenta

[Dodaj nowego kontrahenta](#)

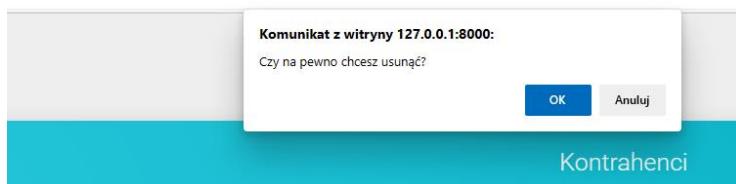
Rysunek 20 - moduł 'kontrahenci' - lista



Rysunek 21 - moduł 'kontrahenci' – dodaj kontrahenta



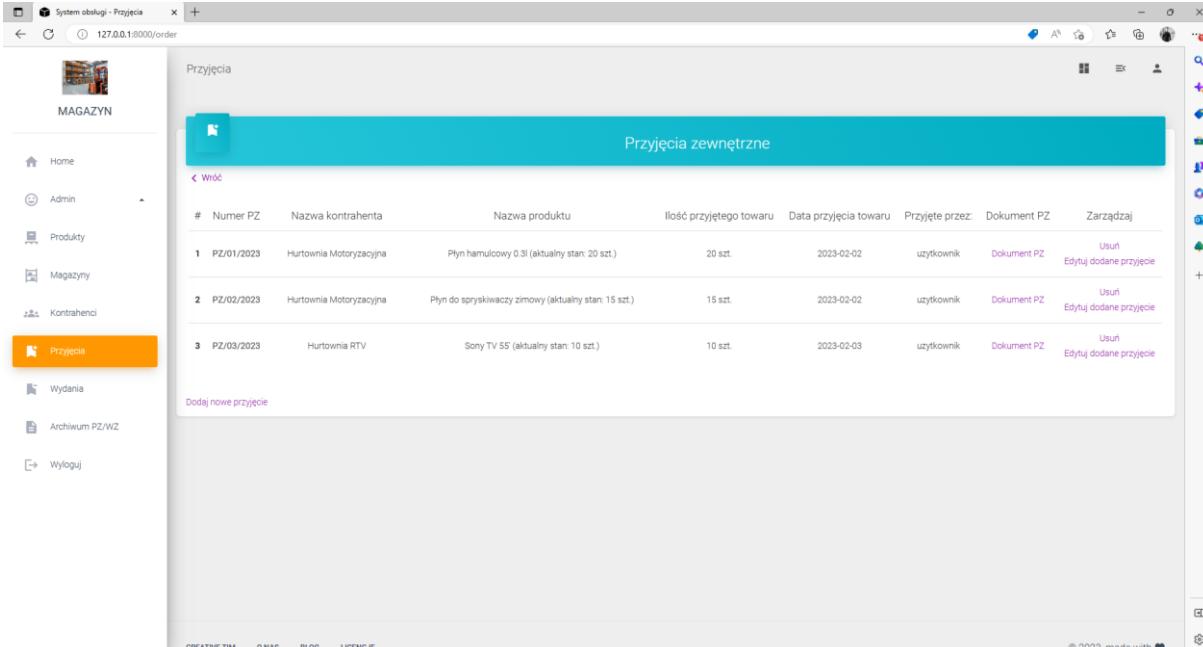
Rysunek 22 - moduł 'kontrahenci' – edytuj kontrahenta



Rysunek 23 - moduł 'kontrahenci' – usuń kontrahenta

Moduł 'Przyjęcia'

Moduł odpowiada za rejestrowanie przyjęć na magazyn. Pierwsza strona modułu zawiera listę wszystkich dodanych przyjęć oraz docelowo ma pokazać użytkownika, który dodał daną pozycję.



Przyjęcia

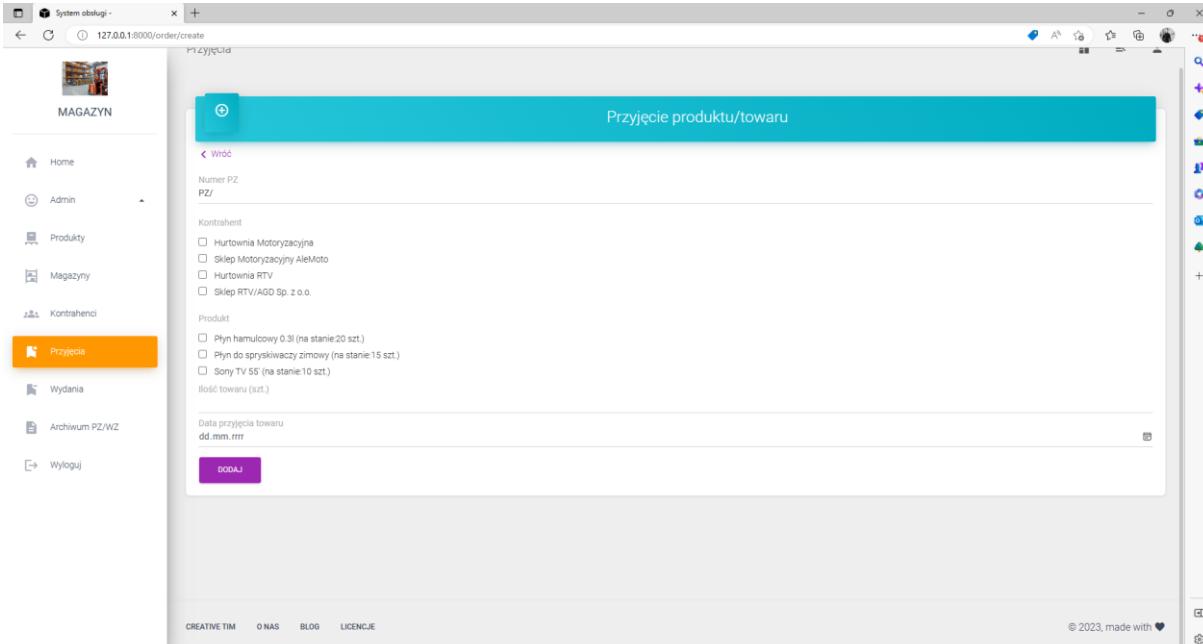
Przyjęcia zewnętrzne

#	Numer PZ	Nazwa kontrahenta	Nazwa produktu	Ilość przyjętego towaru	Data przyjęcia towaru	Przyjęte przez	Dokument PZ	Zarządzaj
1	PZ/01/2023	Hurtownia Motoryzacyjna	Plyn hamulcowy 0.3l (aktualny stan: 20 szt.)	20 szt.	2023-02-02	użytkownik	Dokument PZ	Usuń Edytuj dodane przyjęcie
2	PZ/02/2023	Hurtownia Motoryzacyjna	Plyn do spryskiwaczy zimowy (aktualny stan: 15 szt.)	15 szt.	2023-02-02	użytkownik	Dokument PZ	Usuń Edytuj dodane przyjęcie
3	PZ/03/2023	Hurtownia RTV	Sony TV 55 (aktualny stan: 10 szt.)	10 szt.	2023-02-03	użytkownik	Dokument PZ	Usuń Edytuj dodane przyjęcie

Dodaj nowe przyjęcie

CREATIVE TIME O NAS BLOG LICENCJE © 2023, made with ❤️

Rysunek 24 - moduł 'przyjęcia' – lista



Przyjęcie produktu/towaru

Numer PZ:
PZ/

Kontrahent:

- Hurtownia Motoryzacyjna
- Sklep Motoryzacyjny AleMoto
- Hurtownia RTV
- Sklep RTV/AGD Sp. z o.o.

Produkt:

- Plyn hamulcowy 0.3l (na stanie: 20 szt.)
- Plyn do spryskiwaczy zimowy (na stanie: 15 szt.)
- Sony TV 55 (na stanie: 10 szt.)

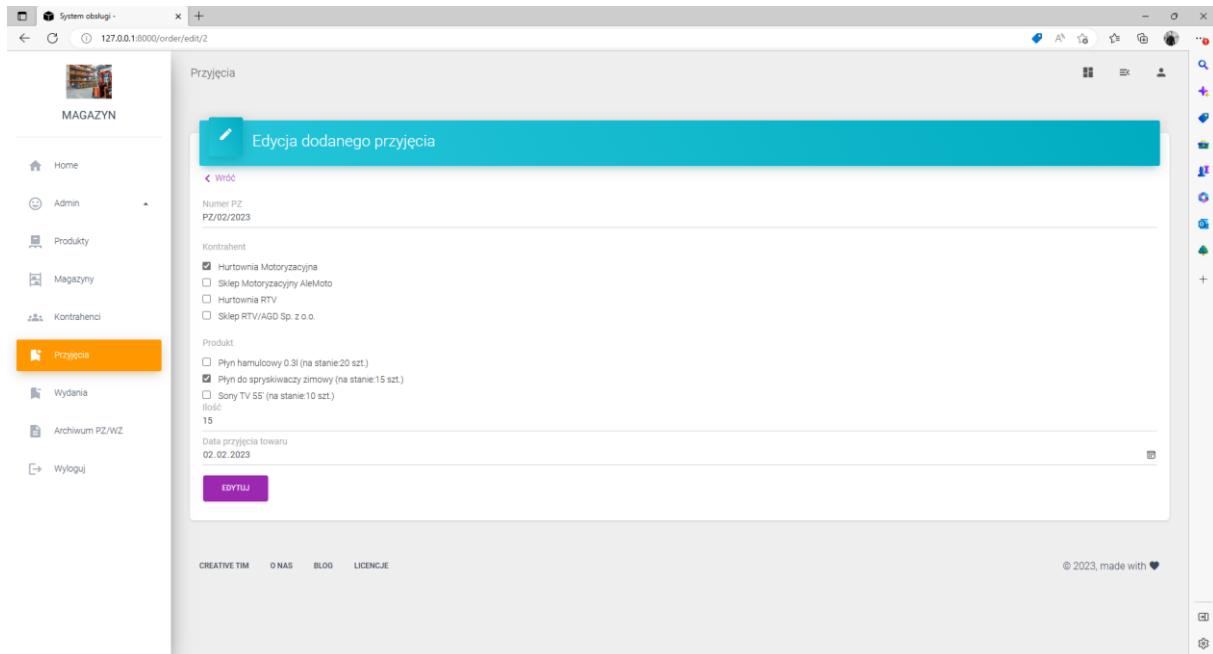
Ilość towaru (szt.)

Data przyjęcia towaru
dd.mm.rrr

DODAJ

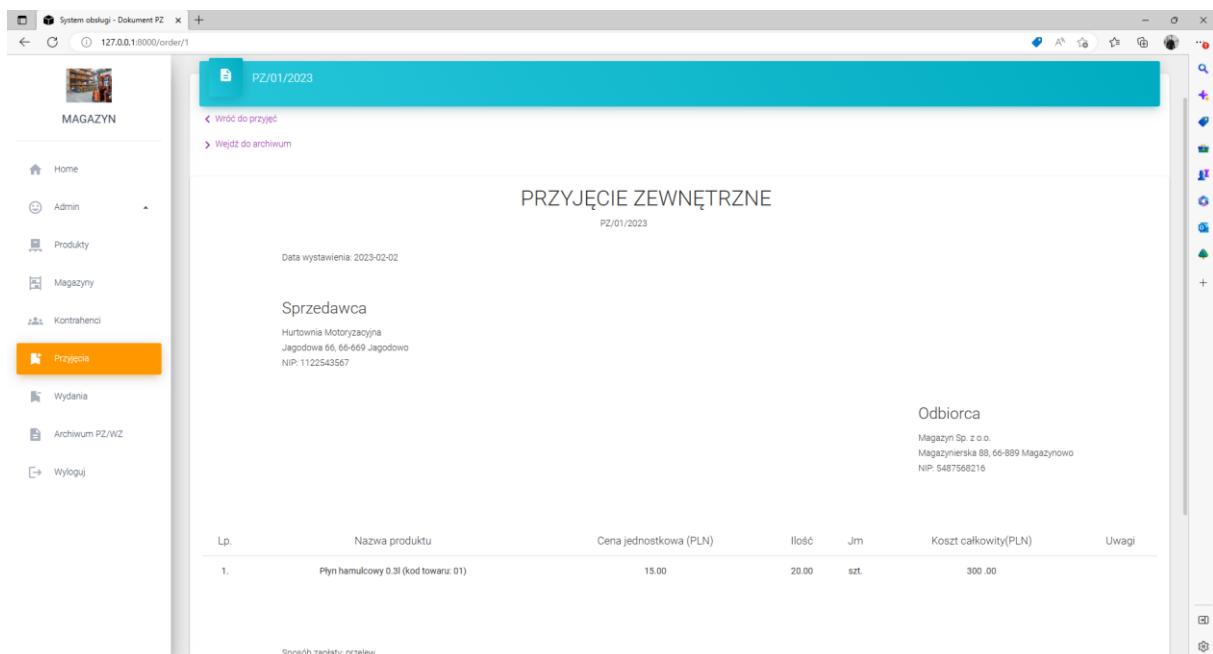
CREATIVE TIME O NAS BLOG LICENCJE © 2023, made with ❤️

Rysunek 25 - moduł 'przyjęcia' – zarejestrowanie/dodanie przyjęcia

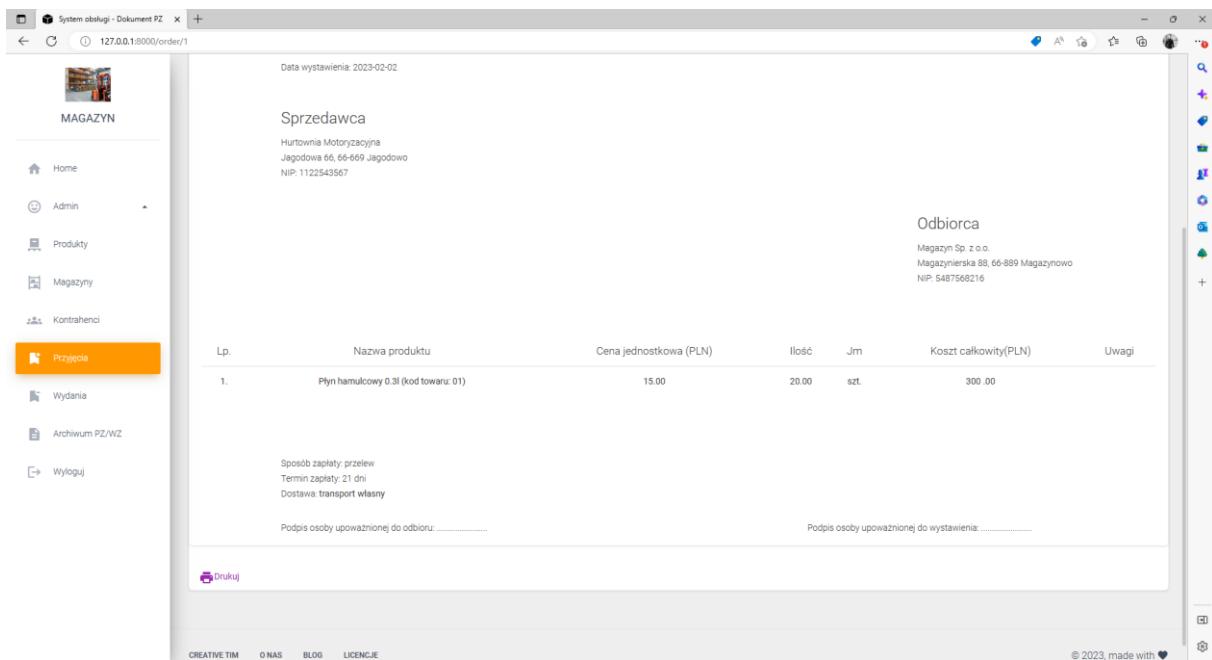


Rysunek 26 - moduł 'przyjęcia' – edycja przyjęcia

Na dole wygenerowanego dokumentu PZ znajduje się przycisk „Drukuj”, który docelowo ma wyświetlić dokument w pdf z możliwością wydrukowania lub pobrania na dysk.



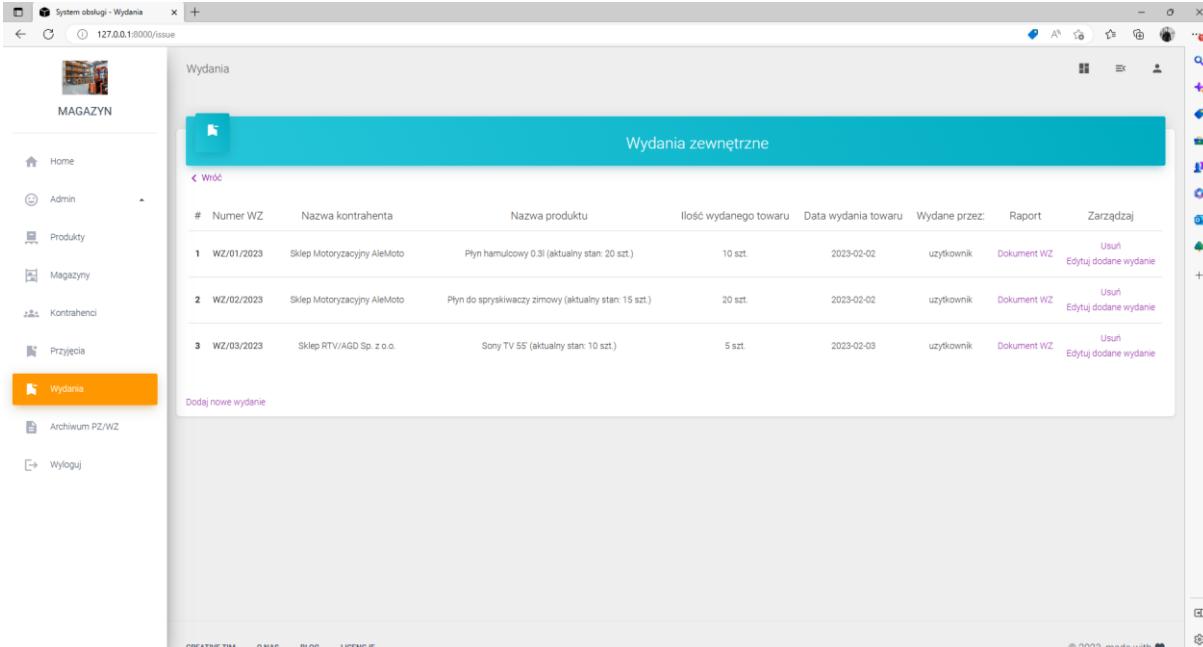
Rysunek 27 - moduł 'przyjęcia' – wygenerowanie dokumentu PZ



Rysunek 28 - moduł 'przyjęcia' – wygenerowanie dokumentu PZ cd.

Moduł 'Wydania'

Moduł odpowiada za rejestrowanie wydań. Pierwsza strona modułu zawiera listę wszystkich dodanych wydań oraz docelowo ma pokazać użytkownika, który dodał daną pozycję.

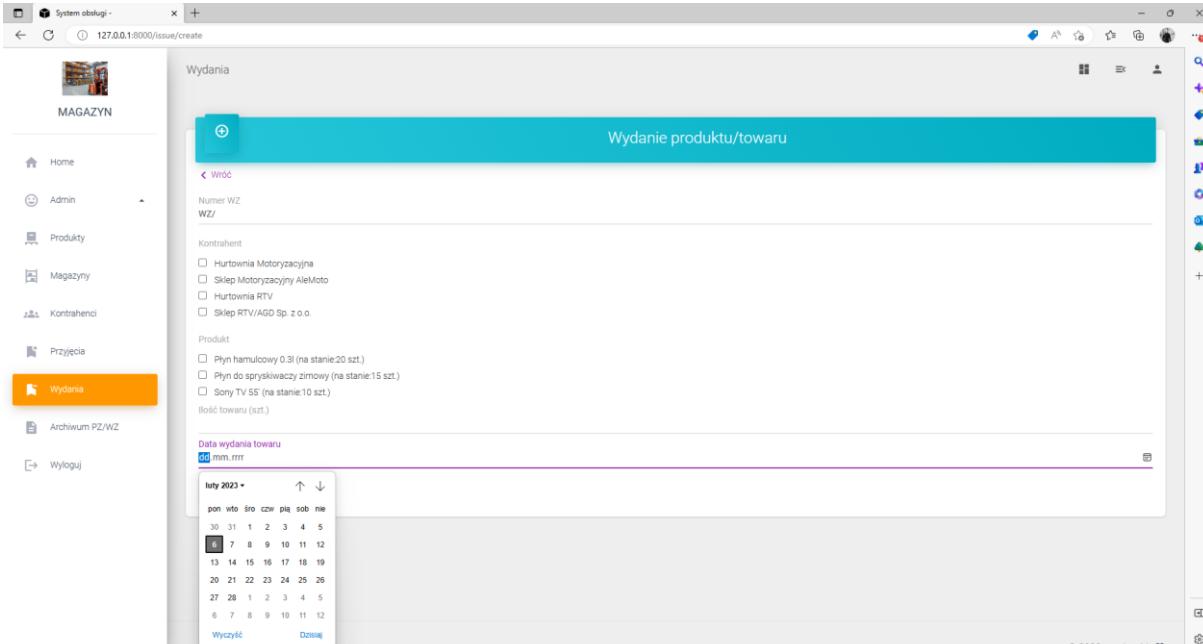


The screenshot shows a web-based application interface for managing deliveries. On the left, a sidebar menu includes links for Home, Admin, Produkty, Magazyny, Kontrahenci, Przyjacia, Wydania (which is highlighted in orange), and Archiwum PZ/WZ. Below the sidebar is a 'MAGAZYN' icon. The main content area has a header 'Wydania' and a sub-header 'Wydania zewnętrzne'. It displays a table of three delivery entries:

#	Numer WZ	Nazwa kontrahenta	Nazwa produktu	Ilość wydanego towaru	Data wydania towaru	Wydane przez:	Raport	Zarządzaj
1	WZ/01/2023	Sklep Motoryzacyjny AleMoto	Plyn hamulcowy 0.3l (aktualny stan: 20 szt.)	10 szt.	2023-02-02	użytkownik	Dokument WZ	Usuń Edytuj dodane wydanie
2	WZ/02/2023	Sklep Motoryzacyjny AleMoto	Plyn do spryskiwaczy zimowy (aktualny stan: 15 szt.)	20 szt.	2023-02-02	użytkownik	Dokument WZ	Usuń Edytuj dodane wydanie
3	WZ/03/2023	Sklep RTV/AGD Sp. z o.o.	Sony TV 55 (aktualny stan: 10 szt.)	5 szt.	2023-02-03	użytkownik	Dokument WZ	Usuń Edytuj dodane wydanie

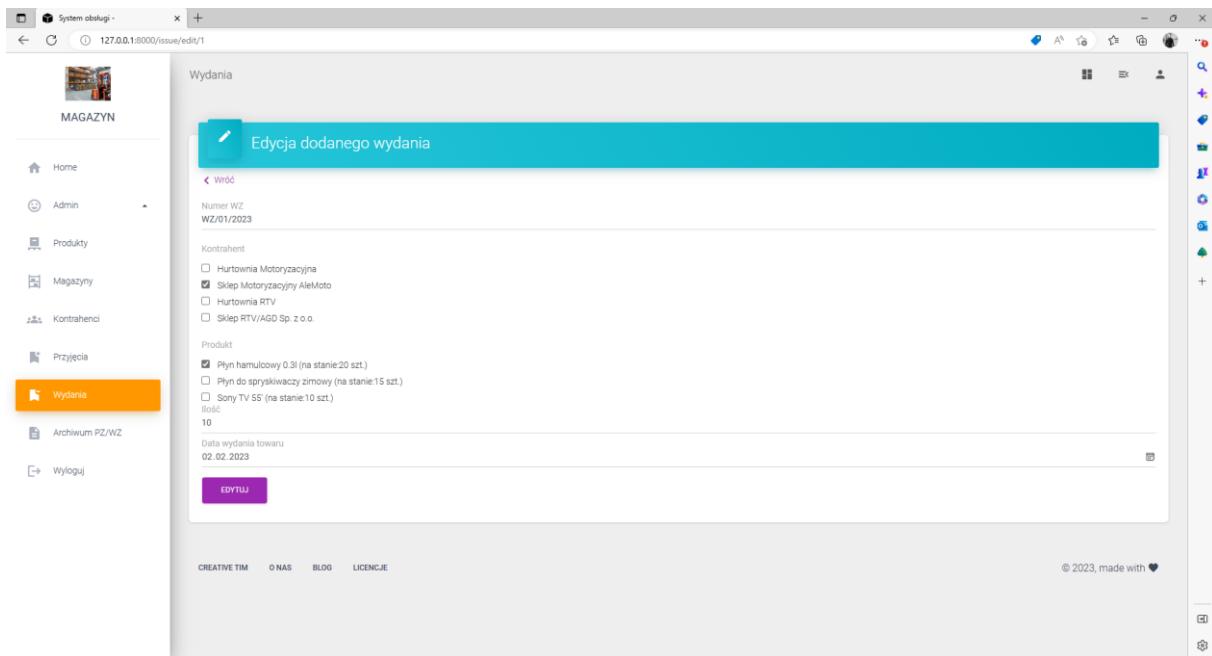
A button 'Dodaj nowe wydanie' is located at the bottom left of the table area. At the very bottom of the page, there are links for CREATIVE TIME, O NAS, BLOG, and LICENCJE, along with a copyright notice: © 2023, made with ❤️.

Rysunek 29 - moduł ‘wydania’ – lista



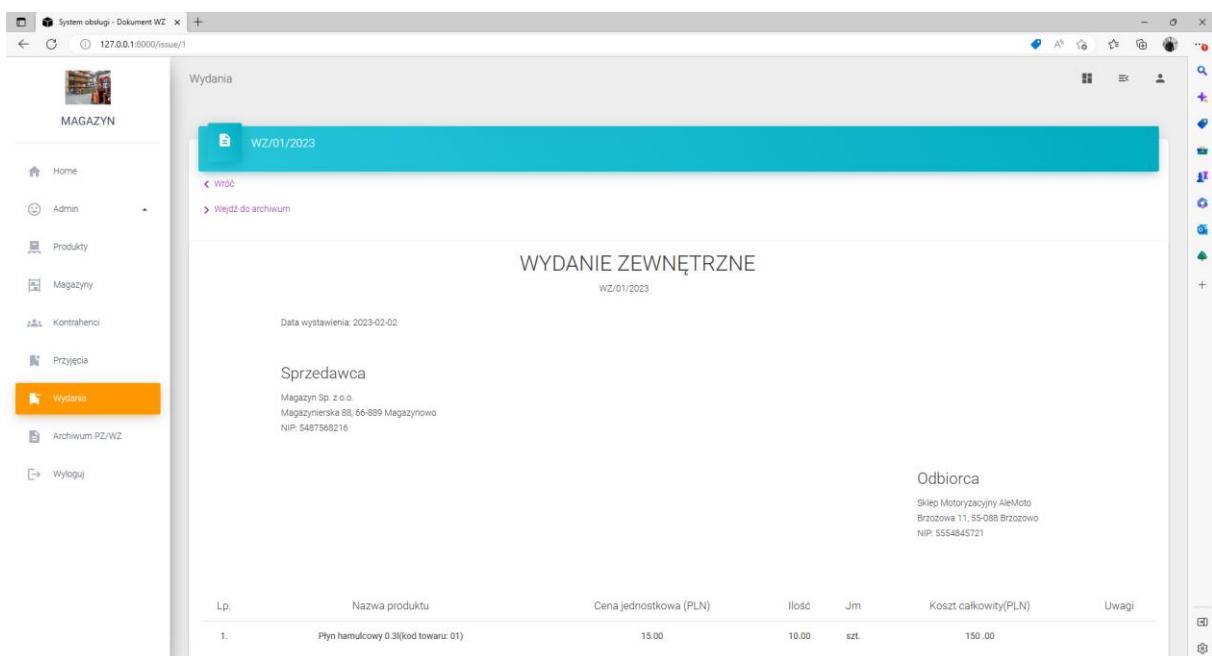
The screenshot shows a web-based application interface for creating a new delivery. On the left, a sidebar menu includes links for Home, Admin, Produkty, Magazyny, Kontrahenci, Przyjacia, Wydania (highlighted in orange), and Archiwum PZ/WZ. Below the sidebar is a 'MAGAZYN' icon. The main content area has a header 'Wydania' and a sub-header 'Wydanie produktu/towaru'. It contains fields for 'Numer WZ' (set to 'WZ/'), 'Kontрагent' (with checkboxes for Hurtownia Motoryzacyjna, Sklep Motoryzacyjny AleMoto, Hurtownia RTV, and Sklep RTV/AGD Sp. z o.o.), 'Produkt' (with checkboxes for Plyn hamulcowy 0.3l, Plyn do spryskiwaczy zimowy, and Sony TV 55), and 'Ilość towaru (szt.)' (set to '50'). Below these is a date picker set to 'Iuty 2023' with the date '2023-07-06' selected. At the bottom right, there are buttons for 'Wyszukaj' and 'Dodać'.

Rysunek 30 - moduł ‘wydania’ – zarejestrowanie/dodanie wydania

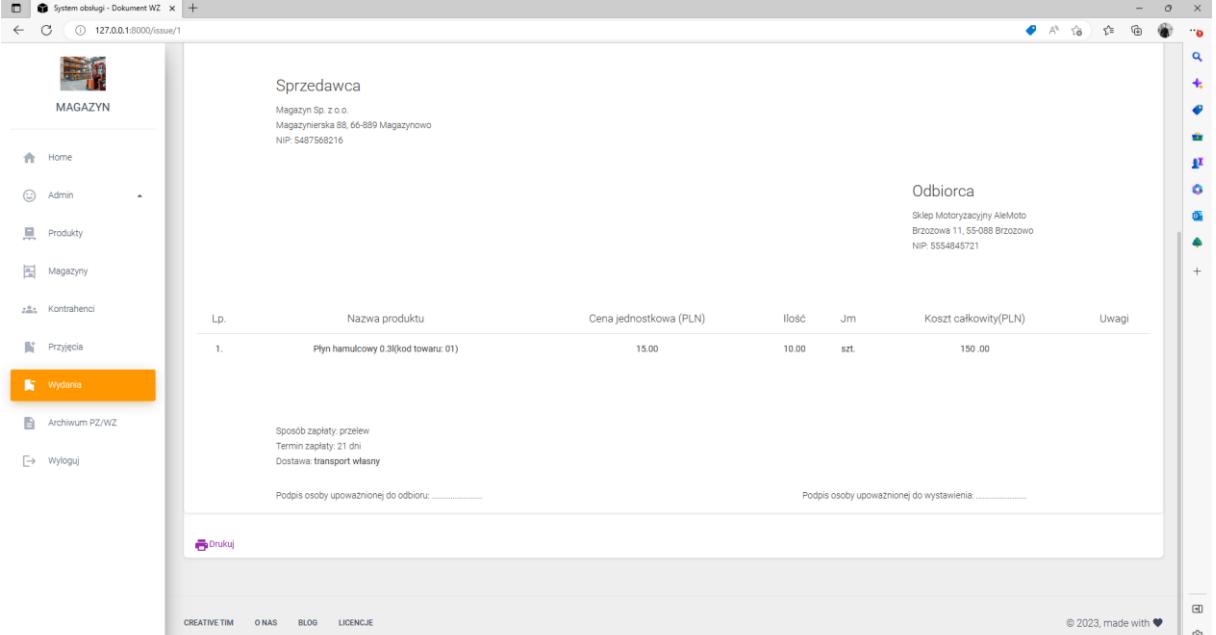


Rysunek 31 - moduł ‘wydania’ – edycja wydania

Na dole wygenerowanego dokumentu WZ znajduje się przycisk „Drukuj”, który docelowo ma wyświetlić dokument w pdf z możliwością wydrukowania lub pobrania na dysk.



Rysunek 32 - moduł ‘wydania’ – wygenerowanie dokumentu WZ



The screenshot shows a web-based application interface for managing documents. On the left, a sidebar menu includes: Home, Admin, Produkty, Magazyny, Kontrahenci, Przyjęcia, and **Wydania** (which is highlighted). Other options like Archiwum PZ/WZ and Wyloguj are also visible.

The main content area displays a document template for 'issue/1'. It includes sections for Sprzedawca (Seller) and Odbiorca (Recipient), both with address details. A table lists the product information:

Lp.	Nazwa produktu	Cena jednostkowa (PLN)	Ilość	Jm	Koszt całkowity(PLN)	Uwagi
1.	Plyn hamulcowy 0.3l(kod towaru: 01)	15.00	10.00	szt.	150.00	

Below the table, payment terms are specified: Sposób zapłaty: przelew, Termin zapłaty: 21 dni, Dostawa: transport własny. There are fields for signatures: Podpis osoby upoważnionej do odbioru: _____ and Podpis osoby upoważnionej do wystawienia: _____.

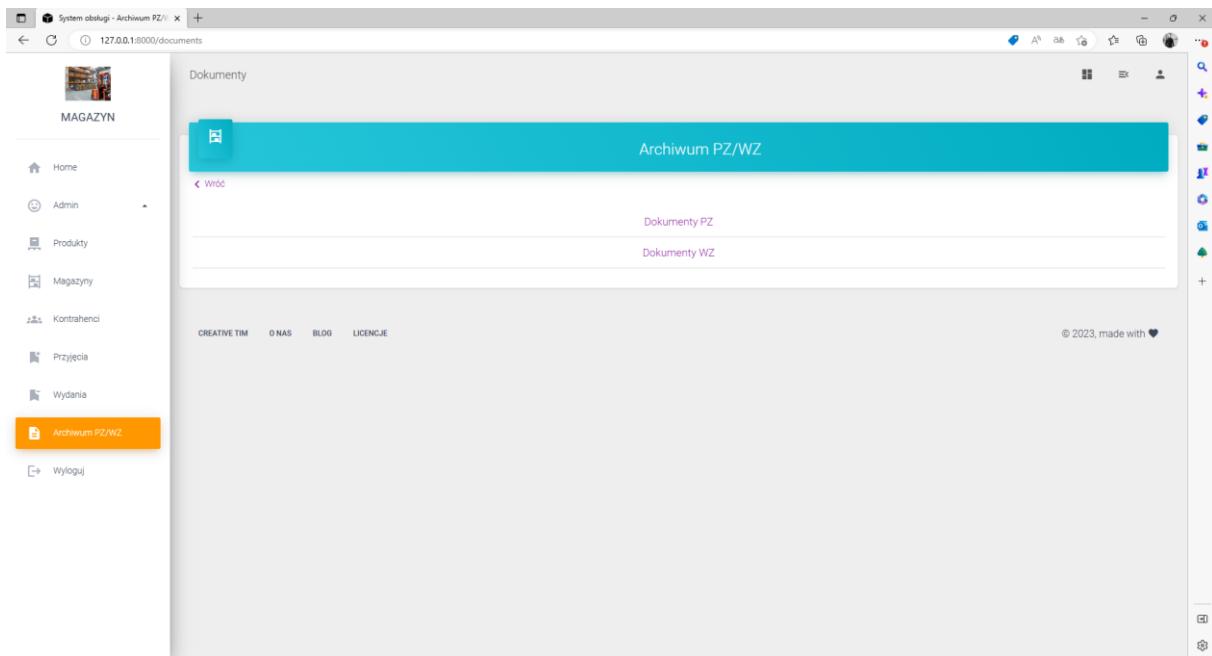
At the bottom, there's a 'Drukuj' (Print) button and footer links: CREATIVE TIM, O NAS, BLOG, LICENCJE, and a copyright notice: © 2023, made with ❤️.

Rysunek 33 - moduł ‘wydania’ – wygenerowanie dokumentu WZ cd.

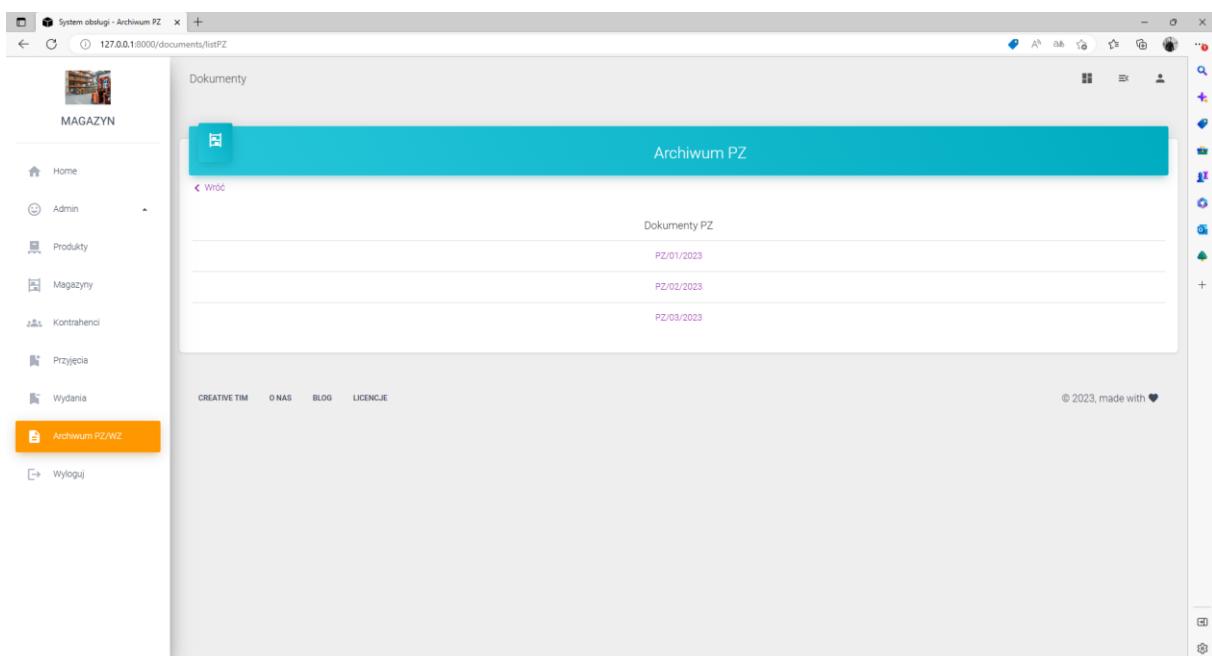
Moduł ‘Archiwum PZ/WZ’

Moduł odpowiada za zbieranie w jednym miejscu wszystkich dokumentów PZ i WZ.

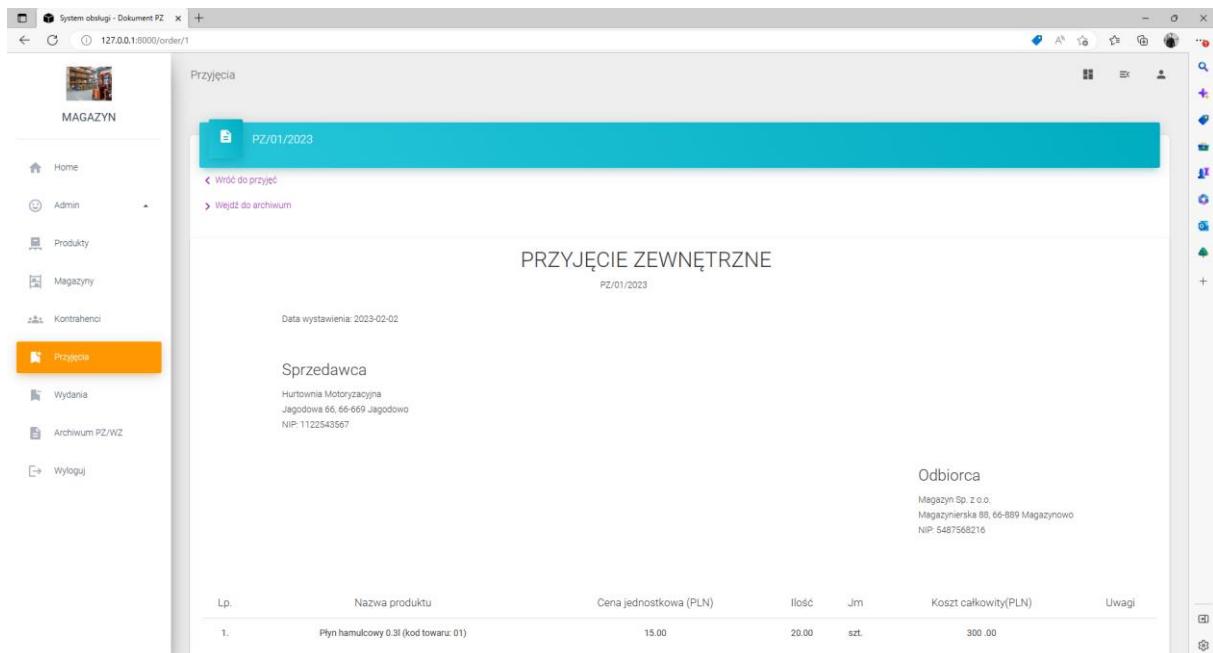
Pierwsza strona modułu zawiera listę kategorii.



Rysunek 34 - moduł ‘archiwum’ – lista kategorii

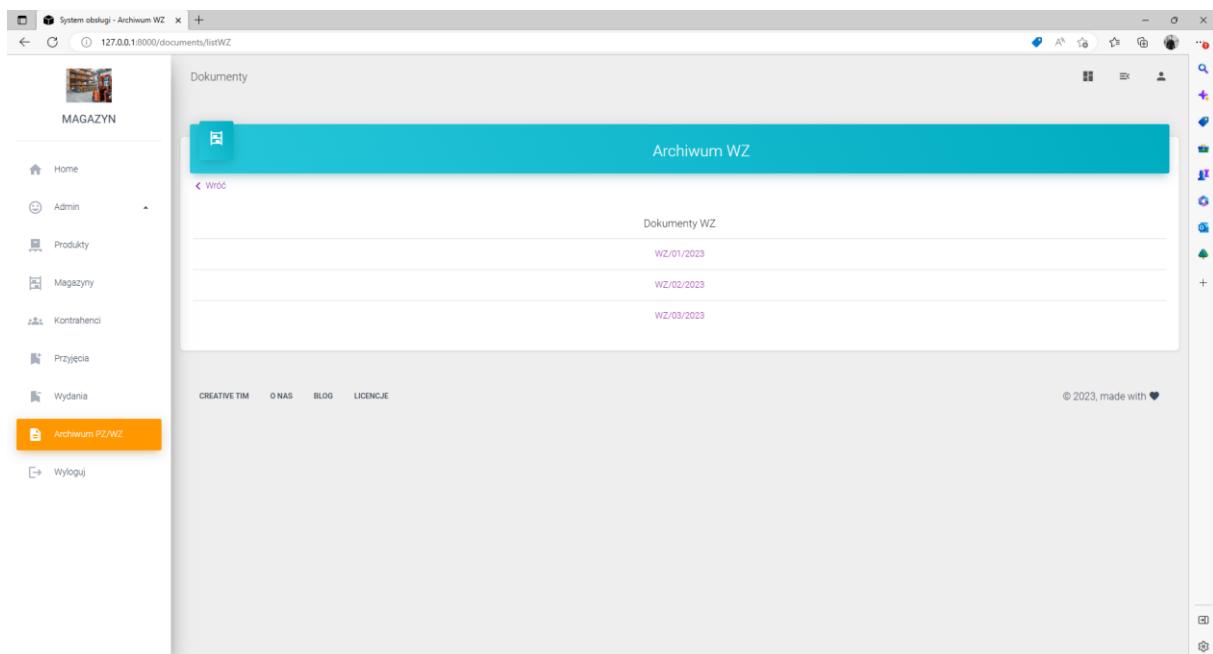


Rysunek 35 - moduł ‘archiwum’ – lista dokumentów PZ



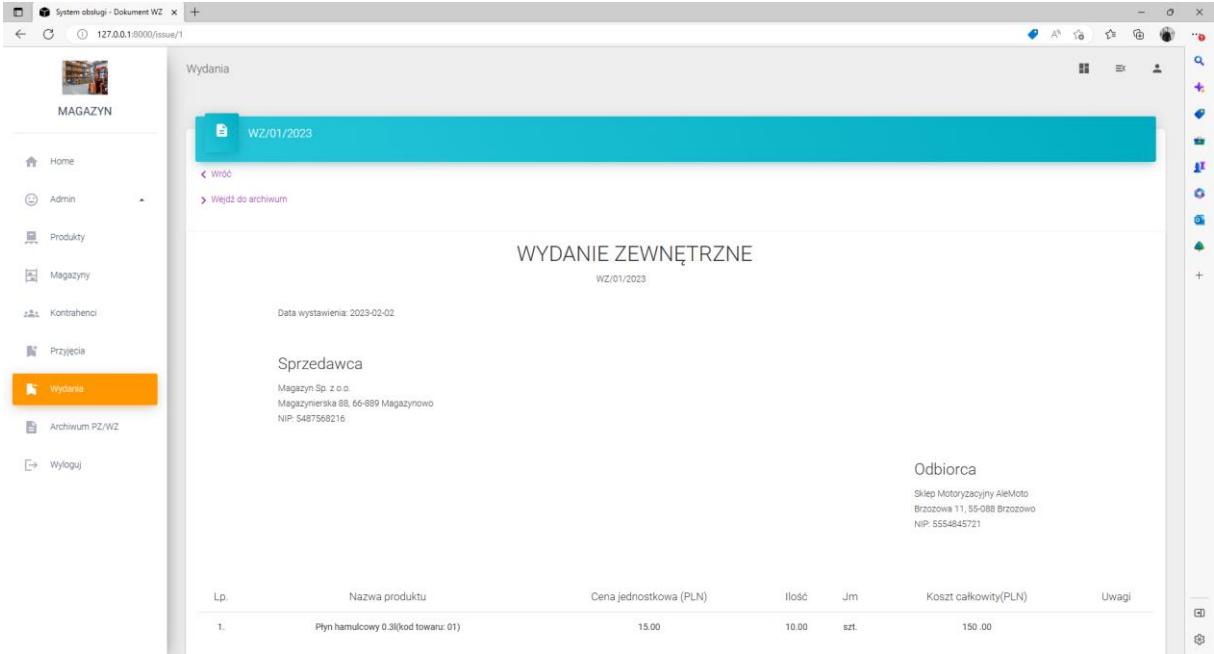
L.p.	Nazwa produktu	Cena jednostkowa (PLN)	Ilość	Jm	Koszt całkowity(PLN)	Uwagi
1.	Plyn hamulcowy 0.3l (kod towaru: 01)	15.00	20.00	szt.	300.00	

Rysunek 36 - moduł ‘archiwum’ – dokument PZ



Dokumenty
WZ/01/2023
WZ/02/2023
WZ/03/2023

Rysunek 37 - moduł ‘archiwum’ – lista dokumentów WZ



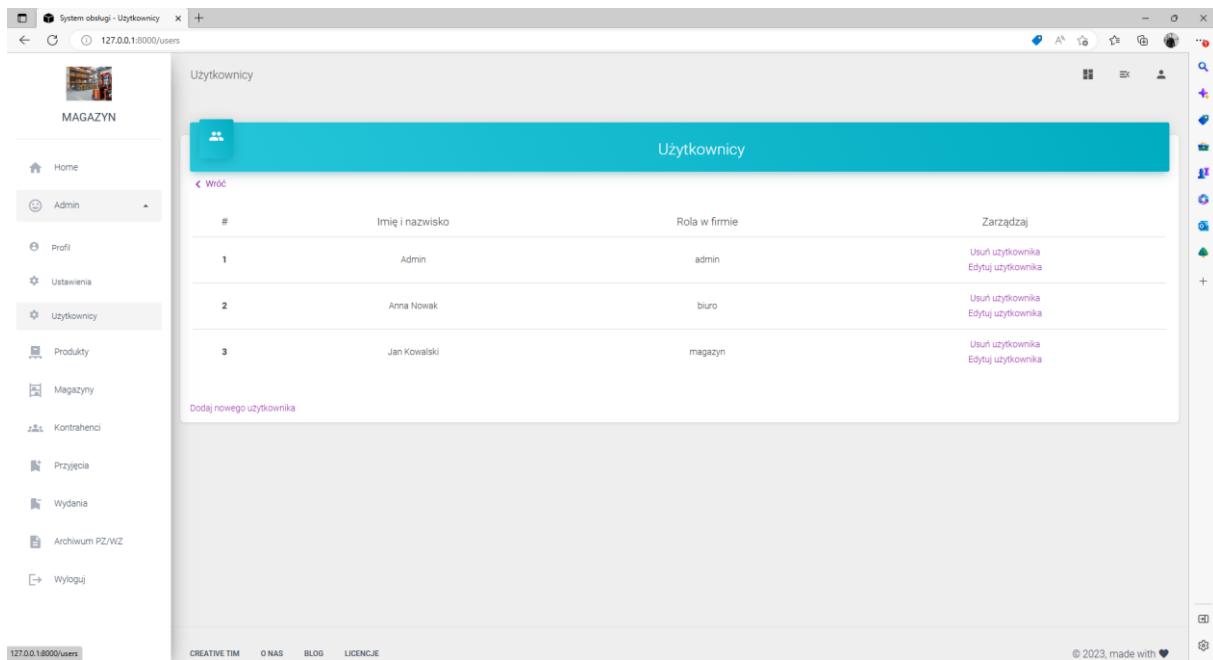
The screenshot shows a software application window titled "System obsługi - Dokument WZ". The URL in the address bar is "127.0.0.1:8000/issue/1". The left sidebar has a tree structure with nodes: MAGAZYN, Home, Admin, Produkty, Magazyny, Kontrahenci, Przyjęcia, **Wydania** (highlighted in orange), Archiwum PZ/WZ, and Wyloguj. The main content area is titled "Wydania" and shows a document titled "WZ/01/2023". Below it are links "Wróć" and "Wejdź do archiwum". The document header "WYDANIE ZEWNĘTRZNE" and date "WZ/01/2023" are displayed. The text "Data wystawienia: 2023-02-02" is present. The "Sprzedawca" section lists "Magazyn Sp. z o.o.", "Magazynierska 88, 66-889 Magazynowo", and "NIP: 5487568216". The "Odbiorca" section lists "Sklep Motoryzacyjny AleMoto", "Brzozowa 11, 55-088 Brzozowo", and "NIP: 5554845721". A table below shows the sale details:

Lp.	Nazwa produktu	Cena jednostkowa (PLN)	Ilość	Jm	Koszt całkowity(PLN)	Uwagi
1.	Plyn hamulcowy 0.3l(kod towaru: 01)	15.00	10.00	szt.	150.00	

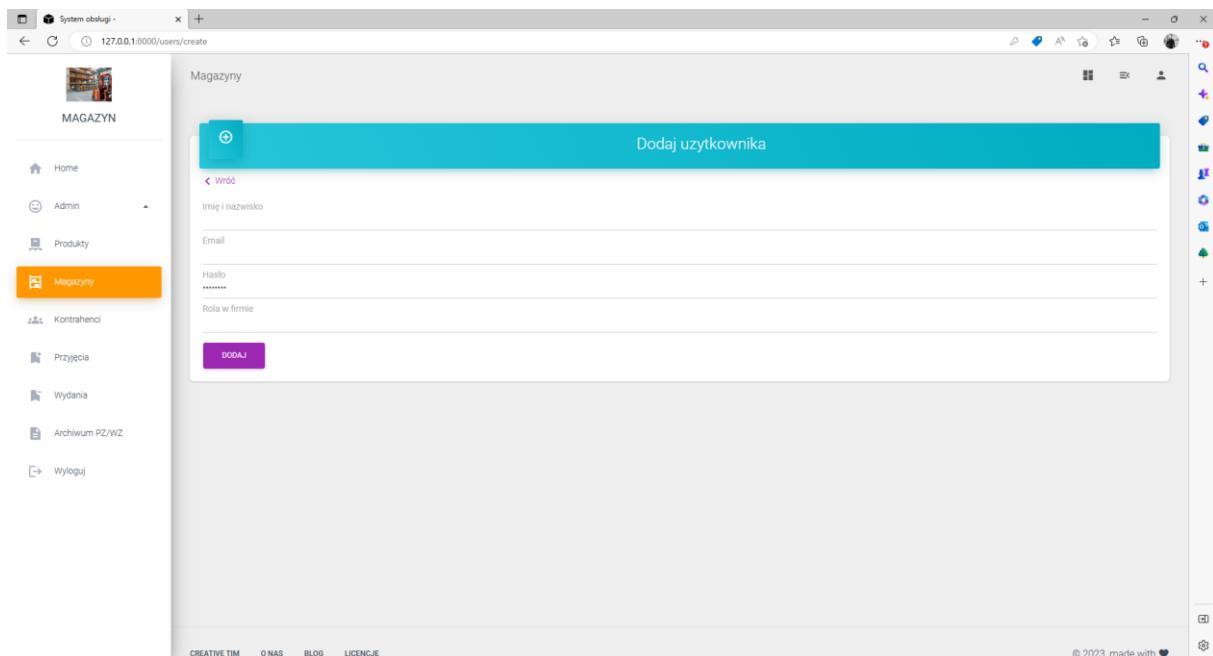
Rysunek 38 - moduł ‘archiwum’ - dokument WZ

Moduł ‘Użytkownicy’

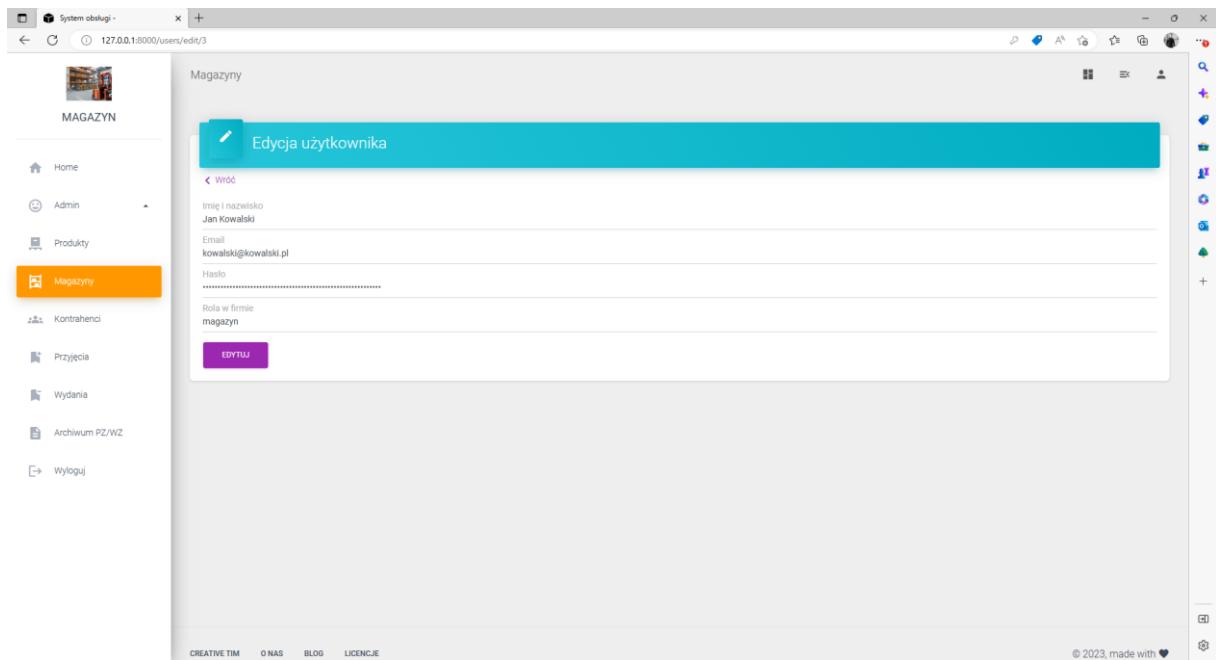
Moduł odpowiada za zarządzanie użytkownikami. Dodawanie, edytowanie, usuwanie. Do tego modułu dostęp ma tylko i wyłącznie użytkownik z rolą ‘admina’. Moduł dostępny po kliknięciu w nazwę użytkownika po prawej stronie na ‘sidebarze’.



Rysunek 39 - moduł ‘użytkownicy’ - lista

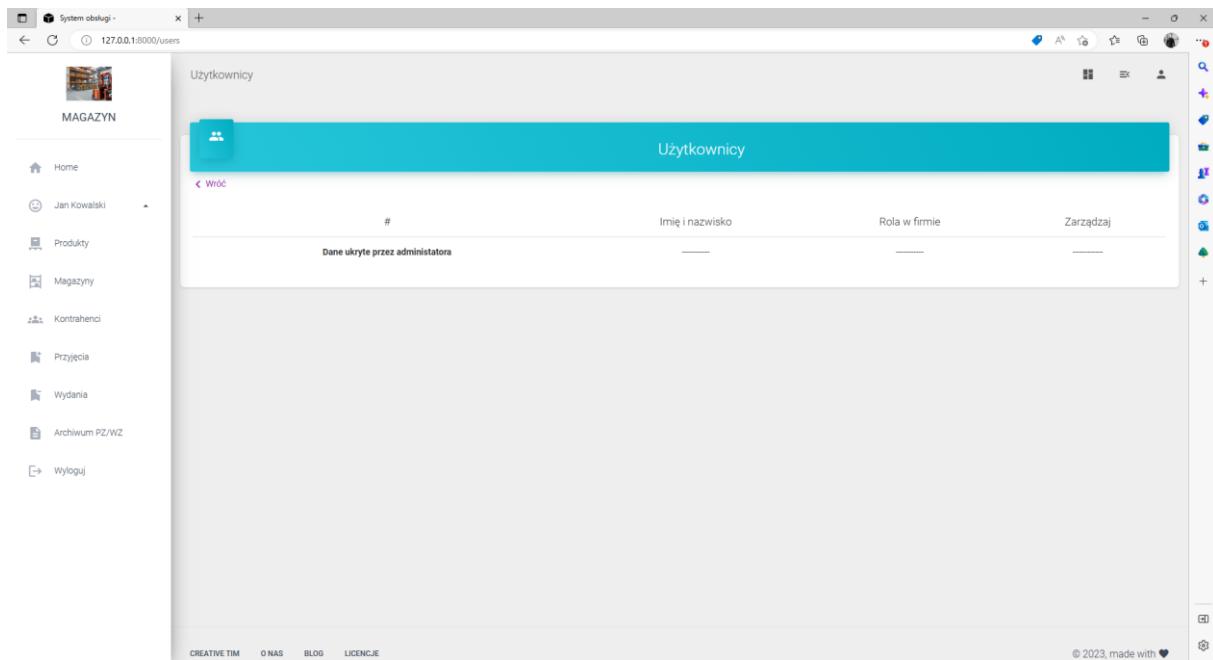


Rysunek 40 - moduł ‘użytkownicy’ – dodaj użytkownika

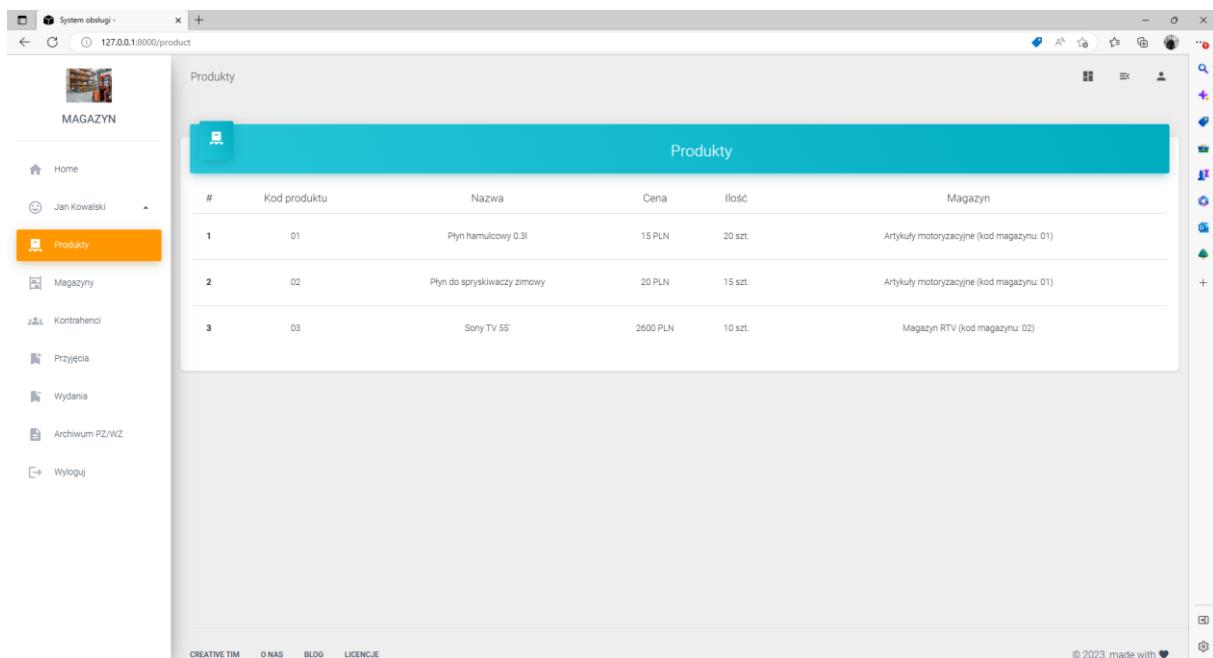


Rysunek 41 - moduł ‘użytkownicy’ – edytuj użytkownika

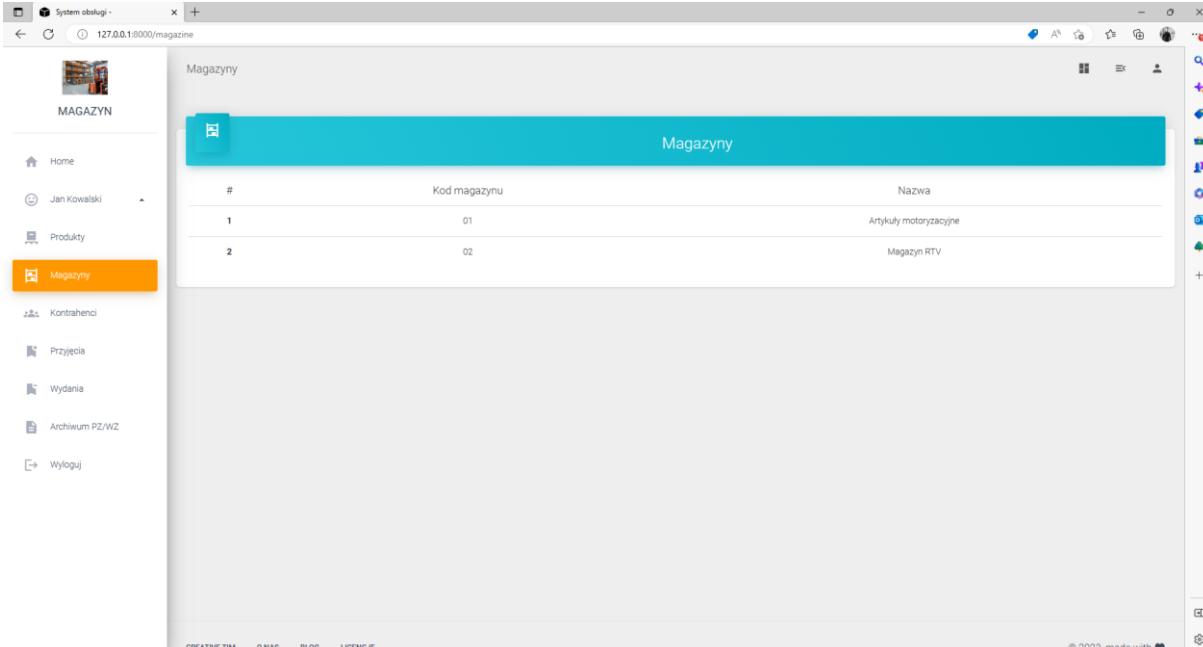
Użytkownik, który pełni rolę magazyniera w firmie nie ma dostępu do zaawansowanych opcji zarządzania poszczególnymi modułami: użytkownicy (dane ukryte), produkty (tylko lista), magazyny (tylko lista), kontrahenci (tylko lista – okrojona). Ma natomiast pełen dostęp do modułów przyjęć i wydań. Użytkownik pełniący rolę pracownika biurowego ma pełen dostęp do wszystkich modułów oprócz modułu zarządzania użytkownikami.



Rysunek 42 - moduł ‘użytkownicy’ – widok listy z poziomu użytkownika ‘magazyniera’



Rysunek 43 - moduł ‘produkty’ – widok listy z poziomu użytkownika ‘magazyniera’

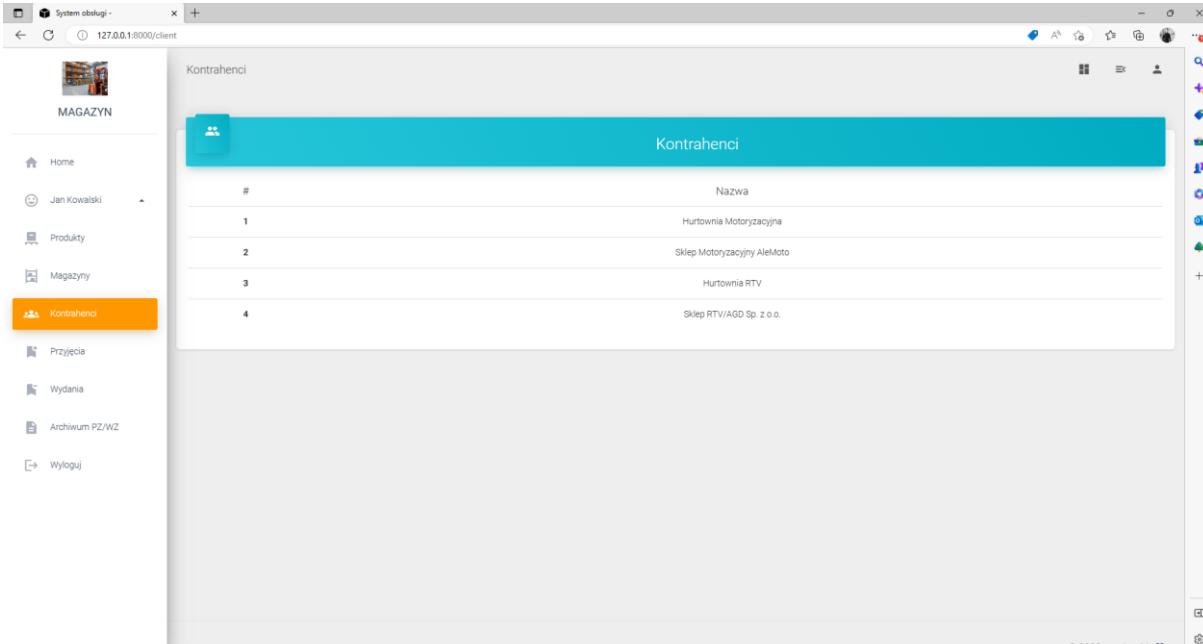


The screenshot shows a web-based application interface for managing a warehouse system. The top navigation bar includes links for 'System obsługi', 'Home', 'Jan Kowalski', 'Produkty', 'Magazyny' (which is highlighted in orange), 'Kontrahenci', 'Przyjęcia', 'Wydania', 'Archiwum PZ/WZ', and 'Wyloguj'. The main content area is titled 'Magazyny' and displays a table with two rows of data:

#	Kod magazynu	Nazwa
1	01	Artykuły motoryzacyjne
2	02	Magazyn RTV

At the bottom of the page, there are links for 'CREATIVE TIM', 'O NAS', 'BLOG', and 'LICENCJE', along with a copyright notice: '© 2023, made with ❤'.

Rysunek 44 - moduł ‘magazyny’ – widok listy z poziomu użytkownika ‘magazyniera’



The screenshot shows a web-based application interface for managing supplier information. The left sidebar menu is identical to the one in the previous screenshot. The main content area is titled 'Kontrahenci' and displays a table with four rows of data:

#	Nazwa
1	Hurtownia Motoryzacyjna
2	Sklep Motoryzacyjny AleMoto
3	Hurtownia RTV
4	Sklep RTV/AGD Sp. z o.o.

At the bottom of the page, there are links for 'CREATIVE TIM', 'O NAS', 'BLOG', and 'LICENCJE', along with a copyright notice: '© 2023, made with ❤'.

Rysunek 43 - moduł ‘kontrahenci’ – widok listy z poziomu użytkownika ‘magazyniera’

Dane techniczne

Serwer WWW:

- Apache/2.4.52 (Win64) OpenSSL/1.1.1m PHP/8.0.25
- Wersja klienta bazy danych: libmysql - mysqlnd 8.0.25
- Rozszerzenie PHP: mysqli curl mbstring
- Wersja PHP: 8.0.25

Framework Laravel:

- Laravel 8.73.2

Szablon aplikacji:

- Material Dashboard Bootstrap

Linki do dokumentacji:

- PHP: <https://www.php.net/manual/en/index.php>
- Laravel 8: <https://laravel.com/docs/8.x/installation>
- Material Dashboard Bootstrap: <https://material-dashboard-laravel.creative-tim.com/documentation/getting-started/overview.html>

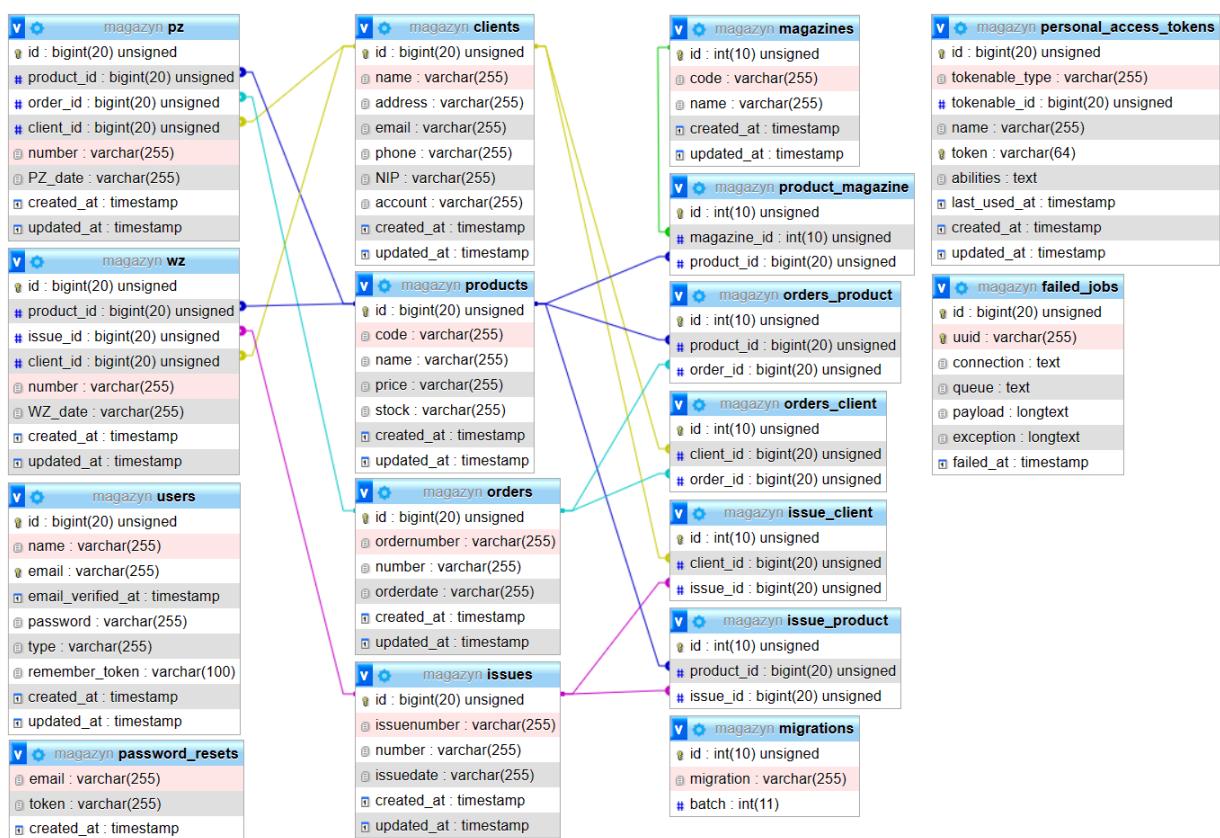
Baza danych

Serwer bazy danych:

- Serwer: 127.0.0.1 via TCP/IP
- Typ serwera: MariaDB
- Połaczenie z serwerem: SSL nie jest używany
- Wersja serwera: 10.4.22-MariaDB - mariadb.org binary distribution
- Wersja protokołu: 10
- Użytkownik: root@localhost

- Kodowanie znaków serwera: UTF-8 Unicode (utf8mb4)

Schemat bazy danych



Rysunek 44 - schemat bazy danych

Struktura poszczególnych tabel

Tabela 1.

Tabela failed_jobs

Kolumna	Typ	Atrybuty	Nu ll	Ustawienia domyślne	Dodatkowo	Odsyłacze do
id	bigint(20)	UNSIGNED	Nie		auto_increm ent	
uuid	varchar(25 5)		Nie			
connecti on	text		Nie			
queue	text		Nie			
payload	longtext		Nie			
exceptio n	longtext		Nie			
failed_at	timestamp		Nie	current_tim estamp()		

Tabela 2.

Tabela migrations

Kolumna	Typ	Atrybuty	Nu ll	Ustawien ia domyśln e	Dodatkowo	Odsyłacze do
id	int(10)	UNSIGNED	Nie		auto_increme nt	
migration	varchar(255)		Nie			
batch	int(11)		Nie			

Tabela 3.

Tabela password_reset

Kolumna	Typ	Atrybuty	Nu ll	Ustawienia domyślne	Dodatkowo	Odsyłacze do
email	varchar(255)		Nie			
token	varchar(255)		Nie			
created_at	timestamp		Tak	NULL		

Tabela 4.
Tabela personal_access_tokens

Kolumna	Typ	Atrybuty	Nul l	Ustawieni a domyślne	Dodatkowo	Odsyłacze do
id	bigint(20)	UNSIGNED	Nie		auto_incremente	
tokenable_type	varchar(255)		Nie			
tokenable_id	bigint(20)	UNSIGNED	Nie			
name	varchar(255)		Nie			
token	varchar(64)		Nie			
abilities	text		Tak	NULL		
last_used_at	timestamp		Tak	NULL		
created_at	timestamp		Tak	NULL		
updated_at	timestamp		Tak	NULL		

Tabela 5.

Tabela users

Kolumna	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatkowo	Odsyłacze do
id	bigint(20)	UNSIGNED	Nie		auto_increment	
name	varchar(255)		Nie			
email	varchar(255)		Nie			
email_verified_at	timestamp		Tak	NULL		
password	varchar(255)		Nie			
remember_token	varchar(100)		Tak	NULL		
created_at	timestamp		Tak	NULL		

updated_at	timestamp		T a k	NULL		
type	varchar(10)		N ie			

Tabela 6.

Tabela magazines

Kolumna	Typ	Atrybuty	Nu ll	Ustawienia domyślne	Dodatkowo	Odsyłacze do
id	int(10)	UNSIGNED	N ie		auto_increm ent	
code	varchar(2 55)		N ie			
name	varchar(2 55)		N ie			
created_at	timestam p		T ak	NULL		
updated_at	Timestamp		T ak	NULL		

Tabela7.

Tabela products

Kolumna	Typ	Atrybuty	Nu ll	Ustawien ia domyślne	Dodatkow o	Odsyłacze do
id	bigint(2 0)	UNSIG NED	Ni e		auto_incre ment	
Code	varchar(255)		Ni e			
name	varchar(255)		Ni e			
price	varchar(255)		Ni e			
stock	varchar(255)		Ni e			
created_at	timesta mp		Ta k	NULL		
updated_at	timesta mp		Ta k	NULL		

Tabela 8.
Tabela clients

Kolumna	Typ	Atrybuty	N ull	Ustawien ia domyślne	Doda tkow o	Odsyłacze do

id	int(10)	UNSIGNED	Nie		auto_incre ment	
name	varchar(255)		Nie			
address	varchar(255)		Nie			
email	varchar(255)		Nie			
phone	varchar(255)		Nie			
NIP	varchar(255)		Tak	Null		
account	varchar(255)		Nie			
created_at	timestam		Tak	Null		
updated_a t	timestam		Tak	Null		

Tabela 9.

Tabela orders

Kolumna	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatakowo	Odsyłacze do
id	Bigint(20)	UNSIGNED ED	Nie		auto_incre ment	
ordernumber	varchar(255)		Nie			
number	varchar(255)		Nie			
Orderdate	varchar(255)		Nie			
created_at	timesta mp		Tak	Null		
update_at	timesta mp		Tak	Null		

Tabela 10.

Tabela product_magazine

Kolumna	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatakowo	Odsyłacze do
id	int(10)	UNSIGNED	Nie		auto_increment	
magazine_id	int(10)	UNSIGNED	Nie			magazines.id ON UPDATE RESTRICT ON DELETE CASCADE
product_id	bigint(20)	UNSIGNED	Nie			products.id ON UPDATE RESTRICT ON DELETE CASCADE

Tabela 11.

Tabela orders_client

Kolumna	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatakowo	Odsyłacze do
id	int(10)	UNSIGNED	Nie		auto_increment	
client_id	bigint(20)	UNSIGNED	Nie			clients.id ON UPDATE RESTRICT ON DELETE CASCADE
order_id	bigint(20)	UNSIGNED	Nie			orders.id ON UPDATE RESTRICT ON DELETE CASCADE

Tabela 12.

Tabela orders_product

Kolumna	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatakowo	Odsyłacze do
id	int(10)	UNSIGNED	Nie		auto_increment	
product_id	bigint(20)	UNSIGNED	Nie			products.id ON UPDATE RESTRICT ON DELETE CASCADE
order_id	bigint(20)	UNSIGNED	Nie			orders.id ON UPDATE RESTRICT ON DELETE CASCADE

Tabela 13.

Tabela issues

Kolumna	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatakowo	Odsyłacze do
id	int(10)	UNSIGNED	Nie		auto_increment	
issuenumber	varchar(255)		Nie			

number	varchar(255)		Nie			
issuedate	varchar(255)		Nie			
created_at	timestamp		Tak	Null		
updated_at	timestamp		Tak	Null		

Tabela 14.
Tabela issue_client

Kolumna	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatkowo	Odsyłacze do
id	int(10)	UNSIGNED	Nie		auto_increment	
client_id	bigint(20)	UNSIGNED	Nie			clients.id ON UPDATE RESTRICT ON DELETE CASCADE
issue_id	bigint(20)	UNSIGNED	Nie			issues.id ON UPDATE RESTRICT ON DELETE CASCADE

Tabela 15.

Tabela issue_product

Kolumna	Typ	Atrybuty	N ull	Ustawien ia domyślne	Doda tkow o	Odsyłacze do
id	int(10)	UNSIGNED	Nie		auto_incre ment	
product_id	bigint(2 0)	UNSIGNED	Nie			products.id ON UPDATE RESTRICT ON DELETE CASCADE
issue_id	bigint(2 0)	UNSIGNED	Nie			issues.id ON UPDATE RESTRICT ON DELETE CASCADE

Tabela 16.

Tabela pz

Kolumna	Typ	Atrybuty	N ull	Ustawien ia domyślne	Doda tkow o	Odsyłacze do
id	int(10)	UNSIGNED	Nie		auto_incre ment	
product_id	bigint(2 0)	UNSIGNED	Nie			products.id ON UPDATE RESTRICT ON DELETE CASCADE

order_id	bigint(20)	UNSIGNED	Nie			orders.id ON UPDATE RESTRICT ON DELETE CASCADE
client_id	bigint(20)	UNSIGNED	Nie			clients.id ON UPDATE RESTRICT ON DELETE CASCADE
number	varchar(255)		Nie			
PZ_date	varchar(255)		Nie			

Tabela 17.
Tabela wz

Kolumna	Typ	Atrybuty	Null	Ustawienia domyślne	Dodatakowo	Odsyłacze do
id	int(10)	UNSIGNED	Nie		auto_increment	
product_id	int(10)	UNSIGNED	Nie			products.id ON UPDATE RESTRICT ON DELETE CASCADE
issue_id	bigint(20)	UNSIGNED	Nie			issues.id ON UPDATE RESTRICT ON DELETE CASCADE
client_id	bigint(20)	UNSIGNED	Nie			clients.id ON UPDATE RESTRICT ON DELETE CASCADE

number	varchar(255)		Nie			
WZ_date	varchar(255)		Nie			

Kod źródłowy

Migracje

Migracje opisują bazę danych i pozwalają na zdefiniowanie bazy danych aplikacji. Framework Laravel posiada wbudowane statyczne fasady, które można opcjonalnie wykorzystać. Jedną z nich jest Schema – wykorzystywana na potrzeby definiowania bazy danych.

Każda z migracji składa się z:

Klasy

Nazwy klas pochodzą od poszczególnych funkcjonalności, np. CreateProductTable.

Metod

Klasa migracji składa się z dwóch głównych metod. Metody up i metody down.

Metoda up jest używana do dodawania nowej tabeli, kolumny lub indeksu w bazie danych. Z kolei metoda down jest odwrotnością działania metody up. W ramach fasady Schema używane są metody: create, table, dropIfExists. Create odpowiada za stworzenie tabeli, table za aktualizację tabeli, dropIfExists za usuwanie tabeli.

Zmiennych

Zmienna \$table przyjmuje nazwy poszczególnych kolumn.

Poniżej przedstawiony został kod źródłowy utworzonych migracji.

Tworzenie tabeli products

```
2022_12_17_232838_create_product_table.php

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateProductTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('products',function(Blueprint $table){ //stworzenie tabeli
17             $table->id();
18             $table->string('name');
19             $table->string('price');
20             $table->string('stock');
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::dropIfExists('products');
33     }
34 }
35 }
```

Rysunek 45 - migracje, tworzenie tabeli 'products'

Tworzenie tabeli magazines

```
2022_12_18_004327_create_magazines_table.... | 1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateMagazinesTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('magazines', function (Blueprint $table) { //stworzenie tabeli
17             $table->increments('id');
18             $table->string('name');
19             $table->timestamps();
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      *
26      * @return void
27      */
28     public function down()
29     {
30         Schema::dropIfExists('magazines');
31     }
32 }
33
```

Rysunek 46- migracje, tworzenie tabeli 'magazines'

Tworzenie tabeli product_magazine

W ramach fasady Schema, metoda table aktualizuje tabelę product_magazine o klucze obce.

```

2022_12_18_010935_create_product_magazin...
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateProductMagazineTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::dropIfExists('product_magazine');
17         Schema::create('product_magazine', function (Blueprint $table) {           //stworzenie tabeli pivot
18             $table->increments('id');
19             $table->unsignedInteger('magazine_id')->unsigned();
20             $table->unsignedBigInteger('product_id')->unsigned();
21         });
22         Schema::table('product_magazine', function(Blueprint $table) {           //nadanie kluczy obcych
23             $table->foreign('magazine_id')->references('id')->on('magazines')->onDelete('cascade');
24             $table->foreign('product_id')->references('id')->on('products')->onDelete('cascade');
25         });
26     }
27
28
29     /**
30      * Reverse the migrations.
31      *
32      * @return void
33      */
34     public function down()
35     {
36         Schema::dropIfExists('product_magazine');
37     }
38 }
39

```

Rysunek 47- migracje, tworzenie tabeli 'product_magazine'

Reszta tabel bazy danych została stworzona w analogiczny sposób. Kod różni się jedynie nazwami tabel, poszczególnych pól oraz relacjami.

Seeding

Seeder składa się z:

Klasy - UserTableSeeder

Metod:

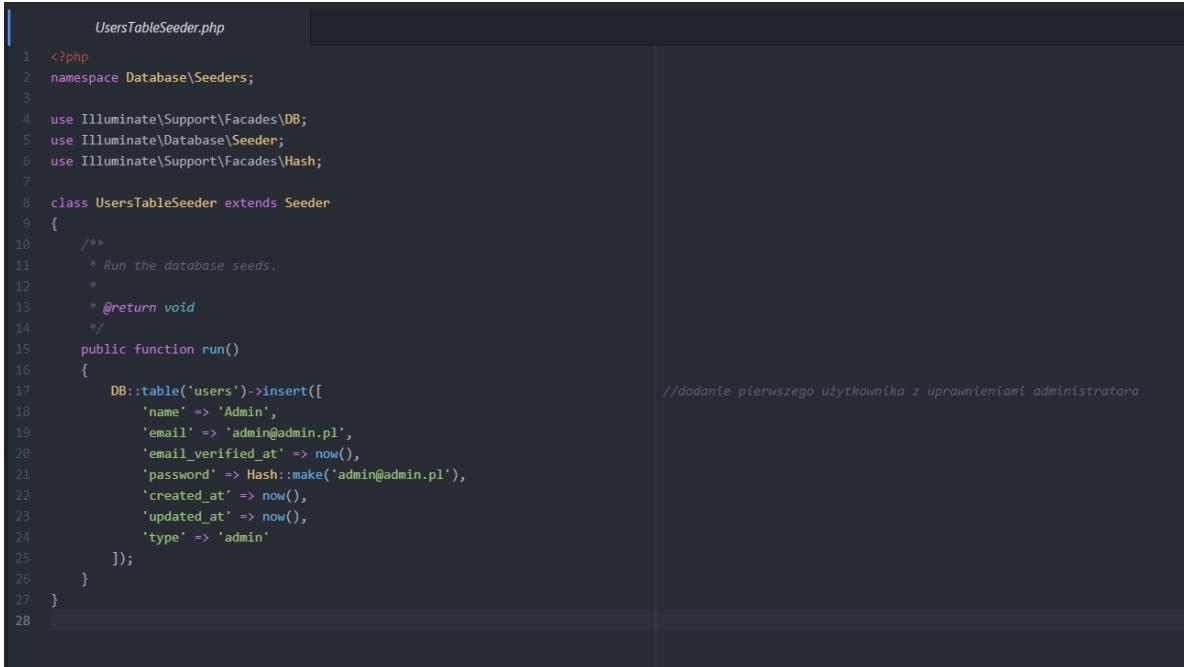
run – wywoływana podczas wykonywania komendy php artisan db:seed
 table – w ramach fasady DB, aktualizuje stan tabeli dodając do niej przypisane wartości

Poniżej znajduje się kod źródłowy wykorzystywanego seedera.

UserTableSeeder.php

Statycznie zapisane parametry poszczególnych pól tabeli users, takich jak:

- name
- email
- email_verified_at
- password
- created_at
- update_at
- type



```

1  <?php
2  namespace Database\Seeders;
3
4  use Illuminate\Support\Facades\DB;
5  use Illuminate\Database\Seeder;
6  use Illuminate\Support\Facades\Hash;
7
8  class UsersTableSeeder extends Seeder
9  {
10     /**
11      * Run the database seeds.
12      *
13      * @return void
14      */
15     public function run()
16     {
17         DB::table('users')->insert([
18             'name' => 'Admin',
19             'email' => 'admin@admin.pl',
20             'email_verified_at' => now(),
21             'password' => Hash::make('admin@admin.pl'),
22             'created_at' => now(),
23             'updated_at' => now(),
24             'type' => 'admin'
25         ]);
26     }
27 }
28

```

//dodanie pierwszego użytkownika z uprawnieniami administratora

Rysunek 48- seeders

Modele

Modele generowane są poprzez wykonanie polecenia w wierszu poleceń:
 php artisan make:model nazwamodelu

Struktura modelu:

Klasy - o ścisłe określonej nazwie nawiązującej do poszczególnej tabeli bazy danych, są rozszerzeniem podstawowej klasy Model

Właściwości klasy – zmienna tablicowa \$fillable zawierająca wartości, które należało wypełnić w bazie danych metodą create, domyślne zabezpieczenie wymagane przez Eloquent

Poniżej znajduje się kod źródłowy wygenerowanych modeli.

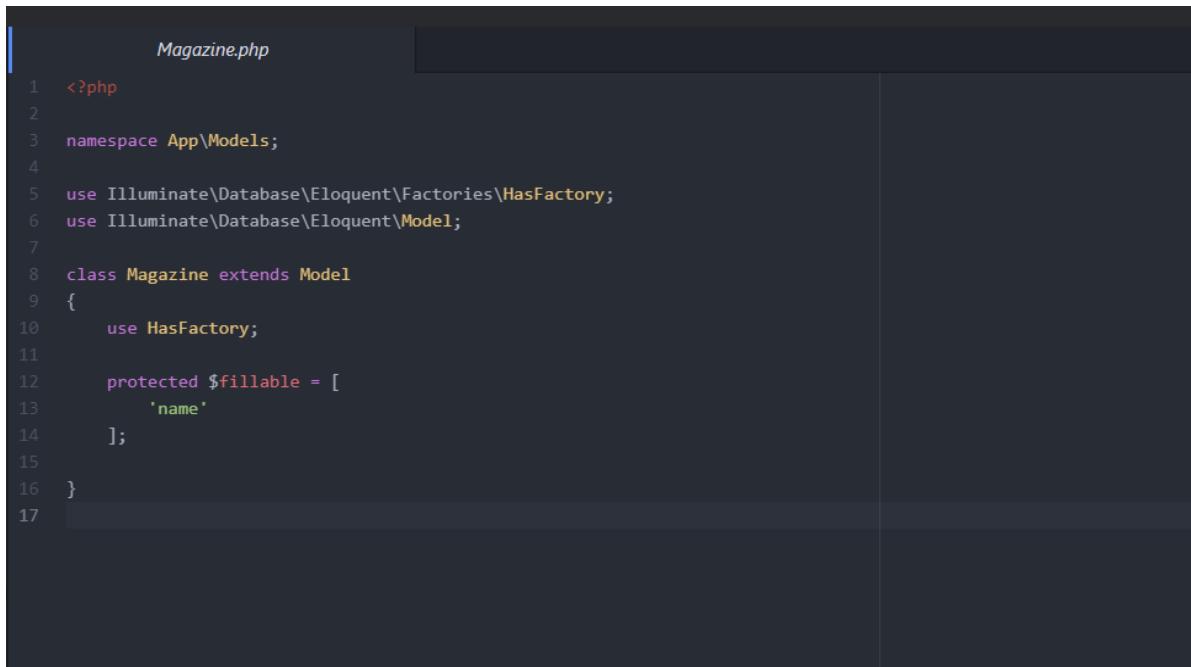
Model Product.php

Model zawiera dodatkowo metodę magazine, która określa relację wiele do wielu pomiędzy tabelą product_magazine.

```
Product.php
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Product extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'name',
14         'price',
15         'stock'
16     ];
17
18     public function magazine(){ //metoda zawierająca określenie relacji pomiędzy tabelami
19         return $this->hasMany(Magazine::class, 'product_magazine');
20     }
21 }
22 }
```

Rysunek 49 - modele, Product.php

Model Magazine.php



```
Magazine.php
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Magazine extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'name'
14     ];
15
16 }
17
```

Rysunek 50 - modele, Magazine.php

Model User.php

Klasa class User rozszerza podstawową klasę Authenticatable. Model zawiera dodatkowe właściwości klasy: \$hidden (ukrycie atrybutu) oraz \$casts (rzutowanie atrybutu).

```
User.php
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Contracts\Auth\MustVerifyEmail;
6 use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use Illuminate\Foundation\Auth\User as Authenticatable;
8 use Illuminate\Notifications\Notifiable;
9 use Laravel\Sanctum\HasApiTokens;
10 use Illuminate\Database\Eloquent\Model;
11
12 class User extends Authenticatable
13 {
14     use HasApiTokens, HasFactory, Notifiable;
15
16     /**
17      * The attributes that are mass assignable.
18      *
19      * @var string[]
20      */
21     protected $fillable = [
22         'name',
23         'email',
24         'password',
25         'type'
26     ];
27
28     /**
29      * The attributes that should be hidden for serialization.
30      *
31      * @var array
32      */
33     protected $hidden = [
34         'password',
35         'remember_token',
36     ];
37
38     /**
39      * The attributes that should be cast.
40      *
41      * @var array
42      */
43     protected $casts = [
44         'email_verified_at' => 'datetime',
45     ];
46 }
47 }
```

Rysunek 51 - modele, User.php

Kod źródłowy wszystkich modeli został stworzony w analogiczny sposób przedstawiony na powyższych przykładach.

Kontrolery

Kontrolery grupują powiązaną logikę obsługi żądań w jedną klasę. Na przykład klasa class UserController obsługuje wszystkie przychodzące żądania związane z użytkownikami. Kontrolery obsługujące moduły Produktu oraz Magazynu korzystają z osobnego, dodatkowego repozytorium opisanego w punkcie Repozytoria.

Poniżej kod źródłowy kontrolerów.

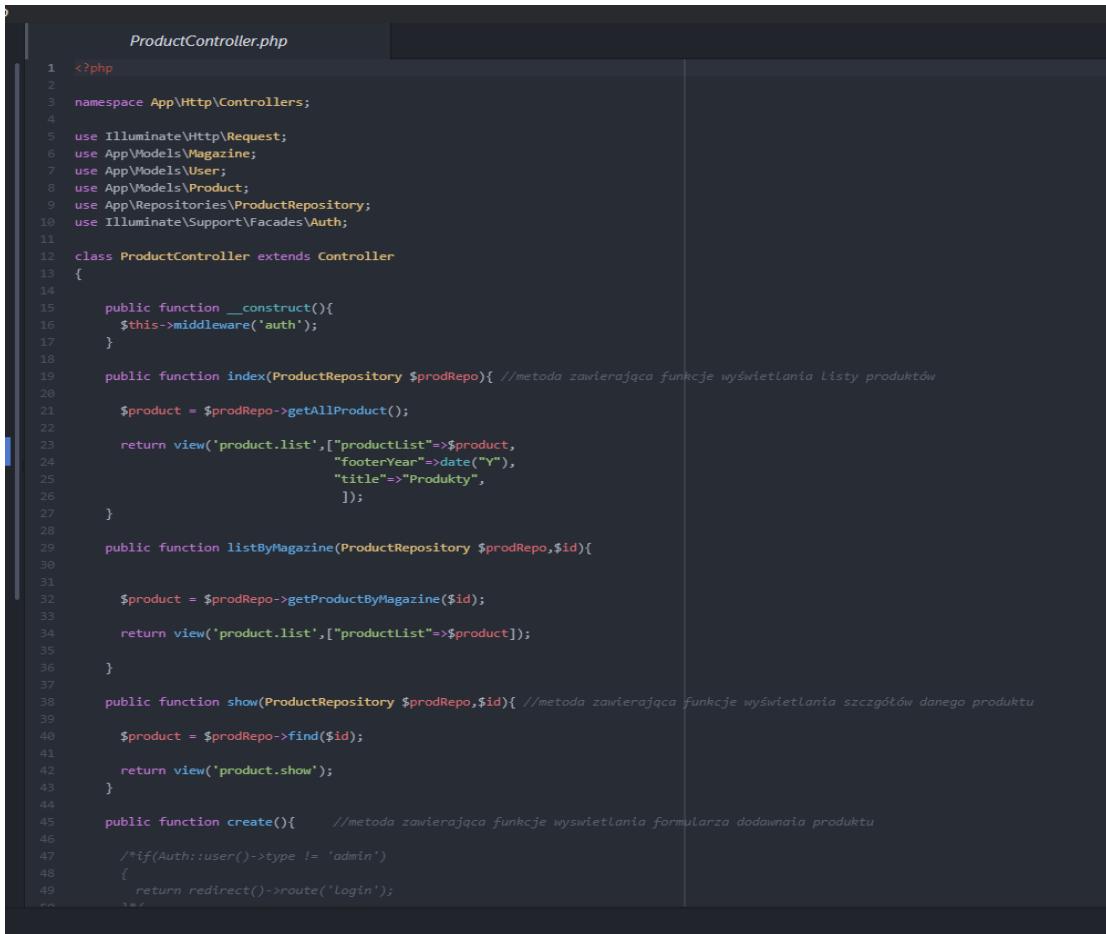
ProductController.php

Struktura kontrolera produktów:

Klasa – ProductController rozszerzająca podstawową klasę Controller

Metody:

- index: obsługa głównej listę produktów
- listByMagazine: wyświetla nazwę magazynu przypisanego do produktu
- show: wyświetla szczegółowy produktu
- create: obsługa formularza dodawania produktu
- store: obsługa validacji wprowadzonych danych oraz zapisu ich do bazy danych
- edit: obsługa formularza edycji
- delete: obsługa żądania usunięcia produktu
- editStore: obsługa walidacji edytowanych danych oraz zapisu ich do bazy danych



```

ProductController.php

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\Magazine;
7 use App\Models\User;
8 use App\Models\Product;
9 use App\Repositories\ProductRepository;
10 use Illuminate\Support\Facades\Auth;
11
12 class ProductController extends Controller
13 {
14
15     public function __construct(){
16         $this->middleware('auth');
17     }
18
19     public function index(ProductRepository $prodRepo){ //metoda zawierająca funkcje wyświetlania listy produktów
20
21         $product = $prodRepo->getAllProduct();
22
23         return view('product.list',[ "productList"=>$product,
24                                     "footerYear"=>date("Y"),
25                                     "title"=>"Produkty",
26                                     ]);
27     }
28
29     public function listByMagazine(ProductRepository $prodRepo,$id){
30
31
32         $product = $prodRepo->getProductByMagazine($id);
33
34         return view('product.list',[ "productList"=>$product]);
35
36     }
37
38     public function show(ProductRepository $prodRepo,$id){ //metoda zawierająca funkcje wyświetlania szczegółów danego produktu
39
40         $product = $prodRepo->find($id);
41
42         return view('product.show');
43     }
44
45     public function create(){ //metoda zawierająca funkcje wyświetlania formularza dodawania produktu
46
47         /*if(Auth::user()->type != 'admin')
48         {
49             return redirect()->route('Login');
50         }*/
51
52     }
53
54 }

```

Rysunek 52- ProductController.php

```

ProductController.php

47  /*if(Auth::user()->type != 'admin')
48  {
49      return redirect()->route('Login');
50 }
51
52 $magazine = Magazine::all(); // zmienna $magazine pobierajaca dane z tabeli "Magazine"
53
54 return view('product.create', ["magazine" => $magazine]);
55 }
56
57 public function store(Request $request){ //zapisywanie danych do tabeli
58
59     $request->validate([
60         'name' => 'required|max:255', //pole wymagane max 255 znaków
61         'price' => 'required', //pole wymagane
62         'stock' => 'required'
63     ]);
64
65     $product = new Product;
66     $product->name = $request->input('name');
67     $product->price = $request->input('price');
68     $product->stock = $request->input('stock');
69     $product->save();
70     $product->magazine()->sync($request->input('magazine.id'));
71
72     return redirect()->action('App\Http\Controllers\ProductController@index');
73 }
74
75 public function edit(Repository $prodRepo,$id){ //edytowanie istniejących danych, wyświetlanie formularza edycji
76
77     $magazine = Magazine::all();
78     $product = $prodRepo->find($id);
79
80     return view('product.edit',[ "magazine" => $magazine, "product" => $product]);
81 }
82
83 public function delete(Repository $prodRepo,$id){ //usuwanie danego magazynu z tabeli, usuwanie po "id"
84
85     $product = $prodRepo->delete($id);
86     return redirect('product');
87 }
88
89 public function editStore(Request $request){ //zapisywanie edytowanych danych
90
91     $product = Product::find($request->input('id'));
92     $product->name = $request->input('name');
93     $product->price = $request->input('price');
94     $product->stock = $request->input('stock');
95     $product->save();
96     $product->magazine()->sync($request->input('magazine'));
97
98     return redirect()->action('App\Http\Controllers\ProductController@index');
99 }
100 }
101

```

Rysunek 53 - ProductController.php cd.

MagazineController.php

Struktura kontrolera magazynów:

Klasa – MagazineController rozszerzająca podstawową klasę Controller

Metody:

- index: obsługa głównej listę magazynów
- show: wyświetla szczegóły magazynu
- create: obsługa formularza dodawania magazynu
- store: obsługa walidacji wprowadzonych danych oraz zapisu ich do bazy danych
- edit: obsługa formularza edycji
- delete: obsługa żądania usunięcia produktu
- editStore: obsługa walidacji edytowanych danych oraz zapisu ich do bazy danych

```

MagazineController.php

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\Magazine;
7 use App\Repositories\MagazineRepository;
8 use Illuminate\Support\Facades\Auth;
9
10 class MagazineController extends Controller
11 {
12
13     public function __construct(){
14         $this->middleware('auth');
15     }
16
17     public function index(MagazineRepository $magRepo){ //metoda zawierająca funkcje wyświetlania listy magazynów
18
19         $magazine = $magRepo->getAllMagazine();
20
21         return view('magazine.list',[ "magazineList"=>$magazine,
22                                     "footerYear"=>date("Y"),
23                                     "title"=>"Magazyny",
24                                     ]);
25     }
26
27
28     public function show(MagazineRepository $magRepo,$id){ //metoda zawierająca funkcje wyświetlania szczegółów danego magazynu
29
30         $magazine = $magRepo->find($id);
31
32         return view('magazine.show',[ "magazine"=>$magazine,"title"=>"Magazyny"]);
33     }
34
35     public function create(){ //metoda zawierająca funkcje wyświetlania formularza dodawania produktu
36
37         return view('magazine.create');
38     }
39
40     public function store(Request $request){ //zapisywanie danych do tabeli
41
42         $magazine = new Magazine;
43         $magazine->name = $request->input('name');
44         $magazine->save(); //dodawanie Magazynu do bazy poprzez przechwytcenie danych z formularza
45
46     }
47
48
49     public function edit(MagazineRepository $magRepo,$id){ //edytowanie istniejących danych, wyświetlanie formularza edycji
50
51         $magazine = $magRepo->find($id);
52
53         return view('magazine.edit',[ "magazine" => $magazine]);
54     }
55
56     public function delete(MagazineRepository $magRepo,$id){ //usuwanie danego magazynu z tabeli, usuwanie po "id"
57
58         $magazine = $magRepo->delete($id);
59         return redirect('magazine');
60     }
61
62     public function editStore(Request $request){ //zapisywanie edytowanych danych
63
64         $magazine = Magazine::find($request->input('id'));
65         $magazine->name = $request->input('name');
66         $magazine->save();
67
68         return redirect()->action('App\Http\Controllers\MagazineController@index');
69     }
70 }
71

```

Rysunek 54 - MagazineController.php

```

39     public function store(Request $request){ //zapisywanie danych do tabeli
40
41         $magazine = new Magazine;
42         $magazine->name = $request->input('name'); //dodawanie Magazynu do bazy poprzez przechwytcenie danych z formularza
43         $magazine->save();
44
45         return redirect()->action('App\Http\Controllers\MagazineController@index');
46     }
47
48
49     public function edit(MagazineRepository $magRepo,$id){ //edytowanie istniejących danych, wyświetlanie formularza edycji
50
51         $magazine = $magRepo->find($id);
52
53         return view('magazine.edit',[ "magazine" => $magazine]);
54     }
55
56     public function delete(MagazineRepository $magRepo,$id){ //usuwanie danego magazynu z tabeli, usuwanie po "id"
57
58         $magazine = $magRepo->delete($id);
59         return redirect('magazine');
60     }
61
62     public function editStore(Request $request){ //zapisywanie edytowanych danych
63
64         $magazine = Magazine::find($request->input('id'));
65         $magazine->name = $request->input('name');
66         $magazine->save();
67
68         return redirect()->action('App\Http\Controllers\MagazineController@index');
69     }
70 }
71

```

Rysunek 55 - MagazineController.php cd.

Poniższe dwie funkcje odpowiadają kolejno za:

- public function report(OrderRepository \$orderRepo,\$id) – generowanie dokumentu PZ lub WZ w przypadku kontrolera IssueController.php

-public function pdf() – generowanie widoku PDF z wygenerowanego dokumentu PZ lub WZ w przypadku kontrolera IssueController.php

```
public function report(OrderRepository $orderRepo,$id){ //metoda zawierająca funkcje wyświetlania dokumentu PZ  
  
    $order = $orderRepo->find($id);  
  
    return view('order.report',[ "order"=>$order,"title"=>"Dokument PZ"]);  
}  
  
public function pdf()  
{  
  
    $pdf = PDF::loadView('order.report');  
  
    return $pdf->download("dokumentPZ.pdf");  
}
```

Rysunek 55 - OrderController.php

Repozytoria

Repozytoria zawierają podstawowe odwołania do bazy danych. Takie jak : zwracanie wszystkich danych z tabeli, dodawanie (create), edytowanie (update) oraz usuwanie (delete) rekordu z tabeli, wyszukiwanie rekordu po id (find).

BaseRepository.php

To repozytorium definiuje podstawowe funkcjonalności związane z obsługą bazy danych.

Struktura BaseRepository:

Klasa – klasa abstrakcyjna BaseRepository zapewniająca szablon dla korzystających z niej podklas

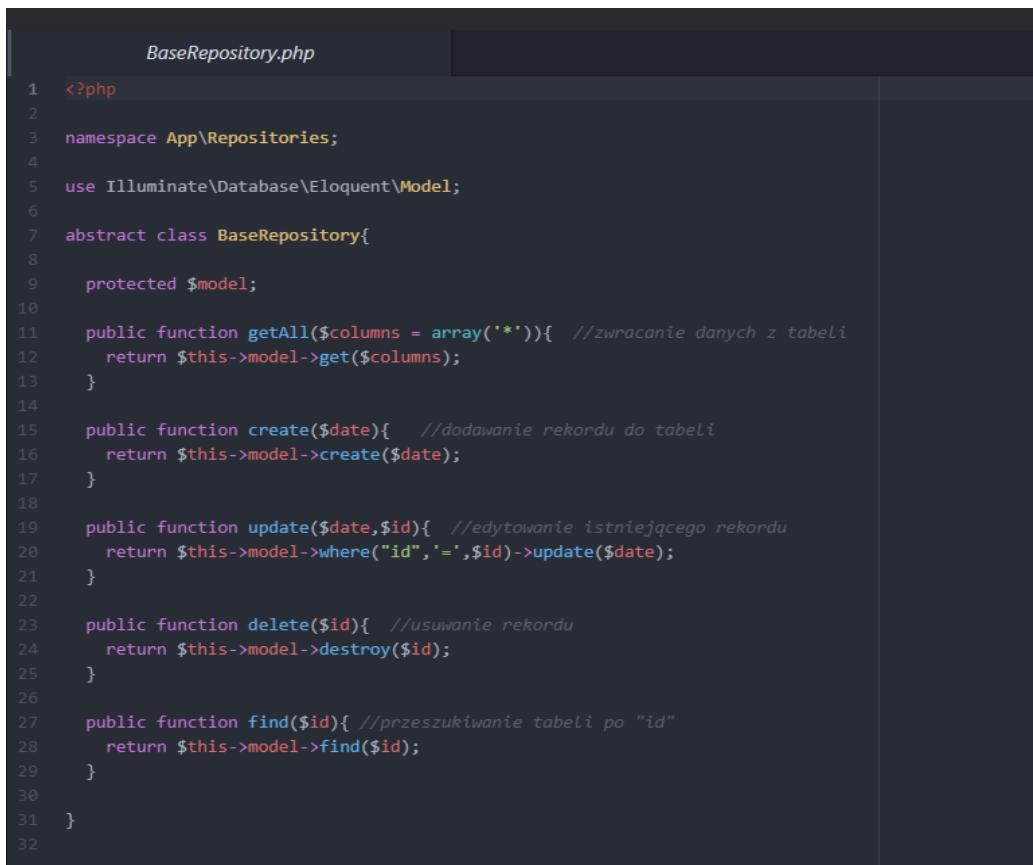
Metody:

- getAll: zwraca wszystkie dostępne kolumny tabeli

- create: dodaje datę dodania rekordu
- update: dodaje datę aktualizacji rekordu
- delete: usuwa rekord
- find: przeszukiwanie tabeli po numerze id

Zmienna - \$model z modyfikatorem dostępu - protected

Poniżej znajduje się kod źródłowy.



The screenshot shows a code editor window with the file name "BaseRepository.php" at the top. The code is written in PHP and defines an abstract class "BaseRepository". The class contains several methods: "getAll", "create", "update", "delete", and "find". Each method includes a comment describing its purpose. The code uses Laravel's Eloquent ORM syntax, such as "\$this->model->get(\$columns)" for getAll and "\$this->model->create(\$date)" for create.

```
1 <?php
2
3 namespace App\Repositories;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 abstract class BaseRepository{
8
9     protected $model;
10
11    public function getAll($columns = array('*')){ //zwieracanie danych z tabeli
12        return $this->model->get($columns);
13    }
14
15    public function create($date){ //dodawanie rekordu do tabeli
16        return $this->model->create($date);
17    }
18
19    public function update($date,$id){ //edytowanie istniejącego rekordu
20        return $this->model->where("id",'','=',$id)->update($date);
21    }
22
23    public function delete($id){ //usuwanie rekordu
24        return $this->model->destroy($id);
25    }
26
27    public function find($id){ //przeszukiwanie tabeli po "id"
28        return $this->model->find($id);
29    }
30
31 }
32 }
```

Rysunek 56 - BaseRepository.php

ProductRepository.php

Jest to rozszerzenie podstawowej klasy BaseRepository.

Struktura ProductRepository:

Klasa: ProductRepository rozszerzająca klasę BaseRepository, opowiada za obsługę zapytań do bazy danych

Metody:

- `_construct(Product $model)`: tworzenie obiektu zawierającego jako parametr zmienną `$model` z klasy abstrakcyjnej `BaseRepository`
- `getAllProduct`: zwraca wszystkie rekordy z tabeli `products`
- `getAllMagazine`: zwraca wszystkie rekordy z tabeli `magazines`
- `getProductByMagazine`: zwraca id magazynu z tabeli `magazines`

Poniżej znajduje się kod źródłowy.

```
ProductRepository.php
1 <?php
2
3 namespace App\Repositories;
4
5 use App\Models\Product;
6 use DB;
7
8
9
10 class ProductRepository extends BaseRepository{
11
12     public function __construct(Product $model){
13
14         $this->model = $model;
15     }
16
17     public function getAllProduct(){ //funkcja pobierająca wszystkie rekordy tabeli
18         return $this->model->get();
19     }
20
21     public function getAllMagazine(){ //funkcja pobierająca wszystkie rekordy tabeli
22         return DB::table('magazines')->get();
23     }
24
25     public function getProductByMagazine($id){ //funkcja pobierająca "id" magazynu, w celu przypisania produktu do odpowiedniego magazynu
26
27         return $this->model->where('magazine.id',$id)
28             ->where('magazine.id',$id);
29             ->orderBy('name','asc')->get();
30     }
31
32 }
33 }
```

Rysunek 57 - ProductRepository.php

MagazineRepository.php

Struktura MagazineRepository:

Klasa: `MagazineRepository` rozszerzająca klasę `BaseRepository`, opowiada za obsługę zapytań do bazy danych

Metody:

- `_construct(Product $model)`: tworzenie obiektu zawierającego jako parametr zmienną `$model` z klasy abstrakcyjnej `BaseRepository`
- `getAllMagazine`: zwraca wszystkie rekordy z tabeli `magazines`

Poniżej znajduje się kod źródłowy.

```
MagazineRepository.php

1 <?php
2
3 namespace App\Repositories;
4
5 use App\Models\Magazine;
6 use DB;
7
8
9 class MagazineRepository extends BaseRepository{
10
11     public function __construct(Magazine $model){
12
13         $this->model = $model;
14     }
15     public function getAllMagazine(){ //funkcja pobierająca wszystkie rekodry tabeli
16         return DB::table('magazines')->get();
17     }
18 }
19
20 }
```

Rysunek 58 - MagazineRepository.php

Routes

Trasy (routes) są zdefiniowane w pliku web.php. Zostały one podzielone na grupy poszczególnych funkcjonalności.

web.php

```

<?php

use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Auth::routes();

//routes dla funkcji Produkty
Route::group(['middleware' => 'auth'], function () {
    Route::get('/product/edit/{id}', 'App\Http\Controllers\ProductController@edit');           //formularz edycji rekordu
    Route::post('/product/edit/', 'App\Http\Controllers\ProductController@editStore');          //zapisywanie edytowanych danych
    Route::get('/product/create', 'App\Http\Controllers\ProductController@create');            //formularz dodawania rekordu
    Route::post('/product', 'App\Http\Controllers\ProductController@store');                  //zapisywanie dodanego rekordu
    Route::get('/product/', 'App\Http\Controllers\ProductController@index');                 //dane z tabeli w formie Listy
    Route::get('/product/{id}', 'App\Http\Controllers\ProductController@show');              //szczegółowy wybranego rekordu
    Route::get('/product/magazine/{id}', 'App\Http\Controllers\ProductController@listByMagazine'); //""id"" magazynu dla danego produktu
    Route::get('/product/delete/{id}', 'App\Http\Controllers\ProductController@delete');        //usuwanie rekordu
});

//routes dla funkcji Magazyny
Route::group(['middleware' => 'auth'], function () {
    Route::get('/magazine/edit/{id}', 'App\Http\Controllers\MagazineController@edit');          //formularz edycji rekordu
    Route::post('/magazine/edit/', 'App\Http\Controllers\MagazineController@editStore');         //zapisywanie edytowanych danych
    Route::get('/magazine/create', 'App\Http\Controllers\MagazineController@create');            //formularz dodawania rekordu
    Route::post('/magazine', 'App\Http\Controllers\MagazineController@store');                  //zapisywanie dodanego rekordu
    Route::get('/magazine/', 'App\Http\Controllers\MagazineController@index');                 //dane z tabeli w formie Listy
    Route::get('/magazine/{id}', 'App\Http\Controllers\MagazineController@show');              //szczegółowy wybranego rekordu
    Route::get('/magazine/delete/{id}', 'App\Http\Controllers\MagazineController@delete');        //usuwanie rekordu
});

```

CRLF UTF-8

Rysunek 59- web.php

```
web.php
Route::get('/magazine', 'App\Http\Controllers\MagazineController@index');
Route::get('/magazine/{id}', 'App\Http\Controllers\MagazineController@show');
Route::get('/magazine/delete/{id}', 'App\Http\Controllers\MagazineController@delete');
});

//routes dla funkcji Kontrahenci
Route::group(['middleware' => 'auth'], function () {
    Route::get('/client/edit/{id}', 'App\Http\Controllers\ClientController@edit');
    Route::post('/client/edit/', 'App\Http\Controllers\ClientController@editStore');
    Route::get('/client/create', 'App\Http\Controllers\ClientController@create');
    Route::post('/client', 'App\Http\Controllers\ClientController@store');
    Route::get('/client', 'App\Http\Controllers\ClientController@index');
    Route::get('/client/{id}', 'App\Http\Controllers\ClientController@show');
    Route::get('/client/delete/{id}', 'App\Http\Controllers\ClientController@delete');
});

//routes dla funkcji Przyjęcia
Route::group(['middleware' => 'auth'], function () {
    Route::get('/order/edit/{id}', 'App\Http\Controllers\OrderController@edit');
    Route::post('/order/edit/', 'App\Http\Controllers\OrderController@editStore');
    Route::get('/order/create', 'App\Http\Controllers\OrderController@create');
    Route::post('/order', 'App\Http\Controllers\OrderController@store');
    Route::get('/order/', 'App\Http\Controllers\OrderController@index');
    Route::get('/order/{id}', 'App\Http\Controllers\OrderController@report');
    Route::get('/pdf', 'App\Http\Controllers\OrderController@pdf');
    Route::get('/order/delete/{id}', 'App\Http\Controllers\OrderController@delete');
});

//routes dla funkcji Wydania
Route::group(['middleware' => 'auth'], function () {
    Route::get('/issue/edit/{id}', 'App\Http\Controllers\IssueController@edit');
    Route::post('/issue/edit/', 'App\Http\Controllers\IssueController@editStore');
    Route::get('/issue/create', 'App\Http\Controllers\IssueController@create');
    Route::post('/issue', 'App\Http\Controllers\IssueController@store');
    Route::get('/issue/', 'App\Http\Controllers\IssueController@index');
    Route::get('/issue/{id}', 'App\Http\Controllers\IssueController@report');
    Route::get('/issue/delete/{id}', 'App\Http\Controllers\IssueController@delete');
});
//routes documents
Route::group(['middleware' => 'auth'], function () {
    Route::get('/documents/', 'App\Http\Controllers\DocumentsController@index');
```

Rysunek 60- web.php cd.

```

web.php
0 Route::get('/order/', 'App\Http\Controllers\OrderController@index');
1 Route::get('/order/{id}', 'App\Http\Controllers\OrderController@report');
2 Route::get('pdf', 'App\Http\Controllers\OrderController@pdf');
3 Route::get('/order/delete/{id}', 'App\Http\Controllers\OrderController@delete');
4 });
5
6 //routes dla funkcji Wydania
7 Route::group(['middleware' => 'auth'], function () {
8     Route::get('/issue/edit/{id}', 'App\Http\Controllers\IssueController@edit');
9     Route::post('/issue/edit/', 'App\Http\Controllers\IssueController@editStore');
0     Route::get('/issue/create', 'App\Http\Controllers\IssueController@create');
1     Route::post('/issue', 'App\Http\Controllers\IssueController@store');
2     Route::get('/issue/', 'App\Http\Controllers\IssueController@index');
3     Route::get('/issue/{id}', 'App\Http\Controllers\IssueController@report');
4     Route::get('/issue/delete/{id}', 'App\Http\Controllers\IssueController@delete');
5 });
6 //routes documents
7 Route::group(['middleware' => 'auth'], function () {
8     Route::get('/documents/', 'App\Http\Controllers\DocumentsController@index');
9     Route::get('/documents/listPZ', 'App\Http\Controllers\DocumentsController@indexPZ');
0     Route::get('/documents/listWZ', 'App\Http\Controllers\DocumentsController@indexWZ');
1 });
2
3 //routes usermanagement
4 Route::group(['middleware' => 'auth'], function () {
5     Route::get('/users/', 'App\Http\Controllers\UserManagement@index');
6     Route::get('/users/edit/{id}', 'App\Http\Controllers\UserManagement@edit');
7     Route::post('/users/edit/', 'App\Http\Controllers\UserManagement@editStore');
8     Route::get('/users/create', 'App\Http\Controllers\UserManagement@create');
9     Route::post('/users', 'App\Http\Controllers\UserManagement@store');
0     Route::get('/users/delete/{id}', 'App\Http\Controllers\UserManagement@delete');
1 });
2
3 //routes Login
4 Route::post('/login', 'App\Http\Controllers\Auth\LoginController@login')->name('login');
5
6 Route::group(['middleware' => 'auth'], function () {
7     Route::get('/home', 'App\Http\Controllers\HomeController@index')->name('home');
8 });
9

```

Rysunek 61- web.php cd.

Autentykacja

Za chronienie dostępu do poszczególnych modułów odpowiada zaimplementowana w odpowiednich kontrolerach klauzula **if**. Poniżej znajduje się wycinek z kodu na przykładzie kontrolera modułu zarządzania magazynami – MagazineController.php.

Klauzula sprawdza typ zalogowanego użytkownika i na tej podstawie wyświetla odpowiedni widok dla danego użytkownika.

```
public function index(MagazineRepository $magRepo){ //metoda zawierająca funkcje wyświetlania listy magazynów  
  
    $magazine = $magRepo->getAllMagazine();  
  
    if(Auth::user()->type != 'biuro' && Auth::user()->type != 'admin')  
    {  
        return view('magazine.listauthmag',['magazineList'=>$magazine]);  
    }  
  
    return view('magazine.list',[ "magazineList"=>$magazine,  
                                "footerYear"=>date("Y"),  
                                "title"=>"Magazyny",  
                                ]);  
}
```

Rysunek 62- klauzula if - autentykacja