

Kierunek: IST

ZESPOŁOWE PRZEDSIĘWZIĘCIE INFORMATYCZNE

Scan and Store System do Zarządzania Inwentarzem z Użyciem Kodów Kreskowych i Kodów QR

Adam Klementowski

Adam Rudnicki

Adam Skowron

Wojciech Skrzypiec

Opiekun pracy: dr inż. Rafał Palak

SPIS TREŚCI

1 Wykaz symboli, oznaczeń i akronimów (opcja)	3
2 Cel i zakres przedsięwzięcia	4
2.1 Cel	4
2.2 Zakres	4
3 Słownik pojęć (opcja)	5
4 Stan wiedzy w obszarze przedsięwzięcia (opcja)	7
5 Założenia wstępne	8
5.1 Dobór technologii	8
5.2 Ograniczenia	8
6 Specyfikacja wymagań na produkt programowy	9
6.1 Baza danych	9
6.2 Historyjki	9
7 Projekt produktu programowego	11
7.1 Baza danych	11
7.2 Architektura	11
7.3 Pozostałe schematy i diagramy	11
8 Implementacja (opcja)	14
8.1 Warstwa użytkownika - frontend	14
8.2 Warstwa systemowa - backend	17
8.3 Architektura chmurowa	20
9 Demonstracja produktu programowego	23
9.1 Wstęp	23
9.2 Widoki - komputer stacjonarny	23
9.3 Widoki - telefon komórkowy	23
9.4 Opis aplikacji	23

1 WYKAZ SYMBOLI, OZNACZEŃ I AKRONIMÓW (OPCJA)

- API - (ang.) Application Programming Interface
- CORS - (ang.) Cross-Origin Resource Sharing
- CRUD - (ang.) Create, Read, Update, Delete
- CSRF - (ang.) Cross-Site Request Forgery
- DTO - (ang.) Data Transfer Object
- EAN - (ang.) European Article Number
- IaC - (ang.) Infrastructure as Code
- JWT - (ang.) JSON Web Token
- OAuth2 - (ang.) Open Authorization 2
- QR - (ang.) Quick Response
- REST - (ang.) Representational State Transfer
- S3 - (ang.) Simple Storage Service
- SCSS - (ang.) Syntactically Awesome Style Sheets
- SNS - Scan and Store - nazwa robocza projektu
- AWS - Amazon Web Services
- PWA - Progressive Web Application, Progressive Web App

2 CEL I ZAKRES PRZEDSIĘWZIĘCIA

2.1 Cel

Celem projektu jest zbadanie dojrzałości technologii Progressive Web Application (PWA) poprzez opracowanie i wdrożenie systemu do zarządzania inventarzem. System będzie wykorzystywał kody kreskowe oraz kody QR do szybszego dostępu do informacji o zasobach i ewentualnej edycji. W ramach przedsięwzięcia zostanie przeprowadzona wstępna analiza technologii PWA, jej możliwości oraz ograniczeń, a także ocena jej przydatności w praktycznych zastosowaniach, takich jak zarządzanie stanem magazynowym.

2.2 Zakres

W fazie planowania projektu ustalono, iż aplikacja końcowa powinna korzystać z funkcjonalności oferowanych przez technologię PWA, co oznacza, że będzie dostępna z poziomu przeglądarki internetowej, ale jednocześnie umożliwiająca instalację na urządzeniu jako aplikacja natywna. Aplikacja ma również umożliwiać zarządzanie stanem magazynu w podstawowym zakresie.

Z funkcjonalności PWA jakie powinny zostać zaimplementowane wybrano dostęp do kamery urządzenia aby skanować kody kreskowe i kody QR w celu szybszego dostępu do informacji na temat produktu lub stanu magazynu. Dostęp do geolokacji podyktowany jest chęcią udoskonalenia i zmniejszenia nadatność błędów wprowadzania odpowiednich danych do danego magazynu. Kolejnym ważnym punktem technologii PWA jest wykorzystanie Service Workera do obsługi trybu online oraz przyśpieszenia wczytywania zasobów (szczególnie zdjęć). Responsywność aplikacji również jest ważnym zagadnieniem. Dzięki korzystaniu z technologii PWA można optymalizować koszty utrzymania produktu (nie ma potrzeby tworzenia poszczególnych typów aplikacji na urządzenia mobilne lub komputery stacjonarne). Przez taki sposób kreowania oprogramowania można uzyskać niższą cenę dla klientów, co może okazać się szczególnie atrakcyjne dla małych i średnich przedsiębiorstw, których nie stać na drogie dedykowane rozwiązania zapewniające wsparcie.

3 SŁOWNIK POJĘĆ (OPCJA)

- API - zestaw poleceń udostępnianych osobom korzystającym z danej aplikacji
- Aplikacja natywna/dedykowana - programy zaprojektowane i stworzone z myślą o konkretnej platformie mobilnej, takiej jak Android lub iOS
- Aplikacja webowa - inaczej aplikacja internetowa, program komputerowy, który pracuje na serwerze i komunikuje się poprzez sieć z hostem użytkownika komputera
- CORS - mechanizm umożliwiający współdzielenie zasobów pomiędzy serwerami znajdującymi się w różnych domenach
- CRUD - cztery podstawowe funkcje w aplikacjach korzystających z pamięci trwałej, które umożliwiają zarządzanie nią
- CSRF - atak polegający na wykorzystaniu zaufania użytkownika do serwisu internetowego w celu wykonania nieautoryzowanych działań
- Docker - otwarte oprogramowanie służące do konteneryzacji, działające jako platforma do tworzenia, wdrażania i uruchamiania aplikacji rozproszonych. Narzędzie to pozwala umieścić program oraz jego zależności w lekkim, przenośnym, wirtualnym kontenerze, który można uruchomić na prawie każdym serwerze z systemem opartym na jądrze Linux
- DTO - w programowaniu obiektowym, obiekt który przechowuje tylko pola publiczne, bez metod, służący do przesyłania danych pomiędzy warstwami aplikacji lub systemami
- EAN - w tym dokumencie w wersji 13-cyfrowej (EAN 13), popularny kod kreskowy
- E-commerce - rodzaj handlu prowadzonego w internecie
- Endpoint - adres URL, pod którym dostępne są zasoby w API
- Framework - biblioteka narzędzi wspomagających implementację danego rozwiązania w danym języku
- IaC - to proces zarządzania zasobami centrum danych komputerowych i ich udostępniania za pośrednictwem plików definicji nadających się do odczytu maszynowego, a nie za pośrednictwem fizycznej konfiguracji sprzętu lub interaktywnych narzędzi konfiguracyjnych
- iOS - mobilny system operacyjny opracowany przez firmę Apple Inc.
- JWT - otwarty standard przemysłowy definiujący sposób wymiany danych w formie JSON pomiędzy stronami, stosowany m.in. w celu autoryzacji
- OAuth2 - standardowy protokół autoryzacji dostępu, skoncentrowany na ułatwieniu użytkownikowi przepływu autoryzacji dla aplikacji
- Poziom izolacji serializacji - poziom izolacji transakcji w bazie danych, który zapewnia, że transakcje są wykonywane w sposób sekwencyjny, eliminując możliwość wystąpienia konfliktów między równocześnie wykonywanymi transakcjami
- Progressive Web App - progresywna aplikacja internetowa uruchamiana tak jak zwykła strona internetowa, ale umożliwiająca stworzenie wrażenia działania jak natywna aplikacja mobilna lub aplikacja desktopowa. PWA może zostać zainstalowana, wtedy może posiadać własną ikonę na pulpicie oraz być niezależna od przeglądarki.
- QR - alfanumeryczny, dwuwymiarowy, matrycowy, kwadratowy kod graficzny, opracowany przez japońskie przedsiębiorstwo Denso Wave
- Presigned GET URL - link do zasobu w chmurze, który pozwala na pobranie zasobu bez konieczności autoryzacji, z ograniczeniem czasowym.
- Responsywność - responsywność aplikacji PWA odnosi się do jej zdolności do dostosowywania się do różnych rozmiarów ekranów, rozdzielcości i urządzeń, takich jak smartfony, tablety, laptopy czy komputery stacjonarne. Oznacza to, że aplikacja zapewnia spójne i optymalne doświadczenie użytkownika bez względu na platformę czy urządzenie.

- Resource server - serwer udostępniający chronione zasoby i zdolny do akceptowania i odpowiadania na żądania dotyczące chronionych zasobów przy użyciu tokenów dostępu
- REST - zbiór zasad i wytycznych o tym jak budować API aplikacji webowych
- REST API - API trzymające się zasad REST
- S3 - internetowy nośnik danych firmy Amazon, ma prosty w obsłudze interfejs WWW, który umożliwia dostęp do przechowywanych danych i zarządzanie nimi
- Service Worker - skrypt działający w tle przeglądarki, który pełni rolę pośrednika między aplikacją a siecią, a także umożliwia wykonywanie określonych zadań bez aktywnej interakcji użytkownika. Jest jednym z kluczowych elementów PWA, który pozwala na realizację funkcji takich jak tryb offline, buforowanie zasobów oraz wysyłanie powiadomień push
- Urządzenia mobilne - Smartfon, tablet

4 STAN WIEDZY W OBSZARZE PRZEDSIĘWZIĘCIA (OPCJA)

Na rynku istnieją aplikacje umożliwiające zarządzanie inventarzem, na przykład:

- Zoho Inventory - umożliwia zarządzanie zamówieniami, magazynem oraz wysyłką. Oferuje integrację z platformami e-commerce, takimi jak Shopify i Amazon, co ułatwia przygotowanie i wysyłkę odpowiednich produktów z magazynu. Cechuje się automatyzacją procesów magazynowych oraz integracją z zewnętrznymi platformami handlowymi.
- Fishbowl Inventory - jest skierowane głównie do średnich i dużych przedsiębiorstw. Oferuje zaawansowaną obsługę magazynów, skanowanie kodów kreskowych oraz zarządzanie lokalizacjami magazynowymi. Intuicyjny interfejs, dedykowany do bardziej złożonych operacji, takich jak zarządzanie wieloma magazynami i lokalizacjami.
- Sortly - narzędzie do zarządzania inventarzem z możliwością skanowania kodów QR i EAN. Przeznaczone głównie dla małych i średnich przedsiębiorstw, oferuje szybki dostęp do edycji produktów oraz statusu magazynowego po zeskanowaniu kodu QR. Prosty w obsłudze interfejs i szybkość dostępu do zmiany stanu magazynowego sprawiają, że jest to idealne rozwiązanie dla małych firm. Nie posiada zaawansowanego wsparcia dla dużych przedsiębiorstw.
- Odoo Inventory - zaawansowane oprogramowanie do zarządzania magazynami z wieloma funkcjonalnościami. Posiada zaawansowane opcje konfiguracji, w tym zarządzanie lokalizacjami magazynowymi z podziałem na alejki, półki czy pomieszczenia. Integruje skanowanie kodów kreskowych na paczkach, co ułatwia śledzenie paczek i ich zawartości. Posiada szeroką gamę funkcji, które są szczególnie przydatne dla większych przedsiębiorstw, integracja z innymi modułami Odoo, a także możliwość personalizacji zgodnie z wymaganiami użytkownika.

Wszystkie wyżej wymienione aplikacje mają na celu ułatwienie zarządzania magazynami, przy czym Zoho Inventory integruje się z internetowymi sklepami co ułatwia przygotowanie i wysyłanie odpowiednich produktów z magazynu. Fishbowl Inventory i Odoo Inventory również mają zbliżone funkcjonalności ale te rozwiązania są dedykowane większym firmom. Sortly - oprogramowanie, którego zakres jest najbliższy projektowi SNS. Jako jedyne umożliwia szybki dostęp do edycji za pomocą zeskanowania kodu QR. To oprogramowanie również udostępnia integrację ale raczej w celach kontaktowo-organizacyjnych (integracja z Microsoft Teams i Slackiem). Żadne z dyskutowanych rozwiązań nie jest napisane w technologii PWA, ale każde posiada swoją mobilną aplikację dostępną w sklepie Google. Wszystkie powyższe programy umożliwiają dostęp do statystyk i raportów, a także integrują skanowanie kodów EAN w celu dostępu do produktów lub ich lokalizacji bądź sprzedaży (Zoho Inventory). Wszystkie powyższe oprogramowania nie oferują bezpiecznej metody wyboru i definiowania lokalizacji magazynu. Oferują jedynie zarządzanie lokalizacją produktu w magazynie.

Inne znane aplikacje PWA:

- Twitter
- Facebook
- Instagram
- YouTube
- Google Photos
- Google Drive

Na rynku można znaleźć wiele aplikacji napisanych w technologii PWA, które można zainstalować na przykład na komputerze. Często jednak są tylko "skrótami" do odpowiednich wytryn przy czym wymagają ciągłego dostępu do internetu. Szczególnie serwisy społecznościowe umożliwiają pobranie aplikacji w formie PWA, choć czasami tylko na komputer, co można traktować jako "skrót" do witryny. Ogólne odczucia korzystania z obecnych aplikacji PWA są gorsze w porównaniu do aplikacji dedykowanych dla danego systemu. Aplikacja PWA Instagrama wygląda i działa gorzej niż dedykowany odpowiednik na system Android. Technologia PWA nie jest wykorzystywana w pełni, a jej popularność jest dość niska.

5 ZAŁOŻENIA WSTĘPNE

5.1 Dobór technologii

Najważniejszą technologią dla projektu jest **Progressive Web App** (PWA).

Aby wybrać framework do implementacji warstwy użytkownika (fronendu) zwrócono szczególną uwagę na dobre wsparcie dla technologii PWA i wybrano framework **Angular**. Dzięki swojej kompleksowości, wbudowanym narzędziom i ekosystemowi, Angular umożliwia szybkie i efektywne budowanie aplikacji PWA spełniające nowoczesne wymagania użytkowników.

Warstwę systemową (backend) zaimplementowano przy użyciu frameworku **Spring Boot**, który ma duże wsparcie dla zabezpieczeń (Spring Security) oraz prosty do zaimplementowania zestaw punktów końcowych (endpointów) jako REST API.

Jako bazę danych wybrano **PostgreSQL**. Zespół nie miał doświadczenia z nierelacyjnymi bazami danych, tylko taka alternatywa była dostępna na AWS.

Całość architektury aplikacji została wdrożona przy pomocy **Terraform HashiCorp** na serwery **Amazon Web Services** (AWS), co niskim kosztem pozwala na przetestowanie różnych rozwiązań bez konieczności samodzielnego utrzymania infrastruktury.

5.2 Ograniczenia

W fazie wstępnej planowania projektu przyjęto wyobrażonego klienta, który ustalił następujące założenia:

Klient

- Aplikacja wytwarzana jest dla jednego klienta
- Klient posiada wiele magazynów
- Klient chce być w stanie sprawdzić stan każdego z magazynów
- Klient chce dać swoim pracownikom możliwość dodawania artykułów do stanu magazynu
- Klient chce mieć wyłączny dostęp do dodawania i modyfikowania artykułów, magazynów i kategorii

Aplikacja

- Aplikacja musi wspierać technologię PWA
- Aplikacja musi działać na popularnych przeglądarkach
- Aplikacja musi być instalowalna na różnych urządzeniach
- Aplikacja musi wspierać skanowanie kodów kreskowych i kodów QR
- Aplikacja musi wspierać autoryzację użytkowników

Inne

- Infrastruktura musi być skalowalna
- Infrastruktura musi być zabezpieczona

6 SPECYFIKACJA WYMAGAŃ NA PRODUKT PROGRAMOWY

6.1 Baza danych

- Magazyn posiada nazwę i adres: państwo, miasto, ulica, kod pocztowy oraz opcjonalnie numer domu
- Magazyn może posiadać opis
- Każdy magazyn może mieć artykuł
- Artykuły posiadają nazwę i kategorię
- Artykuły mogą posiadać zdjęcie, opis i kod EAN13
- Kategorie posiadają nazwę
- Kategorie mogą posiadać opis
- Kategorie mogą być przypisane do więcej niż jednego artykułu
- Artykuł może się znajdować w wielu magazynach w różnych ilościach

6.2 Historyjki

Użytkownik

- Jako użytkownik chcę mieć możliwość zalogowania się na swoje konto
- Jako użytkownik chcę mieć możliwość wylogowania się ze swojego konta
- Jako użytkownik chcę mieć możliwość sprawdzenia listy przedmiotów
- Jako użytkownik chcę mieć możliwość sprawdzenia listy kategorii
- Jako użytkownik chcę mieć możliwość sprawdzenia listy magazynów
- Jako użytkownik chcę mieć możliwość sprawdzenia listy przedmiotów w magazynie
- Jako użytkownik chcę mieć możliwość dodania istniejącego przedmiotu do magazynu
- Jako użytkownik chcę mieć możliwość zwiększenia liczby przedmiotów na stanie magazynu
- Jako użytkownik chcę mieć możliwość zmniejszenia liczby przedmiotów na stanie magazynu
- Jako użytkownik chcę mieć możliwość skanowania kodów kreskowych w celu szybkiego do stanu danego przedmiotu w danym magazynie
- Jako użytkownik chcę mieć możliwość skanowania kodów QR w celu szybkiego dostępu do stanu danego przedmiotu w danym magazynie
- Jako użytkownik chcę mieć możliwość wyboru domyślnego magazynu

Administrator

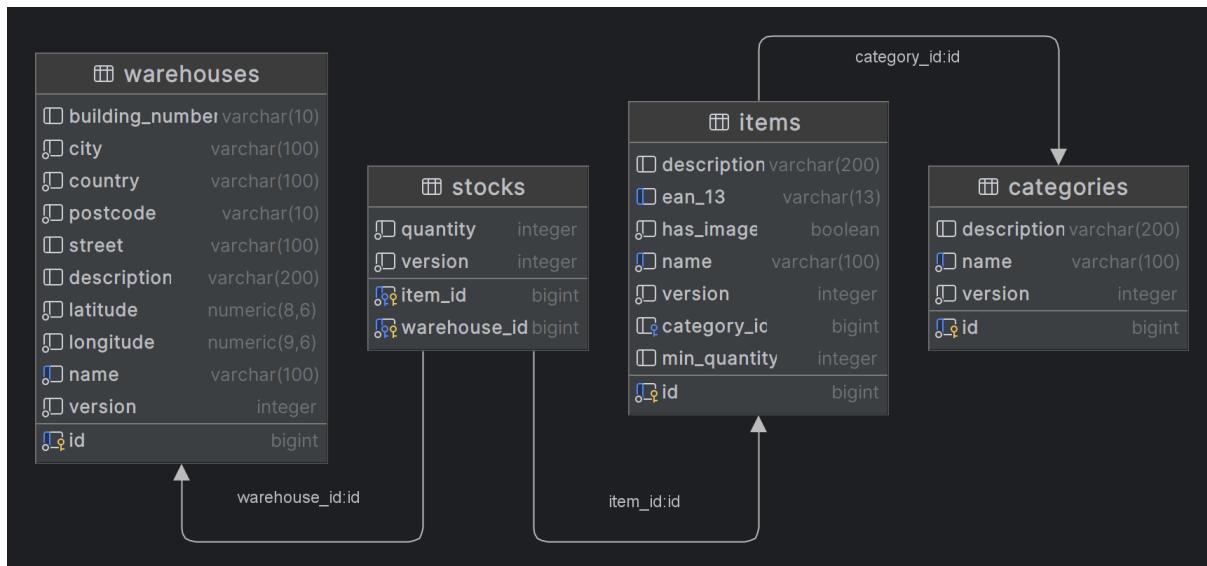
- Jako administrator chcę mieć możliwość zalogowania się na swoje konto
- Jako administrator chcę mieć możliwość wylogowania się ze swojego konta
- Jako administrator chcę mieć możliwość dodania nowego przedmiotu
- Jako administrator chcę mieć możliwość dodania nowej kategorii
- Jako administrator chcę mieć możliwość dodania nowego magazynu
- Jako administrator chcę mieć możliwość dodania istniejącego przedmiotu do magazynu
- Jako administrator chcę mieć możliwość modyfikacji istniejącego przedmiotu
- Jako administrator chcę mieć możliwość modyfikacji istniejącej kategorii
- Jako administrator chcę mieć możliwość modyfikacji istniejącego magazynu

- Jako administrator chcę mieć możliwość zwiększenia liczby przedmiotów na stanie magazynu
- Jako administrator chcę mieć możliwość zmniejszenia liczby przedmiotów na stanie magazynu
- Jako administrator chcę mieć możliwość usunięcia istniejącego przedmiotu
- Jako administrator chcę mieć możliwość usunięcia istniejącej kategorii
- Jako administrator chcę mieć możliwość usunięcia istniejącego magazynu
- Jako administrator chcę mieć możliwość usunięcia istniejącego przedmiotu z magazynu

7 PROJEKT PRODUKTU PROGRAMOWEGO

7.1 Baza danych

Baza danych została zaimplementowana w PostgreSQL według diagramu na Rysunku 1.



Rysunek 1: Diagram reprezentujący fizyczny schemat bazy danych

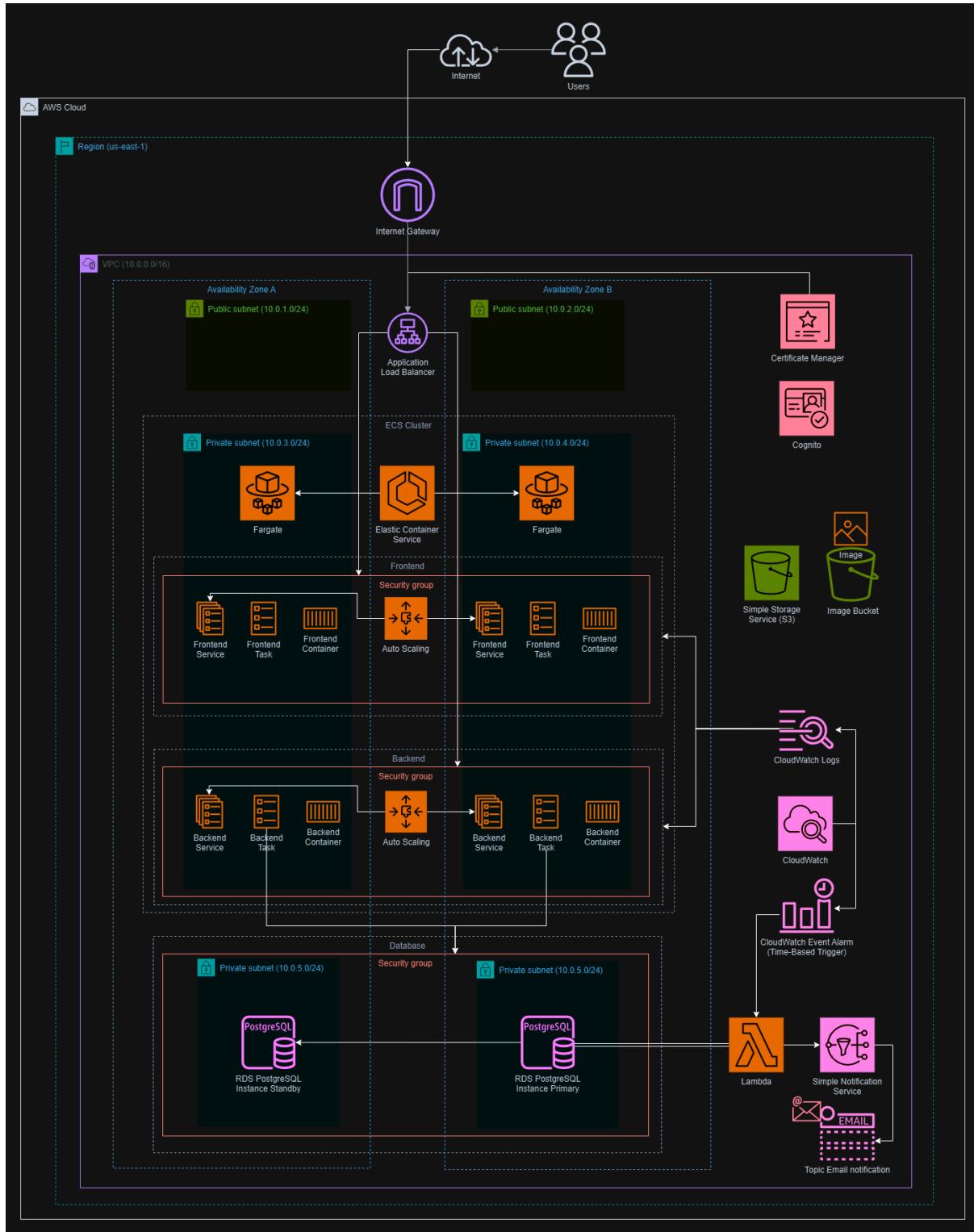
7.2 Architektura

Architektura aplikacji została wdrożona przy pomocy narzędzia HashiCorp Terraform na platformie AWS. Schemat zawierający najważniejsze elementy architektury przedstawiony jest na Rysunku 2

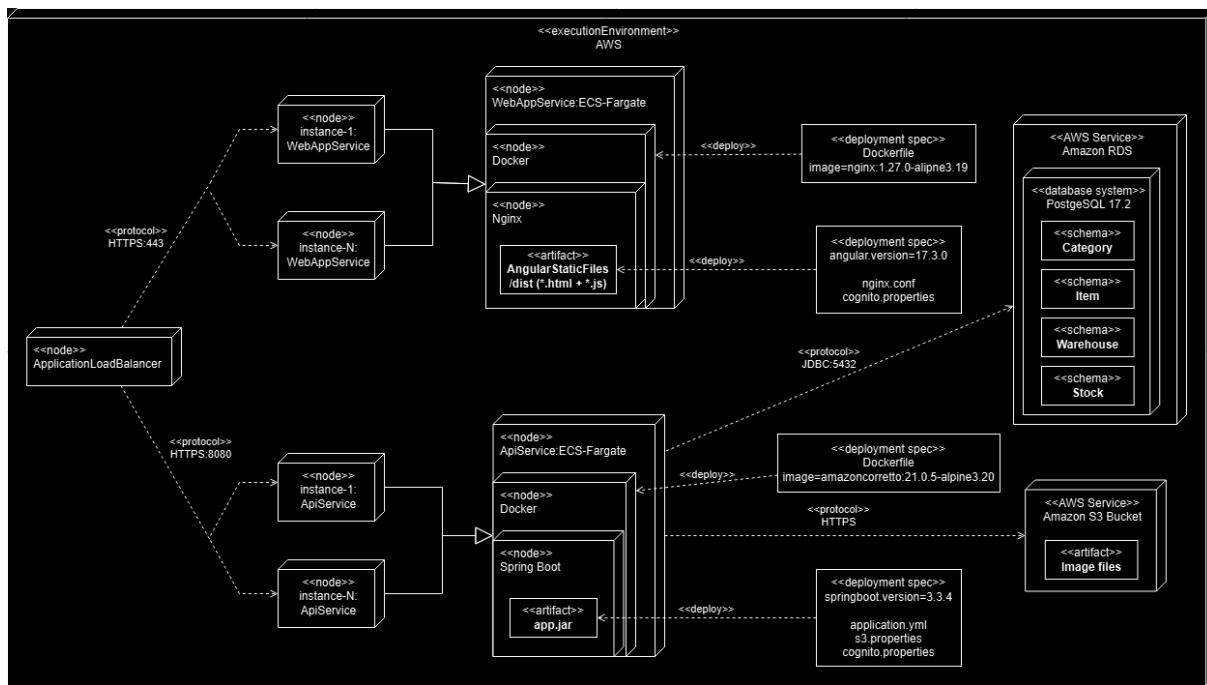
7.3 Pozostałe schematy i diagramy

Diagram wdrożenia systemu

Diagram UML wdrożenia aplikacji przedstawia abstrakcyjny zarys połączeń między kluczowymi elementami systemu, uwzględniając protokoły komunikacyjne i inne charakterystyczne elementy takie jak atrakty. Diagram przedstawiony jest na Rysunku 3.



Rysunek 2: Diagram architektury AWS



Rysunek 3: Diagram wdrożenia systemu

8 IMPLEMENTACJA (OPCJA)

Program, który rozwijano w trakcie realizacji badania technologii PWA ma cechować się odpowiednim działaniem jako aplikacja webowa i aplikacja natywna w ramach technologii PWA. Ma również stworzyć możliwość rozbudowania o wiele przydatnych funkcji występujących u konkurencji. Celem jest stworzenie oprogramowania w technologii PWA. Chcemy wykorzystać możliwości, które są udostępniane przez tę technologię, czyli celem tego projektu jest aby odczucia z korzystania z aplikacji PWA były takie jak odczucia z korzystania z aplikacji natywnej. Uniemożliwienie skanowania kodów EAN i QR oraz zautomatyzowanie lokalizacji są funkcjonalnościami wyróżniającymi to rozwiązanie na tle konkurencji. Aplikację SNS ma cechować bezpieczeństwo i niwelowanie błędów spowodowanych przez pracowników (złe przeświadczenie towaru, błędne zdefiniowanie magazynu). Poprzez dostęp do lokalizacji urządzenia pracownik nie będzie musiał być pewny, w którym magazynie się znajduje - system zrobi to za niego.

8.1 Warstwa użytkownika - frontend

8.1.1 Wstęp

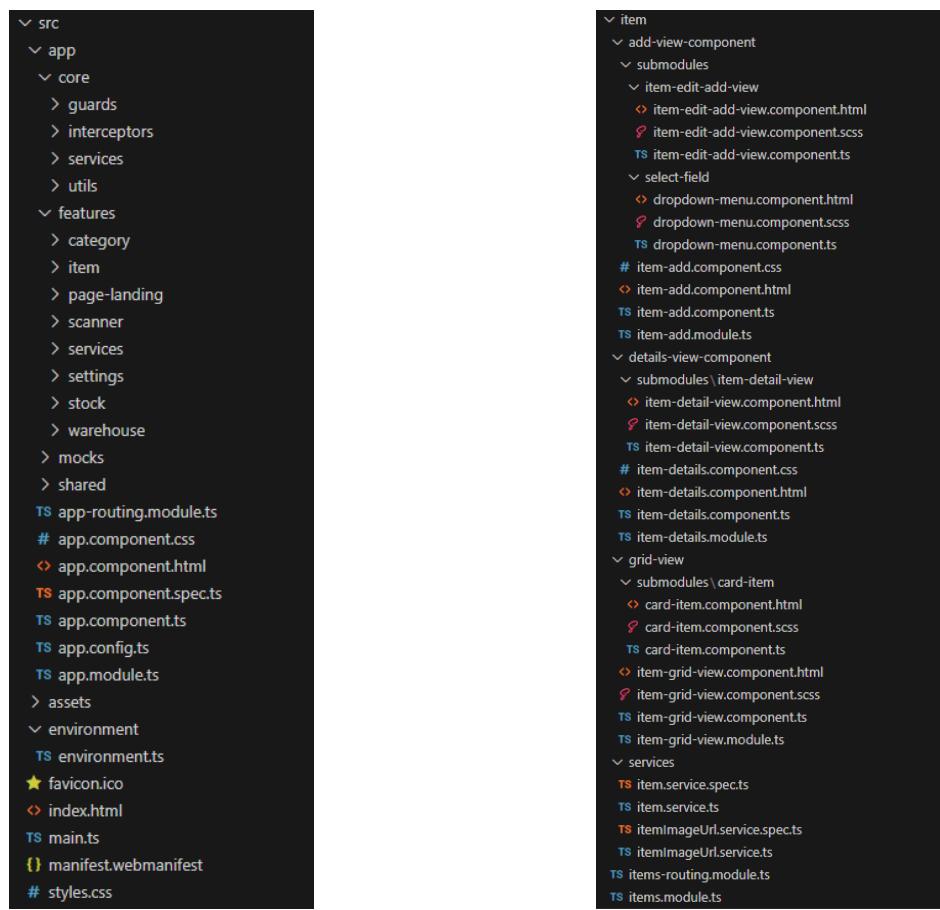
Warstwa użytkownika została zrealizowana przy użyciu frameworka Angular, który umożliwia szybkie tworzenie dynamicznych i responsywnych aplikacji webowych w języku TypeScript. Angular dostarcza zaawansowane narzędzia do zarządzania stanem aplikacji, budowania komponentów wielokrotnego użytku oraz implementacji nawigacji i dynamicznych interfejsów użytkownika. Aplikacja komunikuje się z warstwą systemową poprzez REST API, wysyłając żądania HTTP i odbierając odpowiedzi w formacie JSON. Obsługuje również integrację z usługą Amazon Cognito w celu autoryzacji i uwierzytelniania użytkowników, wykorzystując tokeny dostępu JWT. Dzięki integracji z Cognito, aplikacja umożliwia logowanie i zarządzanie sesjami użytkowników w sposób bezpieczny i zgodny z OAuth2. Przekierowywanie użytkownika do widoków chronionych odbywa się na podstawie posiadania poprawnego tokenu. Komunikacja z zasobami w Amazon S3 (np. przesyłanie lub pobieranie plików) odbywa się przy wykorzystaniu predefiniowanych adresów URL (pre-signed URLs), które są generowane przez backend. Dzięki temu frontend nie ma bezpośredniego dostępu do zasobów w S3, co podnosi bezpieczeństwo aplikacji. Dzięki zastosowaniu Angulara i TypeScript warstwa użytkownika jest modularna, łatwa w utrzymaniu oraz zorientowana na dostarczenie intuicyjnego i płynnego doświadczenia dla użytkownika końcowego.

8.1.2 Struktura plików

Warstwa użytkownika została zaimplementowana przy pomocy frameworku Angular [3] opartego o język TypeScript (TS). Strukturę plików przedstawia Rysunek 4.

Aplikacja jest tworzona przy pomocy komponentów. Każdy komponent składa się ze skryptu TS, pliku stylu SCSS oraz pliku układu HTML. Dodatkowo niektóre komponenty korzystają z serwisów napisanych w języku TypeScript.

- **app** - folder, w którym znajduje się prawie cała logika aplikacji
 - **core** - przechowywane są w nim serwisy, interceptory i guardy. Kod, który używany jest w całej aplikacji
 - * **guards** - guardy za pomocą których kontrolowany jest dostęp użytkowników do poszczególnych stron.
 - * **interceptors** - interceptory pozwalają przechwytywać i modyfikować żądania oraz odpowiedzi HTTP.
 - * **services** - serwisy do obsługi ról użytkowników (definiujące dostęp do funkcjonalności) oraz serwis obsługujący cognito
 - * **utils** - serwis, za pomocą którego zdjęcia są odpowiednio cachowane
 - **features** - folder, w którym umiejscowione są konkretne implementacje poszczególnych widoków
 - * **category** - konkretne implementacje poszczególnych widoków dotyczących kategorii
 - * **item** - konkretne implementacje poszczególnych widoków dotyczących produktów
 - * **page-landing** - konkretna implementacja widoku dotyczącego strony startowej
 - * **scanner** - konkretna implementacja widoku dotyczącego strony ze skanerem
 - * **services** - implementacja serwisu z którego korzystam w poszczególnych widokach do obsługi localstorage
 - * **settings** - konkretna implementacja widoku dotyczącego strony z ustawieniami



(a) Ogólna struktura plików

(b) Dokładna struktura jednego z komponentów

Rysunek 4: Struktura plików frontendu

- * **stock** - konkretne implementacje poszczególnych widoków dotyczących stanu magazynów
- * **warehouse** - konkretne implementacje poszczególnych widoków dotyczących magazynów
- **mocks** - folder, w którym przechowuje się klasy modeli wykorzystywane do testowania
- **shared** - komponenty graficzne zazwyczaj wykorzystywane wiele razy: strony z błędami oraz podkomponenty używane w wielu widokach
- **assets** - folder, w którym znajdują się wykorzystane asety w aplikacji
 - **logo** - folder, w którym znajdują się logo o różnych rozmiarach potrzebne do stworzenia manifestu niezbędnego do działania PWA
 - **icons** - folder, w którym znajdują się przechowywane ikonki wykorzystywane podczas tworzenia aplikacji
- **environment** - folder, w którym znajduje się environment.ts - plik z zdefiniowanymi zmiennymi środowiskowymi

8.1.3 Widoki

Warstwa wizualna tworzona jest za pomocą rozszerzonego przez Angular HTML. Styl definiowany jest za pomocą plików SCSS - rozszerzonym CSS. Widoki są tak zaprojektowane w taki sposób, aby oznaczały się wygodą użytkowania oraz odpowiednią skalownością na różnych rozmiarach ekranów, bez znaczenia czy są to urządzenia mobilne, czy komputery siedzibowe.

8.1.4 Routing

Ścieżki w aplikacji frontendowej są zabezpieczone poprzez ekstrakcję ról z tokenu JWT, który kontroluje dostęp do zasobów, zapewniając, że tylko autoryzowani użytkownicy z odpowiednimi uprawnieniami mają dostęp do określonych widoków.

8.1.5 Testy

Testy obejmują wszystkie serwisy, zapewnia to poprawność działania aplikacji w różnych scenariuszach, w tym przypadkach brzegowych i błędnych, w celu zapewnienia niezawodności i bezpieczeństwa.

8.1.6 Konteneryzacja - Docker

Aplikacja frontendowa została skonteneryzowana za pomocą Dockera, co pozwala na łatwe uruchomienie aplikacji w izolowanym środowisku kontenerowym:

- Plik `Dockerfile` wykorzystuje wieloetapowe budowanie, optymalizując proces tworzenia obrazu. Dzięki temu obraz jest lekki i zawiera tylko niezbędne zależności. Na pierwszym etapie (`cache`) pobierane są zależności z plików `package.json` i `package-lock.json` i cache'owane, co przyspiesza kolejne kompilacje. Na drugim etapie (`builder`) aplikacja jest kompilowana za pomocą Angular CLI i polecenia `ng build`, tworząc statyczne pliki w katalogu `/app/dist`, gotowe do serwowania przez Nginx. Na ostatnim etapie (`runner`) uruchamiana jest aplikacja na lekkim obrazie `nginx:1.27.0-alpine3.19`, który serwuje pliki statyczne, z domyślnie wystawionym portem 80. Skrypt `run.sh` dynamicznie nadpisuje adres API do konfiguracji Nginx, umożliwiając komunikację między frontendem a backendem.
- Plik `compose.yml` umożliwia uruchomienie aplikacji frontendowej na maszynie lokalnej w celach deweloperskich, testowych lub produkcyjnych.
- Plik `nginx.conf` definiuje konfigurację serwera Nginx, który będzie serwował pliki statyczne aplikacji frontendowej.
- Plik `nginx.conf.template` jest szablonem konfiguracji Nginx, który po uruchomieniu kontenera jest dynamicznie przekształcany przez skrypt `run.sh`, aby uwzględnić adres API.
- Plik `run.sh` uruchamia proces Nginx, zastępując zmienną `BACKEND_URL` w pliku konfiguracyjnym Nginx, dzięki czemu aplikacja frontendowa może komunikować się z odpowiednim API.
- Plik `env.template` zawiera nazwy zmiennych środowiskowych, które są wymagane do uruchomienia aplikacji.
- Plik `.dockerignore` ogranicza kontekst budowania obrazu, ignorując zbędne pliki, co przyspiesza proces budowania obrazu.

Całość konfiguracji pomaga oraz umożliwia szybkie, niezawodne i powtarzalne uruchamianie aplikacji w kontenerach, co pozwala na łatwe testowanie i rozwijanie aplikacji w różnych środowiskach.

8.1.7 Inne

W celu zaimplementowania funkcji wyboru lokalizacji użytkownika oraz lokalizacji magazynu w aplikacji wykorzystano bibliotekę Leaflet w połączeniu z danymi mapowymi dostarczonymi przez OpenStreetMap. Leaflet jest lekkim i wszechstronnym narzędziem do pracy z mapami interaktywnymi, co pozwoliło na intuicyjne i dynamiczne zarządzanie lokalizacjami w aplikacji. Aby upewnić się, że wskazana lokalizacja użytkownika jest poprawna wykorzystani dostęp do lokalizacji urządzenia.

W aplikacji wdrożono funkcję umożliwiającą szybki dostęp do informacji na temat przechowywanego produktu w magazynie poprzez skanowanie kodów QR lub kodów kreskowych EAN. Ta funkcjonalność znacząco przyspiesza procesy wyszukiwania produktów oraz zarządzaniem stanem magazynowym.

8.2 Warstwa systemowa - backend

8.2.1 Wstęp

Warstwa systemowa została napisana z wykorzystaniem framework'a Spring Boot umożliwiającym szybkie tworzenie aplikacji w języku Java. Dostarcza on w zestawie również dojrzałe narzędzia do testowania aplikacji, zarządzania zależnościami oraz konfiguracji. Warstwa systemowa obsługuje żądania HTTP, komunikuje się z bazą danych i serwisami zewnętrznymi takimi jak Amazon S3. Serwer służy jako bezstalone REST API, co oznacza, że nie przechowuje sesji ani wewnętrznych stanów, co wpływa pozytywnie na skalowalność i bezpieczeństwo aplikacji. Dodatkowo pełni rolę resource servera w architekturze OAuth2, co umożliwia uwierzytelnienie i autoryzację użytkowników za pomocą tokenów dostępu JWT oraz usługi Amazon Cognito, jako dostawcy tożsamości.

8.2.2 Struktura plików

Kod źródłowy aplikacji znajduje się w katalogu `src`, podzielonym na dwie główne części: `main` oraz `test`. Folder `main` zawiera implementację aplikacji, a `test` testy jednostkowe i integracyjne. Struktura katalogów w folderze `main` została zorganizowana tematycznie dla utrzymania przejrzystości i modularności kodu:

- **configuration** - klasy konfiguracyjne aplikacji.
 - Folder `s3` - konfiguracja integracji z Amazon S3.
 - Folder `security` - konfiguracja zabezpieczeń aplikacji, w tym ról użytkowników oraz ustawienia filtrów dla przychodzących zapytań HTTP osobno dla środowiska deweloperskiego i produkcyjnego.
- **controller** - klasy kontrolerów definiują endpointy aplikacji i są odpowiedzialne za obsługę żądań HTTP.
- **exception** - mechanizmy i sposób obsługi wyjątków w aplikacji.
 - Folder `custom` - niestandardowe wyjątki dla obsługi błędów.
 - Klasa `GlobalExceptionHandlingControllerAdvice` - globalna obsługa wyjątków.
- **model** - definicje modeli danych.
 - Folder `constant` - stałe dotyczące modeli danych.
 - Folder `constraint` - niestandardowe adnotacje walidacyjne.
 - Folder `entity` - klasy encji odpowiadające strukturom bazodanowym.
 - Folder `dto` - klasy DTO (Data Transfer Object) dla żądań HTTP i odpowiedzi.
 - * `request` - DTO dla żądań przychodzących do serwera.
 - * `response` - DTO dla odpowiedzi HTTP zwracanych przez serwer.
- **mapper** - klasy mapujące dane między warstwami aplikacji.
 - Folder `custom` - niestandardowe mapowanie pól.
 - Pozostałe klasy - mapowanie DTO na encje i odwrotnie.
- **repository** - interfejsy repozytoriów komunikujące się z bazą danych.
- **service** - klasy serwisowe implementujące logikę biznesową.
- **util** - klasy narzędziowe i pomocnicze.
 - Folder `model` - klasy i metody pomocnicze do obsługi modeli danych.
 - Pozostałe klasy - funkcje ogólnego zastosowania.
- **resources** - pliki konfiguracyjne w formacie YAML dla poszczególnych środowisk.
 - `application.yml` - główny plik konfiguracyjny aplikacji definiujący ustawienia podstawowe i domyślnie wspólne dla wszystkich środowisk.
 - `application-dev.yml`, `application-dev-h2.yml` - konfiguracja środowiska deweloperskiego, z ustawieniami, takimi jak debugowanie, automatyczne aktualizacje schematu bazy danych oraz dostęp do endpointów bez autoryzacji.
 - `application-prod.yml` - konfiguracja środowiska produkcyjnego, z zabezpieczeniami, takimi jak autoryzacja OAuth2 z JWT, ograniczenie dostępu do endpointów oraz logowanie na poziomie ostrzeżeń.

8.2.3 Konfiguracja

W środowisku deweloperskim aplikacja umożliwia pełny dostęp do endpointów oraz włącza CORS dla lokalnych i testowych domen. W środowisku produkcyjnym zabezpieczenia są bardziej restrykcyjne – dostęp do zasobów wymaga autoryzacji OAuth2, a jedynym publicznie dostępnym endpointem jest '/health_check', wymagany przez Application Load Balancer.

Spring Boot serwer działa jako bezstanowe REST API oraz resource server. Aplikacja nie przechowuje sesji ani stanu użytkownika, a jedynie operuje na tokenach JWT w celu uwierzytelnienia przy każdym żądaniu i udzielenia dostępu do zasobów. Mechanizmy zabezpieczeń CSRF są wyłączone, ponieważ aplikacja korzysta z tokenów JWT jako bezpiecznej metody uwierzytelnienia.

Uwierzytelnienie odbywa się na poziomie zapytania HTTP, gdzie serwer weryfikuje tożsamość użytkownika poprzez potwierdzenie jej z Amazon Cognito. Przy pomyślnej weryfikacji, role użytkownika wyciągane są z tokena JWT, dzięki czemu możliwa jest autoryzacja użytkownika względem poszczególnych endpointów.

Integracja z Amazon S3 obejmuje generowanie presigned GET URL, które są generowane po stronie serwera i ograniczone czasowo, co zwiększa bezpieczeństwo i pozwala na bezpieczne udostępnianie zasobów.

8.2.4 Kontrolery

Endpointy kontrolerów są zabezpieczone przed nieautoryzowanym dostępem za pomocą ról zawartych w tokenie JWT. Do autoryzacji na warstwie endpointów kontrolerów zastosowano adnotację `@PreAuthorize`, wymagającą odpowiednich ról użytkownika, np. `ADMIN` dla operacji modyfikacji krytycznych danych. Kontrolery zostały podzielone zgodnie z funkcjonalnościami. Przykładowo, `ItemController` obsługuje operacje CRUD dla przedmiotów, w tym zarządzanie powiązaniami z kategoriami oraz obrazami. Pozostałe kontrolery mają podobną strukturę i działanie, skupiając się na specyficznych zasobach i logice biznesowej, przy zachowaniu spójnych mechanizmów uwierzytelnienia, autoryzacji i odpowiedzi HTTP. Natomiast inny osobny kontroler `HealthCheckController` udostępnia publiczny endpoint `/health_check` do monitorowania stanu aplikacji bez konieczności uwierzytelnienia dla Application Load Balancera.

8.2.5 Inne

Warstwa serwisowa wykorzystuje adnotacje usprawniające i zabezpieczające spójność danych oraz współpracę odczytu i zapisu danych, takie jak `@Transactional`, `@Lock`, czy `@Version`. Zastosowanie transakcyjności oraz odpowiednich mechanizmów synchronizacji, takich jak tryb izolacji serializowalnej, zapewnia spójność danych i unika problemów związanych z równoczesnym dostępem do danych i ich modyfikacją.

Dane są walidowane dodatkowo za pomocą adnotacji na poziomie DTO oraz na poziomie encji.

Globalny mechanizm obsługi wyjątków (`GlobalExceptionHandler`) ułatwia obsługę błędów w aplikacji oraz zwracanie odpowiednich odpowiedzi HTTP w zależności od rodzaju błędu.

8.2.6 Testy

Testy jednostkowe i integracyjne w katalogu `test` obejmują wszystkie istotne komponenty aplikacji, w tym serwisy, kontrolery, repozytoria, mechanizmy walidacyjne i autoryzacyjne. Pokrycie kodu jest wysokie, na poziomie 90 procent, co zapewnia poprawność działania aplikacji w różnych scenariuszach, w tym przypadkach brzegowych i błędnych, w celu zapewnienia niezawodności i bezpieczeństwa.

8.2.7 Konteneryzacja - Docker

Aplikacja została skonteneryzowana za pomocą Dockera, co umożliwia łatwe uruchomienie aplikacji w izolowanym środowisku kontenerowym:

- Plik `Dockerfile` wykorzystuje wieloetapowe budowanie, optymalizując proces tworzenia obrazu. Dzięki temu obraz jest lekki i zawiera tylko niezbędne zależności. Na pierwszym etapie (cache) pobierane są zależności Gradle na podstawie pliku `build.gradle` i cache'owane, co przyspiesza kolejne komplikacje. Na drugim etapie (builder) aplikacja jest budowana jako plik JAR za pomocą polecenia `gradle bootJar`. Na ostatnim etapie (runner) uruchamiana jest aplikacja na lekkim obrazie `amazoncorretto:21-alpine` jako użytkownik nieuprzywilejowany, co zwiększa bezpieczeństwo, z domyślnie wystawionym portem 8080.
- Plik `docker-compose.yml` definiuje dwa serwisy, co umożliwia uruchomienie aplikacji wraz z bazą danych PostgreSQL na maszynie lokalnej w celach deweloperskich, testowych lub produkcyjnych.

1. **springboot** - aplikacja Spring Boot zbudowana na podstawie pliku **Dockerfile** z odpowiednimi zmiennymi środowiskowymi, takimi jak profil aplikacji, baza danych PostgreSQL, integracja z Amazon S3 i Cognito.
 2. **postgresql_database** - baza danych PostgreSQL z mechanizmem **healthcheck** sprawdzającym dostępność bazy danych i trwały wolumen na dane.
- Plik **.env.template** zawiera nazwy zmiennych środowiskowych, które są wymagane do uruchomienia aplikacji.
 - Plik **.dockerignore** ogranicza kontekst budowania obrazu, ignorując zbędne pliki, co przyspiesza proces budowania obrazu.

Całość konfiguracji pomaga oraz umożliwia szybkie, niezawodne i powtarzalne uruchamianie aplikacji w kontenerach, co pozwala na łatwe testowanie i rozwijanie aplikacji w różnych środowiskach.

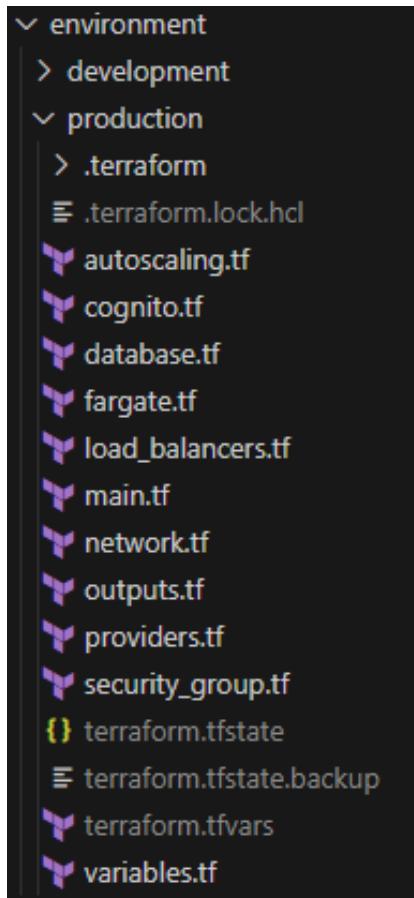
8.3 Architektura chmurowa

8.3.1 Wstęp

Architektura chmurowa została napisana w oprogramowaniu HashiCorp Terraform i wdrożona na serwery Amazon Web Services (AWS) [1]. HashiCorp Terraform (albo krótko Terraform) [4] to oprogramowanie Infrastructure as Code (IaC) pozwalające na skryptowe wdrażanie infrastruktury na serwery dostawcy.

8.3.2 Struktura plików

Struktura plików została podzielona ze względu na środowisko produkcyjne i testowe. Na Rysunku 5 przedstawione jest tylko środowisko produkcyjne ze względu na tą samą strukturę systemu plików. Z tego samego względem omówione zostanie tylko środowisko produkcyjne.



Rysunek 5: Struktura plików Terraform

Pliki `.tf` zawierają opis zasobów wdrażanych na serwery. Plik `.tfstate` przechowuje dokładny spis zasobów już wdrożonych wraz z ich unikalnymi identyfikatorami, sekretami, itd. Opis pozostałych plików:

- `main.tf` - wersje oprogramowania wykorzystywane w projekcie
- `providers.tf` - konfiguracja dostawcy AWS
- `variables.tf` - spis wszystkich zmiennych wraz z ich opisami i niektórymi wartościami domyślnymi
- `terraform.tfvars` - niewersjonowany plik zawierający wartości niektórych zmiennych; przede wszystkim przechowuje sekrety
- `outputs.tf` - zbiór przydatnych wartości wyjściowych, np. nazwa DNS Application Load Balancera
- `network.tf` - zasoby sieciowe: VPC, brama sieciowa, tablice routingu, podsieci
- `security_groups.tf` - grupy bezpieczeństwa (działają jak zapora ogniodziałająca - ang. firewall) konfigurujące przychodzący i wychodzący ruch sieciowy
- `cognito.tf` - dostawca tożsamości Amazon Cognito i jego konfiguracja

- `load_balancers.tf` - Application Load Balancer, zasoby nasłuchujące na portach 80, 443 i 8080 oraz grupy celów
- `fargate.tf` - definicje zadań wykonywanych przy pomocy AWS Fargate oraz serwisy które nimi zarządzają
- `autoscaling.tf` - zestaw reguł pozwalających na automatyczne zmniejszanie lub zwiększenie liczby działających instancji modułów (frontendu i backendu)
- `database.tf` - konfiguracja bazy danych RDS
- `sns.tf` - definicja Simple Notification Service (SNS) do wysyłania powiadomień na dany temat do jego subskrybentów
- `lambda.tf` - definiuje funkcję Lambda napisaną w języku TypeScript, uruchamianą okresowo za pomocą alarmów CloudWatch. Funkcja ta pobiera dane z bazy danych i powiadamia subskrybentów, z odpowiednimi rolami zdefiniowanymi w Cognito, na określony temat SNS poprzez e-mail

8.3.3 Wykorzystane zasoby

Amazon Virtual Private Cloud (VPC): Amazon VPC pozwala na tworzenie izolowanej, logicznie odseparowanej sieci w chmurze AWS. Umożliwia pełną kontrolę nad konfiguracją sieci, w tym zakresami adresów IP, podsieciami, trasami, bramami i konfiguracją reguł zapory sieciowej.

Amazon Cognito: Amazon Cognito umożliwia zarządzanie tożsamościami i autoryzacją użytkowników w aplikacjach internetowych oraz mobilnych. Zapewnia obsługę logowania użytkowników, synchronizacji danych oraz integracji z dostawcami tożsamości. Umożliwia również tworzenie grup użytkowników i zarządzanie rolami.

Amazon Relational Database Service (RDS): Amazon RDS to zarządzana usługa baz danych, która umożliwia łatwe konfigurowanie, działanie i skalowanie relacyjnych baz danych. Obsługuje popularne silniki, takie jak MySQL, PostgreSQL, MariaDB, Oracle oraz SQL Server.

Amazon Application Load Balancer (ALB): Amazon ALB to rodzaj load balansera, który automatycznie rozdziela ruch aplikacji na wiele celów, takich jak kontenery, instancje EC2 lub funkcje Lambda. Jest zoptymalizowany do obsługi ruchu HTTP/HTTPS oraz oferuje zaawansowane funkcje, takie jak routing oparte na zawartości czy target groups.

Amazon Elastic Container Service (ECS): Amazon ECS to zarządzana usługa orkiestracji kontenerów, która umożliwia uruchamianie, zatrzymywanie i zarządzanie kontenerami Docker na klastrach instancji EC2 lub w ramach AWS Fargate (serverless). Jest w pełni zintegrowana z innymi usługami AWS.

Amazon Certificate Manager (ACM): Amazon ACM umożliwia łatwe zarządzanie certyfikatami SSL/TLS. Automatyzuje procesy wydawania, odnawiania i wdrażania certyfikatów dla aplikacji działających w AWS, eliminując potrzebę ręcznej konfiguracji.

Amazon CloudWatch: Amazon CloudWatch to usługa monitorowania zasobów i aplikacji AWS. Umożliwia zbieranie, przeglądanie i analizowanie metryk, logów oraz ustawianie alarmów. Zapewnia również możliwość automatycznej reakcji na zmiany wydajności systemu.

Amazon Simple Storage Service (S3): Amazon S3 to skalowalna usługa magazynowania obiektów, która pozwala na przechowywanie dowolnej ilości danych. Jest idealna do tworzenia kopii zapasowych, archiwizacji, hostowania zasobów statycznych i integrowania z aplikacjami chmurowymi.

AWS Auto Scaling: AWS Auto Scaling automatycznie dostosowuje liczbę zasobów obliczeniowych (np. kontenerów Fargate) w odpowiedzi na zmieniające się obciążenie. Zapewnia elastyczność, wysoką dostępność i optymalizację kosztów poprzez skalowanie w górę lub w dół w oparciu o zdefiniowane zasady.

Amazon Simple Notification Service (SNS): Amazon SNS to usługa zarządzania komunikatami, która umożliwia publikowanie i subskrybowanie tematów. Umożliwia wysyłanie powiadomień do wielu odbiorców, takich jak e-maile, SMS-y, aplikacje mobilne czy inne usługi AWS.

AWS Lambda: Amazon Lambda to usługa obliczeniowa, która uruchamia kod w odpowiedzi na zdarzenia i zarządza zasobami obliczeniowymi w sposób zautomatyzowany. Pozwala na tworzenie funkcji serwerless, które są skalowalne, elastyczne i nie wymagają zarządzania infrastrukturą.

9 DEMONSTRACJA PRODUKTU PROGRAMOWEGO

9.1 Wstęp

W ramach demostracji produktu programowego stworzono przykładowe artykuły, kategorie, magazyny i stany magazynów korzystając z tekstur gry Minecraft. Wszystkie tekstury pochodzą z portalu minecraft.fandom.com [2]

9.2 Widoki - komputer stacjonarny

Podstawowe widoki wyświetlane na komputerze stacjonarnym zostały przedstawione na Rysunkach 6, 8, 10, 15, 20, 25, 28, 29.

9.3 Widoki - telefon komórkowy

Podstawowe widoki wyświetlane na telefonie komórkowym zostały przedstawione na Rysunkach 7, 9, 11, 16, 21, 26, 30, 31.

9.4 Opis aplikacji

9.4.1 Podstawowe widoki

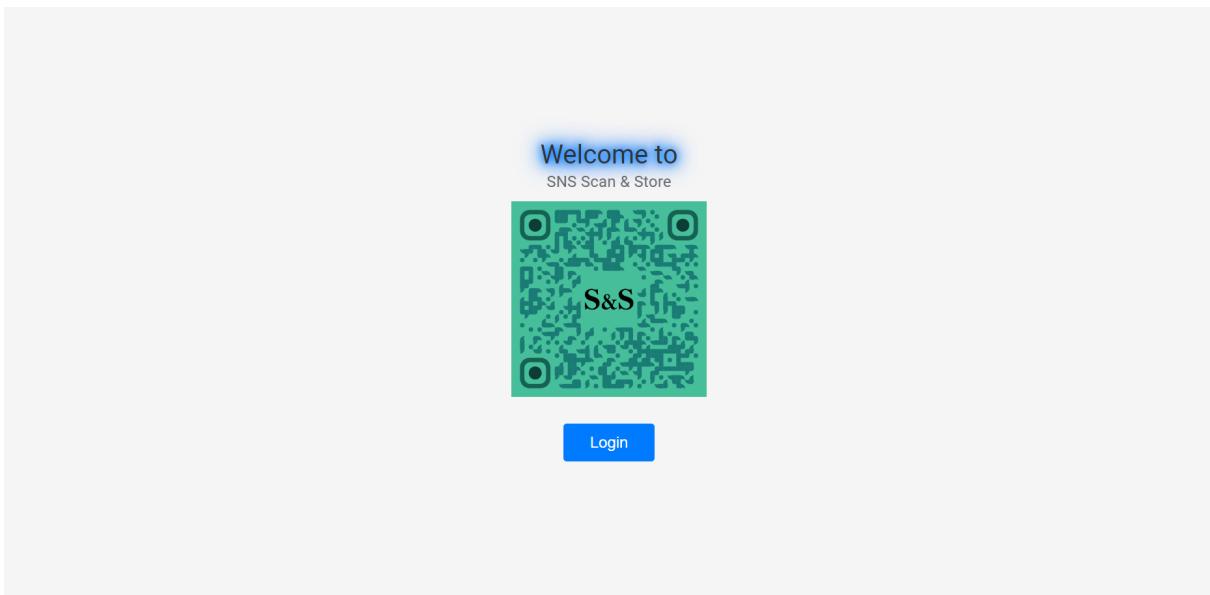
Aplikacja zawiera 7 podstawowych widoków:

- stronę startową (Rysunki 6 i 7),
- artykuły (Rysunki 10 i 11),
- kategorie (Rysunki 15 i 16),
- magazyny (Rysunki 20 i 21),
- ustawienia (Rysunki 25 i 26)
- stany magazynów (Rysunki 29 i 31),
- skaner kodów kreskowych i kodów QR.

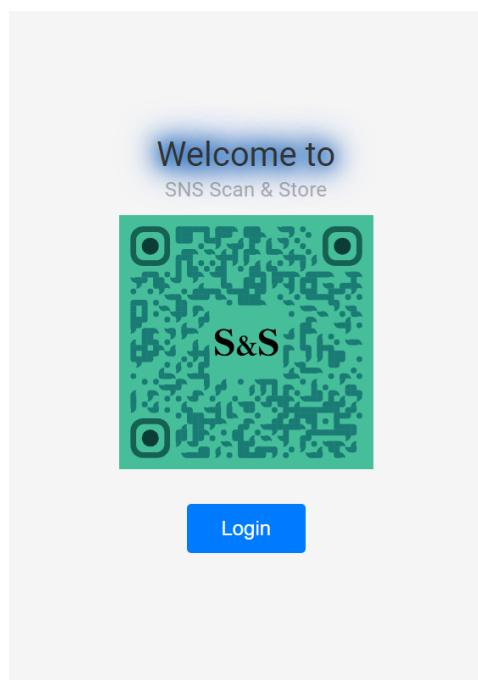
9.4.2 Strona startowa

Strona startowa może przyjąć 2 stany: użytkownik niezalogowany (Rysunek 6) i użytkownik zalogowany (Rysunek 8). W przypadku użytkownika niezalogowanego jedyną funkcjonalnością jest przekierowanie do logowania. W przypadku użytkownika zalogowanego istnieją dwie funkcjonalności: wylogowanie lub przejście do aplikacji.

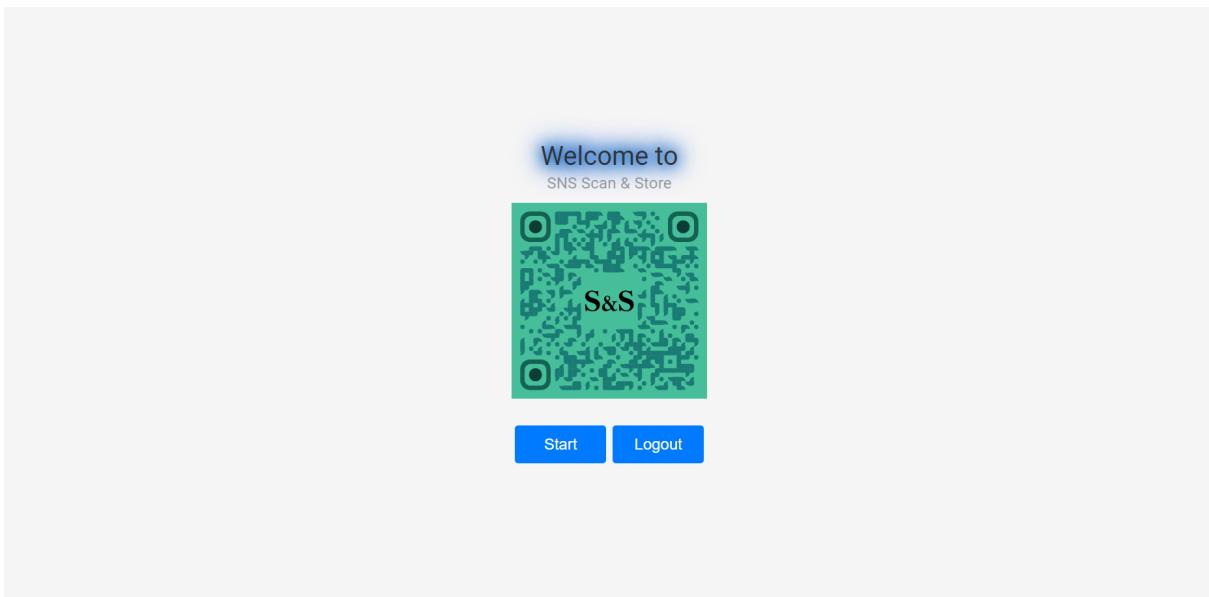
Po poprawnej próbie zalogowania użytkownik zostaje automatycznie przekierowany do aplikacji. Jeśli kiedykolwiek ponownie znajdzie się na stronie startowej to dostanie możliwość przejścia do aplikacji ręcznie lub możliwość wylogowania się. Po wylogowaniu użytkownik zostaje przekierowany z powrotem na stronę startową.



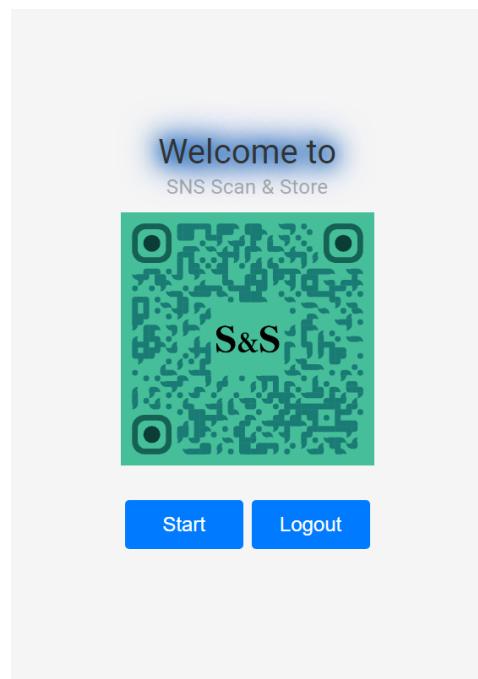
Rysunek 6: Strona startowa na komputerze stacjonarnym



Rysunek 7: Strona startowa na telefonie komórkowym



Rysunek 8: Strona startowa po zalogowaniu na komputerze stacjonarnym



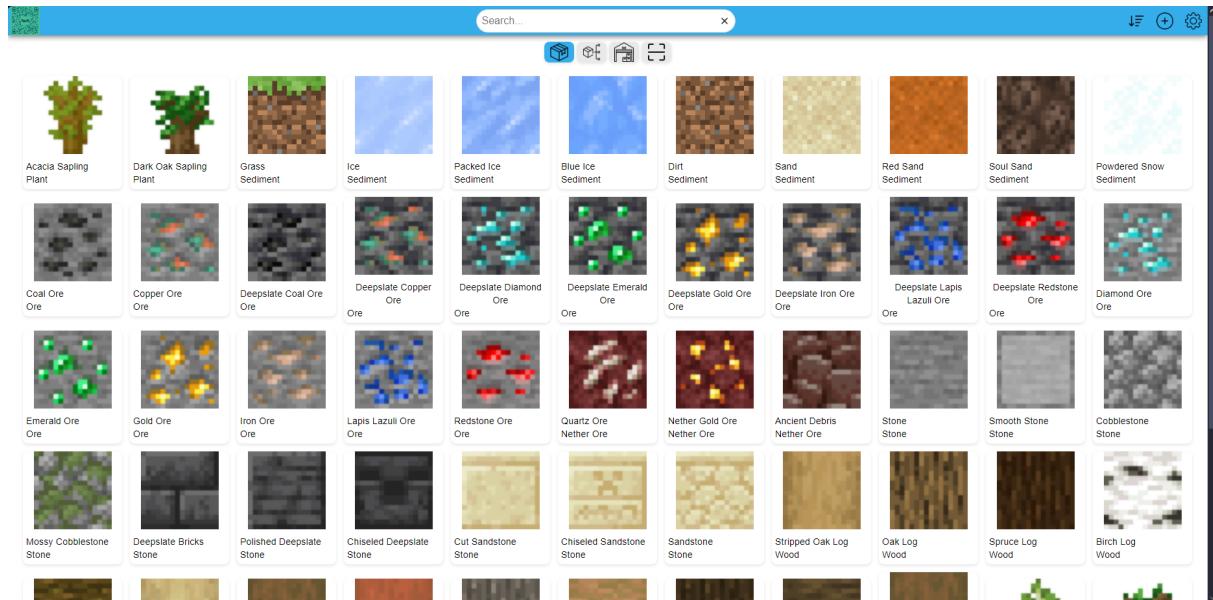
Rysunek 9: Strona startowa po zalogowaniu na telefonie komórkowym

9.4.3 Artykuły

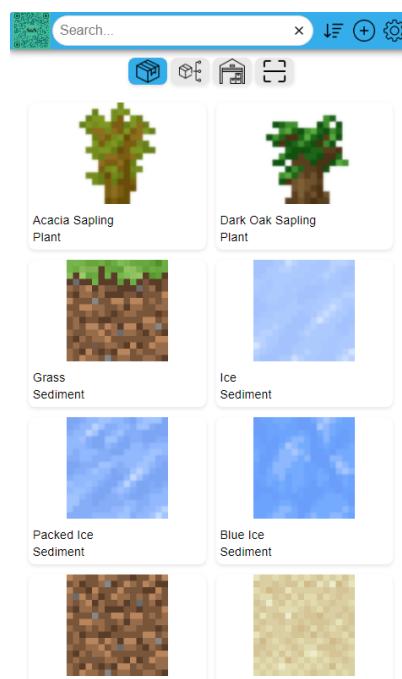
Strona artykułów przedstawiona na rysunku 10 służy do pełnej obsługi wszystkich dostępnych artykułów, które mogą znaleźć się w dowolnym magazynie. W górnym pasku znajdują się opcje pozwalające na kolejno od lewej:

- przejście z powrotem do strony startowej (element wspólny dla wszystkich widoków)
- wyszukanie konkretnego artykułu z uwzględnieniem nazwy i nazwy kategorii
- posortowanie artykułów względem nazwy
- dodanie nowego artykułu
- przejście do widoku ustawień (element wspólny dla wszystkich widoków)

Wybór konkretnego artykułu następuje poprzez naciśnięcie na konkretny wpis. Po wyborze użytkownik jest przeniesiony do strony ze szczegółami artykułu (Rysunek 12), a w górnym pasku otrzymuje możliwość edycji i usunięcia danego artykułu.



Rysunek 10: Widok przedmiotów na komputerze stacjonarnym



Rysunek 11: Widok przedmiotów na telefonie komórkowym

The screenshot shows the 'Details' view of a product. At the top left is a thumbnail image of a brown textured object. To its right are five input fields: 'Name' (Dot), 'Category' (Sediment), 'Description' (empty), 'EAN13' (empty), and 'Minimum Quantity' (empty). Below these fields is a large empty text area.

Rysunek 12: Widok szczegółów przedmiotu

W widoku edycji (Rysunek 13) użytkownik może zmienić dowolny element dotyczący artykułu, włącznie z dodaniem lub usunięciem zdjęcia.

The screenshot shows the 'Edit' view of the same product. It includes all the fields from the previous screenshot plus a 'Remove Image' button above the image area. The image area now displays a small placeholder image with the text 'No Image'.

Rysunek 13: Widok edycji przedmiotu

Po przejściu do widoku dodawania nowego przedmiotu otrzymamy widok, przedstawiony na Rysunku 14, który pozwala na dokładne te same funkcjonalności co edycja, lecz na początku nie posiada żadnych wypełnionych pól.

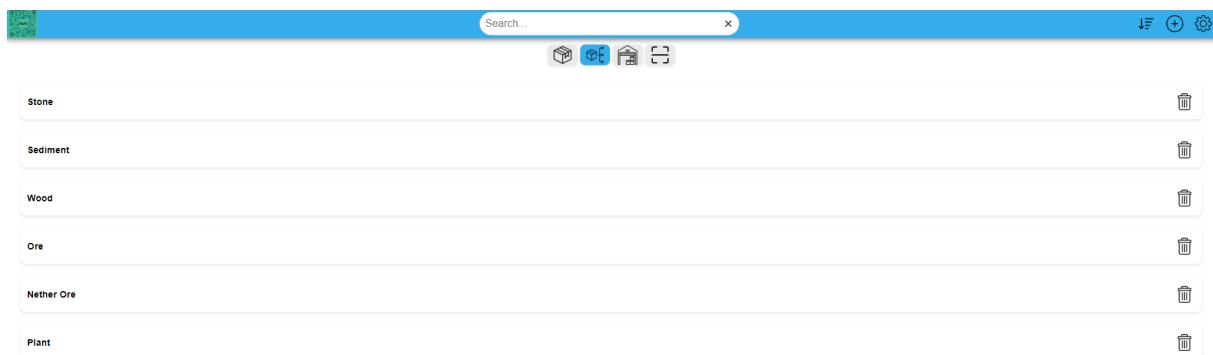
The screenshot shows the 'Add Item' view. It features a large placeholder image area with the text 'No Image'. Below it are five empty input fields: 'Name' (Dot), 'Category' (Select a category), 'Description' (Enter text), 'EAN13' (empty), and 'Minimum Quantity' (empty). A 'Search' button is located at the bottom right.

Rysunek 14: Widok dodawania przedmiotu

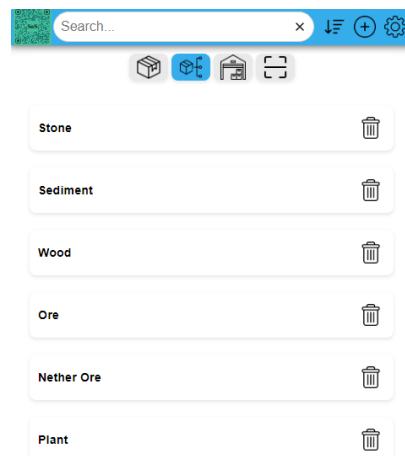
9.4.4 Kategorie

Strona kategorii przedstawiona na rysunku 15 pozwala na przegląd i dodawanie kategorii dla przedmiotów. Podobnie jak w przypadku artykułów w pasku górnym znajdują się te same opcje. Także analogicznie do artykułów wybór kategorii następuje poprzez nacisnięcie odpowiedniego elementu.

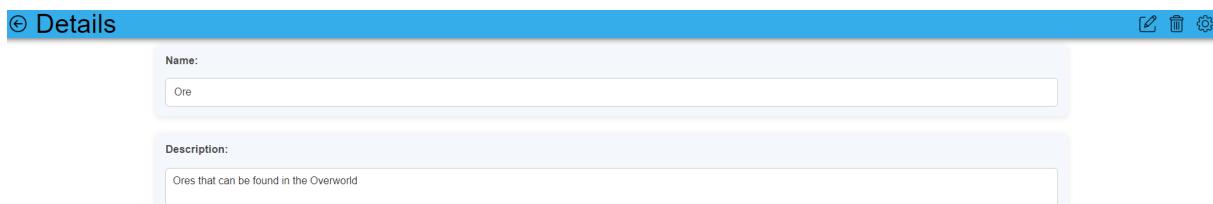
Widok szczegółów (Rysunek 17), edycji (Rysunek 18) i dodawania (Rysunek 19) nowej kategorii działa tak samo jak w przypadku artykułów.



Rysunek 15: Widok kategorii na komputerze stacjonarnym



Rysunek 16: Widok kategorii na telefonie komórkowym



Rysunek 17: Widok szczegółów kategorii

Edit

Name: Ore

Description: Ores that can be found in the Overworld

Submit

Rysunek 18: Widok edycji kategorii

Create

Name:

Description: Enter text...

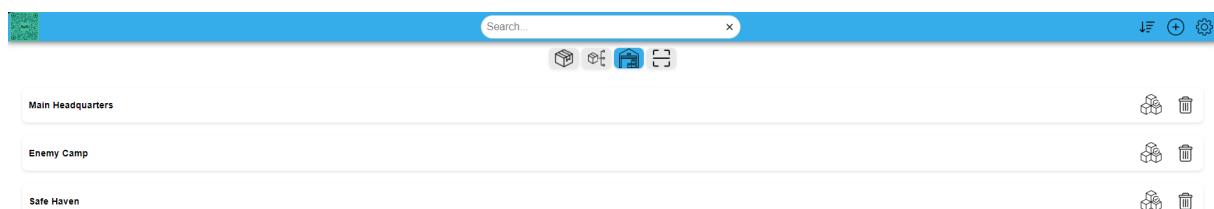
Submit

Rysunek 19: Widok dodawania kategorii

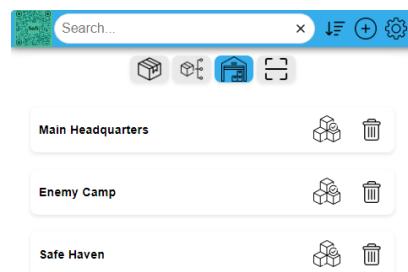
9.4.5 Magazyny

Strona magazynów przedstawiona na rysunku 20 pozwala na przegląd i dodawanie magazynów. Podobnie jak w przypadku artykułów i kategorii w pasku górnym znajdują się te same opcje. Także analogicznie do artykułów i kategorii wybór magazynów następuje poprzez nacisnięcie odpowiedniego elementu. Dodatkowo na każdym wpisie znajduje się przycisk, który przekierowuje do stanu danego magazynu.

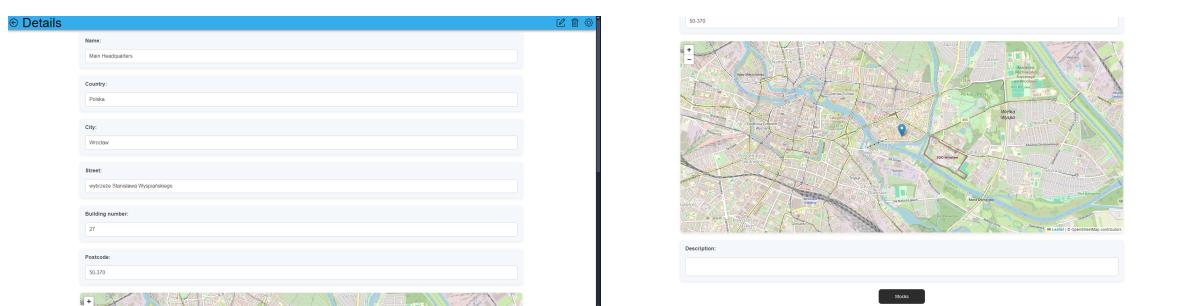
Widok szczegółów, edycji i dodawania nowej kategorii działa tak samo jak w przypadku artykułów i kategorii z dodatkiem interaktywnej mapy. Podczas wyświetlania szczegółów (Rysunek 22), mapa przedstawia wybraną lokalizację magazynu. Dodatkowo widok szczegółów zawiera przycisk, który przekierowuje bezpośrednio do stanu danego magazynu.



Rysunek 20: Widok magazynów na komputerze stacjonarnym

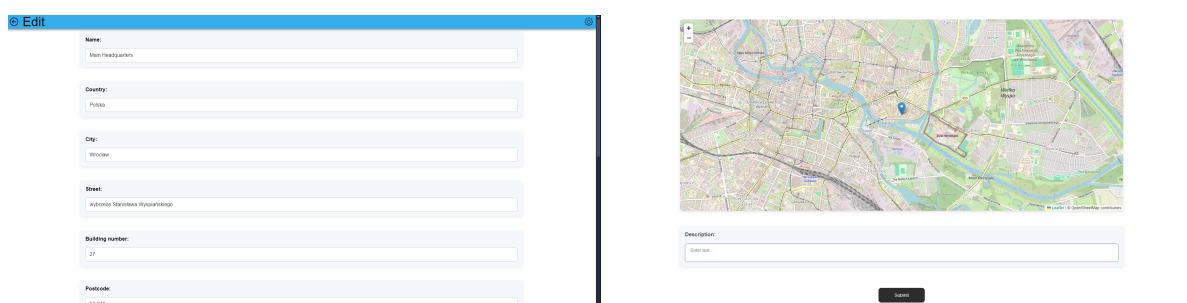


Rysunek 21: Widok magazynów na telefonie komórkowym



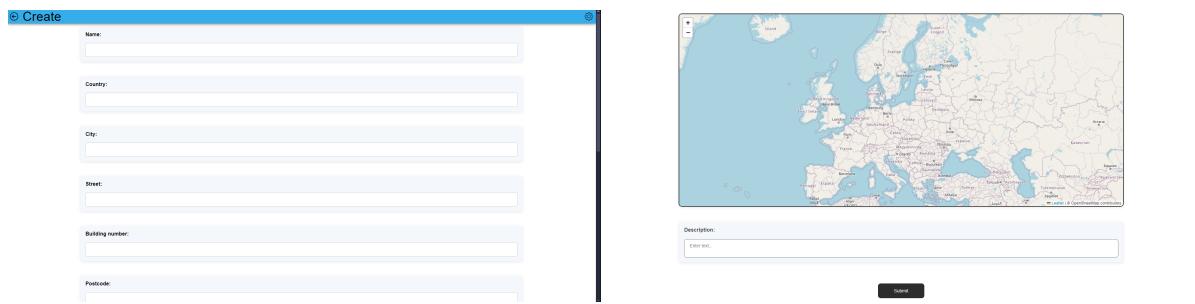
Rysunek 22: Widok szczegółów magazynu

Podczas edycji (Rysunek 23), mapa jest interaktywna i pozwala na wybór lokalizacji poprzez przeniesienie pinezki w wybrane miejsce. Wszystkie pola, które mogą być wypełnione, zostaną automatycznie zaktualizowane. Podobnie w przypadku wprowadzenia nowego adresu mapa zaktualizuje się aby odzwierciedlić zmiany.



Rysunek 23: Widok edycji magazynu

Podobnie jak w przypadku edycji, podczas dodawania nowego magazynu (Rysunek 24) mapa jest interaktywna i zachowuje się w ten sam sposób.



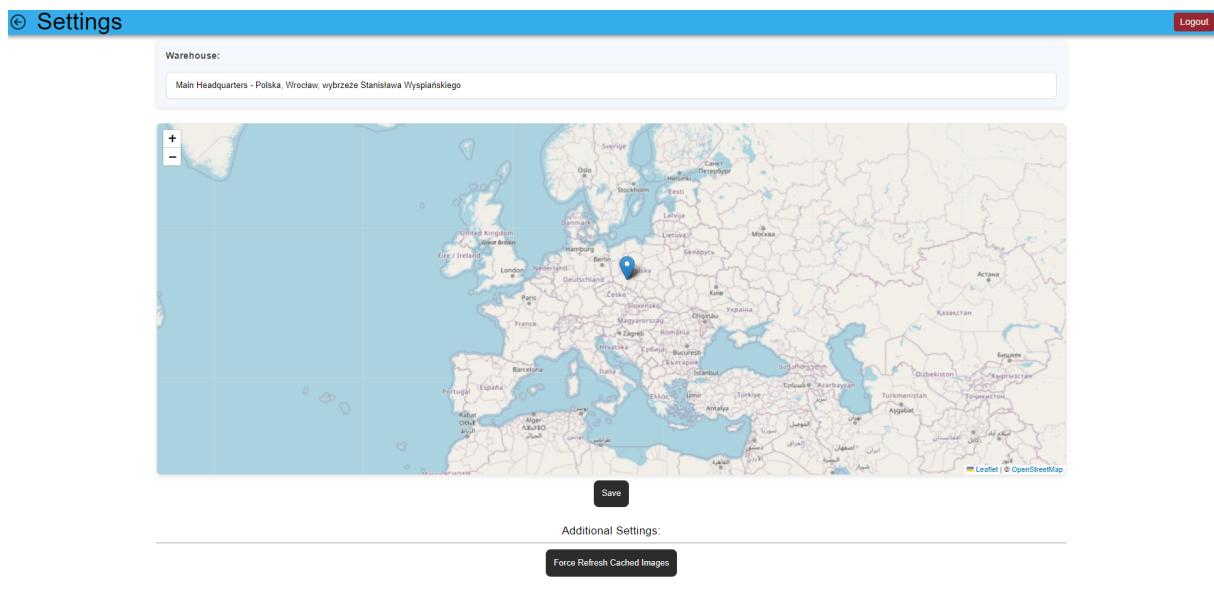
Rysunek 24: Widok dodawania magazynu

9.4.6 Ustawienia

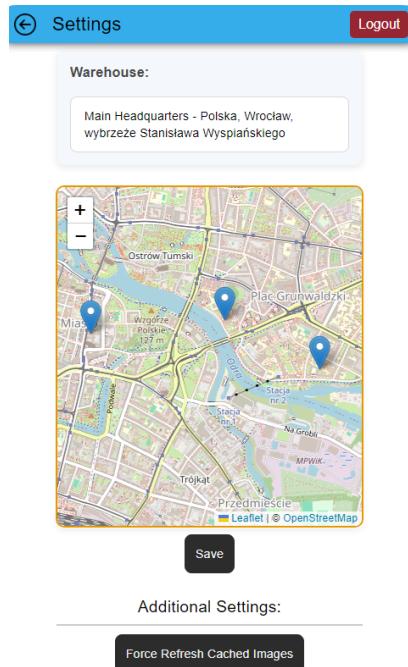
Widok ustawień (Rysunek 25) oferuje 3 funkcjonalności: ustalenie domyślnego magazynu, wylogowanie się z aplikacji oraz odświeżenie obrazków na wypadek jakichkolwiek problemów z funkcją cache. Przycisk wylogowania przekierowuje użytkownika na stronę startową i go wylogowuje. Odświeżenie obrazków powoduje wyczyszczenie wszystkich zapamiętanych obrazków i pozwala na ponowne pobranie ich z serwera.

Selekcja domyślnego magazynu przebiega poprzez wybranie jednego ze stworzonych wcześniej magazynu z listy, lub kliknięcie w odpowiednią pinezkę na mapie.

Po ustaleniu domyślnego magazynu w głównych widokach pojawi się nowa ikona oznaczająca stan domyślnego magazynu co przedstawia Rysunek 27.



Rysunek 25: Widok ustawień na komputerze stacjonarnym



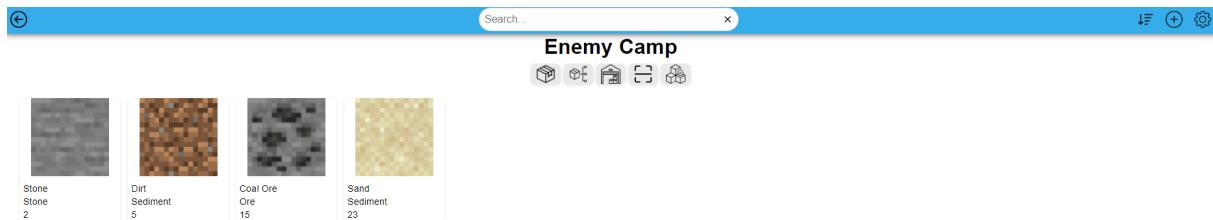
Rysunek 26: Widok ustawień na telefonie komórkowym



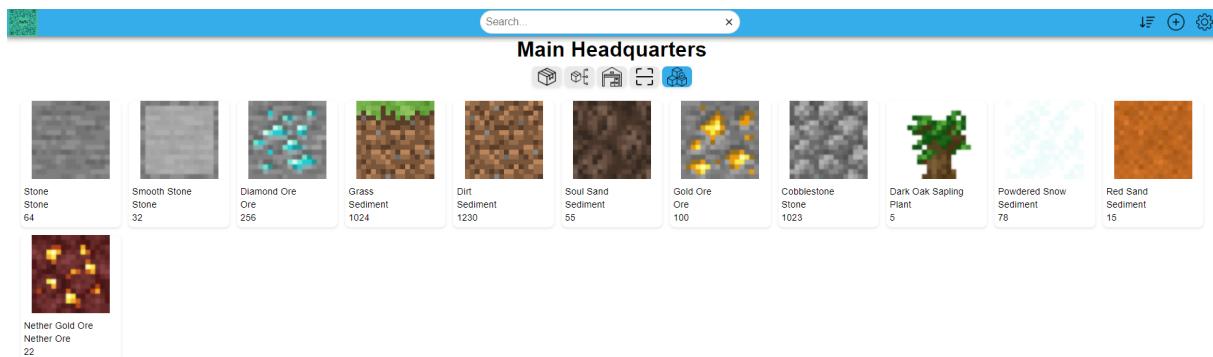
Rysunek 27: Widok magazynów po ustaleniu domyślnego magazynu

9.4.7 Stan magazynu

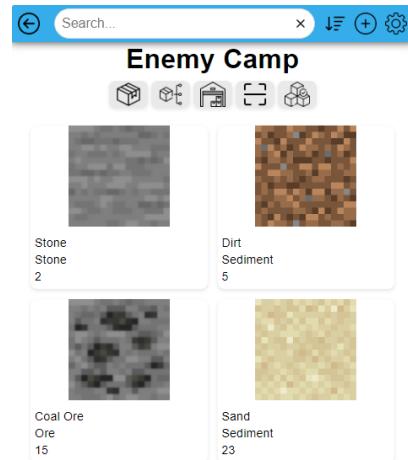
Widok stanu magazynu ma dwie postaci: dowolny magazyn (Rysunek 28) oraz magazyn domyślny (Rysunek 29), gdzie jedyną różnicą jest podświetlenie ikony domyślnego magazynu. Funkcje górnego paska są takie same jak w widoku artykułu, z tą różnicą, że przycisk dodawania pozwala na wprowadzenie nowego przedmiotu na stan magazynu. W widoku szczegółów (Rysunek 32) wyświetlane są informacje o przedmiocie, takie jak obrazek, nazwa, kategoria, opis i kod EAN13, nazwa aktualnego magazynu minimalna liczba sztuk oraz aktualna liczba sztuk danego przedmiotu. Minimalna liczba sztuk jest określana w celu zawiadamiania odpowiednich użytkowników w momencie kiedy stan danego przedmiotu w magazynie spadnie poniżej wymaganej wartości.



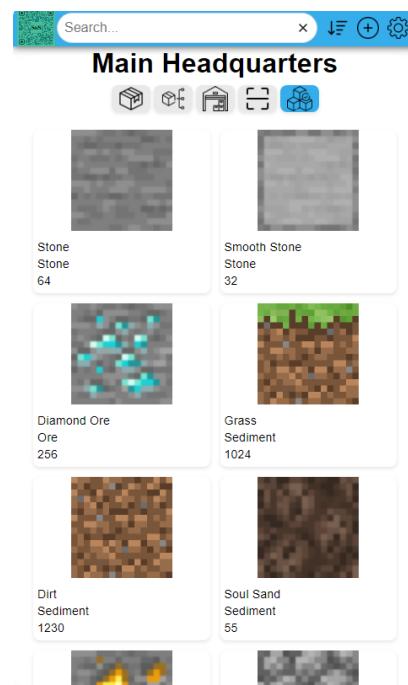
Rysunek 28: Widok stanu magazynu dla dowolnego magazynu na komputerze stacjonarnym



Rysunek 29: Widok stanu magazynu dla domyślnego magazynu na komputerze stacjonarnym



Rysunek 30: Widok stanu magazynu dla dowolnego magazynu na telefonie komórkowym



Rysunek 31: Widok stanu magazynu dla domyślnego magazynu na telefonie komórkowym

Details

Name: DHL

Category: Sediment

Description:

DHL

EAN13:

Warehouse: Stan magazynu

Minimum quantity:

Quantity: 1230

Rysunek 32: Widok szczegółów stanu magazynu

Edycja stanu magazynu przedstawiona na Rysunku 33 pozwala na określenie zmiany w liczbie sztuk danego przedmiotu. Zmiana może być dodatnia (wprowadzanie dodatkowych przedmiotów do magazynu) lub ujemna (wydawanie przedmiotów z magazynu). Zmiany można zatwierdzić lub anulować wybierając odpowiedni przycisk.

Edit

Name: DHL

Category: Sediment

Description:

DHL

EAN13:

Warehouse: Stan magazynu

Minimum quantity:

Changes:

Current quantity: -1230

Future quantity: 1230

Cancel Save

Rysunek 33: Widok edycji stanu magazynu

Dodawanie przedmiotu do stanu magazynu działa analogicznie do edycji przedmiotu na stanie magazynu z tą różnicą, że wszystkie pola, poza nazwą aktualnego magazynu, są wstępnie puste.

Add

No Image

Name: DHL

Category:

Description:

DHL

EAN13:

Warehouse: Stan magazynu

Minimum quantity:

Changes:

Current quantity: -9

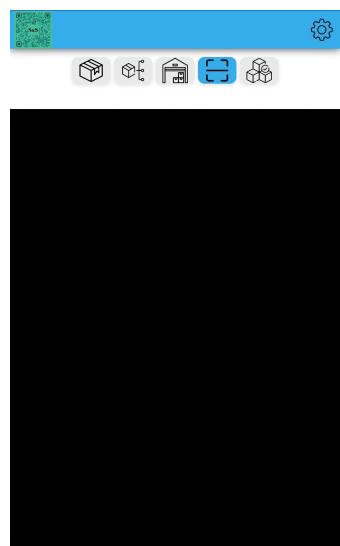
Future quantity: 9

Cancel Save

Rysunek 34: Widok dodawania stanu magazynu

9.4.8 Skaner

Widok skanera przedstawiony jest na Rysunku 35 i zawiera tylko obraz z kamery. Po zeskanowaniu odpowiedniego kodu użytkownik zostaje przekierowany do odpowiedniej strony: artykułu - jeśli zeskanował kod EAN13; stanu domyślnego magazynu - jeśli zeskanował kod QR z zawartym ID artykułu.



Rysunek 35: Widok skanera kodów kreskowych i kodów QR

LITERATURA

- [1] Matt Garman (CEO). Aws dokumentacja. <https://docs.aws.amazon.com>. Dostęp: 2024-11-30.
- [2] Perkins Miller (CEO). Strona internetowa z listą teksturow z gry minecraft. https://minecraft.fandom.com/wiki/List_of_block_textures. Dostęp: 2024-12-14.
- [3] Google. Angular dokumentacja. <https://angular.dev>. Dostęp: 2024-11-30.
- [4] HashiCorp. Hashicorp terraform dokumentacja. <https://developer.hashicorp.com/terraform/docs>. Dostęp: 2024-11-30.