Jakub Biernaczyk
Robert Głowacki
Krystian Kubiak
Piotr Piotrowski
Sławomir Wosek
Michał Wójcik 0000-0002-0934-7815
INSTYTUT MORSKI UNIWERSYTETU MORSKIEGO W GDYNI
adres e-mail do korespondencji: michal.wojcik@im.umg.edu.pl
DOI: 10.26408/FindFISH-11

11. ARCHITEKTURA ORAZ TECHNOLOGIE WYKORZYSTANE PODCZAS TWORZENIA PLATFORMY FINDFISH

WPROWADZENIE

W niniejszym rozdziale przedstawiono szczegółowy opis architektury systemu odpowiedzialnego za udostępnianie danych pomiarowych i modelowych w ramach platformy transferu wiedzy FindFISH. Głównymi założeniami dla architektury zrealizowanego systemu było wykorzystanie nowoczesnych technologii internetowych w złożonej wielomodułowej aplikacji opartej na rozwiązaniach dostępnych bezpłatnie na licencjach otwartego oprogramowania. System został skonstruowany w postaci kilku niezależnie uruchamianych modułów, z których każdy jest odpowiedzialny za inny zbiór funkcjonalności wymaganych od systemu. Moduły systemu zostały oparte na gotowych rozwiązaniach lub w całości zaimplementowane przez zespół projektowy. Zarówno gotowe rozwiązania, jak i biblioteki programistyczne użyte we własnych rozwiązaniach są dostępne bezpłatnie na otwartych licencjach. Cały system jest dostępny dla użytkowników za pośrednictwem interfejsu graficznego działającego z poziomu przeglądarki internetowej.

11.1. KOMPONENTY SYSTEMU

System realizujący udostępnianie danych pomiarowych i modelowych został zaprojektowany w taki sposób, aby składał się z kilku niezależnie uruchamianych komponentów (modułów) komunikujących się ze sobą za pomocą odpowiednich

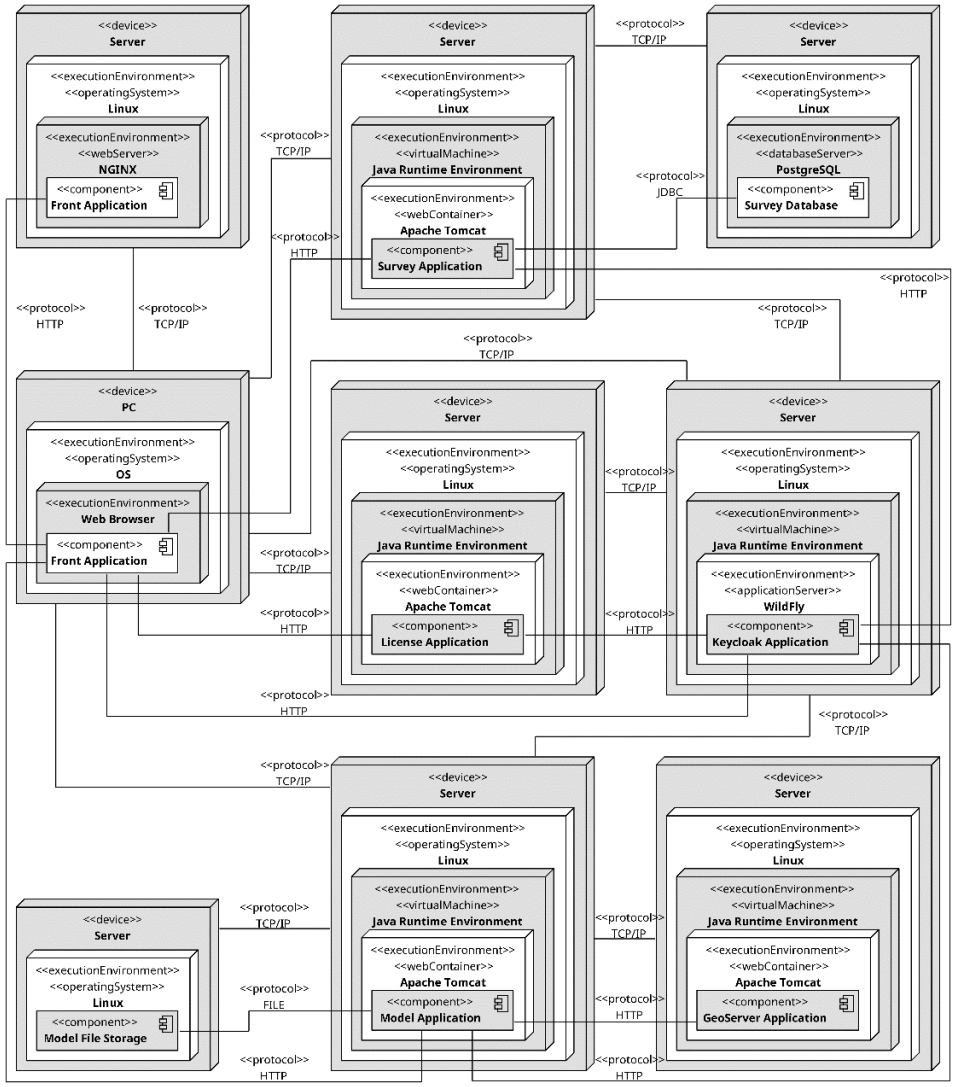
protokołów sieciowych. W założeniu każdy z komponentów realizuje inny zbiór logicznie powiązanych ze sobą funkcjonalności.

Podczas wyboru stosu technologicznego dla poszczególnych modułów kierowano się tym, aby były to rozwiązania oparte na technologiach bazujących na językach programowania Java (Gosling i in., 2018) oraz TypeScript (Microsoft Corporation, 2022a). Język Java jest powszechnie używany do implementacji aplikacji internetowych, zwłaszcza aplikacji serwerowych dostarczających logikę biznesową do systemu. Język TypeScript umożliwia przygotowanie kodu źródłowego wykorzystującego typowe wzorce obiektowe, który jest kompilowany do języka JavaScript (Mozilla Corporation, 2022). Z kolei język JavaScript jest powszechnie używany do tworzenia interaktywnych graficznych interfejsów użytkownika dostępnych z poziomu przeglądarek internetowych.

Drugim kryterium wyboru stosu technologicznego było wykorzystanie zarówno bibliotek programistycznych, jak i gotowych rozwiązań dostępnych bezpłatnie na licencjach otwartego oprogramowania (The Open Source Definition) (Open Source Initiative, 2007). Wykorzystanie otwartego oprogramowania daje możliwość dostosowania oprogramowania pod konkretne potrzeby (np. poprzez modyfikację kodu źródłowego) bez generowania dodatkowych kosztów w projekcie.

Na rys. 11.1 przedstawiono architekturę systemu w postaci diagramu wdrożenia. Zostały na nim zaznaczone wszystkie komponenty systemu udostępniania danych:

- Front Application – zapewnia graficzny interfejs użytkownika dostępny przez przeglądarkę internetową;
- License Application – pozwala na zarządzanie licencjami dostępu do danych;
- Survey Application – udostępnia dane pomiarowe w formacie prezentowanym za pomocą wykresów i tabel;
- Model Application – udostępnia dane modelowe w postaci warstw mapy oraz w formacie prezentowanym za pomocą wykresów i tabel;
- Survey Database – baza danych, w której przechowywane są dane pomiarowe;
- Model File Storage – zasób dyskowy, gdzie przechowywane są dane modelowe;
- GeoServer Application – udostępnia warstwy mapy;
- Keycloak Application – centralny serwer uwierzytelniania użytkowników.



Rys. 11.1. Diagram architektury systemu

Źródło: opracowanie własne.

11.1.1. Interfejs użytkownika

Komponent Front Application (aplikacja Front) został wykonany w całości przez zespół projektowy w technologii Angular (Google LLC, 2022a), która jest dostępna bezpłatnie na licencji otwartego oprogramowania The MIT License

(Massachusetts Institute of Technology, 1980). Aplikacja została opracowana z wykorzystaniem komponentów i serwisów zaimplementowanych w języku TypeScript oraz szablonów opisanych za pomocą języka znaczników HTML (HyperText Markup Language) (Web Hypertext Application Technology Working Group, 2022) oraz języka arkuszy stylów CSS (Cascading Style Sheets) (World Wide Web Consortium (W3C), 2008). Serwisy odpowiadają za logikę aplikacji, w szczególności za pobieranie danych oraz przekazywanie żądań z/do pozostałych aplikacji (License, Survey i Model). Komponenty są odpowiedzialne za przygotowanie danych prezentowanych w ramach interfejsu graficznego (potencjalnie danych pobranych za pomocą serwisu z innej aplikacji) oraz obsługę akcji użytkownika (np. kliknięcie przycisku pobierającego lub wysyłającego dane). Szablony definiują wygląd i układ graficznego interfejsu użytkownika.

Aplikacja jest automatycznie budowana (kompilacja źródeł w języku TypeScript do języka JavaScript, wykonanie testów jednostkowych i integracyjnych) za pomocą otwartego narzędzia npm (npm, Inc., 2022). Oznacza to, że aplikacja nie jest zależna od żadnego konkretnego zintegrowanego środowiska programistycznego (Integrated Development Environment, IDE) i może być w całości zbudowana z poziomu wiersza poleceń.

Aplikacje webowe wymagają środowiska uruchomieniowego w postaci serwera webowego udostępniającego aplikację za pośrednictwem protokołu HTTP (Hypertext Transfer Protocol) (Fielding i Reschke, 2014). Zastosowano tutaj bezpłatne rozwiązanie w postaci aplikacji NGINX Open Source (Nginx, Inc., 2022). Aplikacja NGINX została uruchomiona na serwerze w systemie operacyjnym Linux, ale może być uruchomiona także w innych systemach (Microsoft Windows, macOS).

Aplikacja Front jest w całości realizowana po stronie klienta (użytkownika) w ramach przeglądarki internetowej. Oznacza to, że klient za pomocą protokołu HTTP pobiera kod HTML, JavaScript i CSS, który jest następnie wykonywany w dowolnej nowoczesnej przeglądarce internetowej na dowolnym systemie operacyjnym. To właśnie z poziomu przeglądarki internetowej wysyłane są kolejne żądania do pozostałych aplikacji (License, Survey, Model i Keycloak) w celu pobrania danych, które zostaną zaprezentowane użytkownikowi.

Do implementacji elementów interfejsu użytkownika wykorzystano bibliotekę Angular Material (Google LLC, 2022b). Aplikacja pozwala na zaprezentowanie danych zarówno pomiarowych, jak i modelowych (w wybranym punkcie) za pomocą tabel i wykresów. Do implementacji mechanizmów prezentowania danych za pomocą wykresów została wykorzystana biblioteka ng2-charts (Valor Software, 2022). Główną funkcjonalnością jest prezentowanie danych modelowych w postaci warstw mapy. W celu zaimplementowania mechanizmu wyświetlania warstw wykorzystano bibliotekę OpenLayers (Open Source Geospatial Foundation, 2022c).

Fundusze Europejskie Program Regionalny | Rzeczpospolita Polska | URZĄD MARSZAŁKOWSKI WOJEWÓDZTWA POMORSKIEGO | Unia Europejska Europejski Fundusz Rozwoju Regionalnego

Użytkownik wybiera model, zmienną, punkt w czasie oraz głębokość (jeśli zmienna modelu posiada wymiar głębokości), a odpowiednia warstwa jest pobierana za pomocą usługi WMS (Web Map Service) (Open Geospatial Consortium (OGC), 2006) będącej standardową usługą OGC (Open Geospatial Consortium (OGC), 2022). Użytkownikom z uprawnieniami administratora aplikacja umożliwia dodatkowo zarządzanie licencjami dostępu do danych.

Widoki wyświetlające dane zostały zabezpieczone na poziomie aplikacji na podstawie poświadczeń pobieranych z komponentu Keycloak Application. Oczywiście wszelkie zabezpieczenia na poziomie warstwy widoku służą głównie poprawnemu wyświetleniu interfejsu zawierającego elementy, do których użytkownik posiada odpowiednie uprawnienia.

11.1.2. Aplikacje udostępniające dane

Komponenty License Application, Model Application i Survey Application zostały wykonane w całości przez zespół projektowy w technologii Spring Boot (VMware, Inc., 2022a), która jest dostępna bezpłatnie na licencji otwartego oprogramowania Apache License 2.0 (The Apache Software Foundation, 2004). Aplikacje zostały zrealizowane z wykorzystaniem kontrolerów, serwisów i repozytoriów zaimplementowanych w języku Java. Kontrolery implementują usługi sieciowe zgodne ze stylem architektonicznym REST (Representational State Transfer) (Richardson i Ruby, 2007) przy użyciu biblioteki Spring MVC (VMware, Inc., 2022e). Zadanie kontrolerów polega na odebraniu żądania wysłanego za pomocą protokołu HTTP, przetłumaczeniu go na model danych wykorzystywany w logice biznesowej i przekazaniu do odpowiedniego serwisu. Serwisy realizują logikę biznesową aplikacji. W przypadku gdy jakaś operacja wymaga interakcji z zewnętrznym komponentem (inna aplikacja, baza danych, system plików itd.), sterowanie jest przekazywane do odpowiedniego repozytorium realizującego operacje na zewnętrznych danych.

Wszystkie operacje w aplikacjach zostały zabezpieczone, tak aby dostęp do danych mieli tylko użytkownicy z ważnymi licencjami, a dostęp do informacji o użytkownikach i przypisanych do nich licencjach miał tylko administrator. Informacje o użytkownikach są pobierane z komponentu Keycloak Application. Do mechanizmów uwierzytelniania i autoryzacji wykorzystano bibliotekę Spring Security (VMware, Inc., 2022d).

Aplikacje są automatycznie budowane (kompilacja źródeł w języku Java do kodu maszynowego, wykonanie testów jednostkowych i integracyjnych, przygotowanie paczki dystrybucyjnej w formacie JAR) za pomocą otwartego narzędzia Apache Maven (The Apache Software Foundation, 2022b). Oznacza to, że aplikacja nie jest zależna od żadnego konkretnego zintegrowanego środowiska

programistycznego (IDE) i może być w całości zbudowana z poziomu wiersza poleceń.

Aplikacje webowe wykonane w technologii Spring Boot potrzebują środowiska uruchomieniowego w postaci kontenera webowego. Zastosowano tutaj rozwiązanie dostępne na licencji otwartego oprogramowania Apache Tomcat (The Apache Software Foundation, 2022c). Nie ma potrzeby instalacji ani konfiguracji kontenera webowego, ponieważ zostaje on zintegrowany w paczkę dystrybucyjną na etapie budowania projektu. Podobnie jak opisywane aplikacje, aplikacja Apache Tomcat jest napisana w języku Java i wymaga do działania środowiska uruchomieniowego w postaci maszyny wirtualnej języka Java (Java Virtual Machine, JVM) (Lindholm i in., 2018). W projekcie wykorzystano implementację OpenJDK (Oracle Corporation, 2022). Maszynę wirtualną uruchomiono w systemie operacyjnym Linux, ale może ona być uruchomiona także w innych wspieranych systemach operacyjnych (Microsoft Windows, macOS).

Komponent License Application (aplikacja License) służy do zarządzania licencjami użytkowników. Licencja użytkownika umożliwia dostęp do danych modelowych i pomiarowych. Oznacza to, że użytkownik po rejestracji nie ma od razu dostępu do danych i musi poczekać na przydzielenie odpowiedniej licencji przez administratora. Wszystkie licencje na dostęp do danych w systemie są czasowe, co oznacza, że wymagają od administratora zdefiniowania końcowej daty ważności. Informacje o użytkownikach i licencjach są przechowywane w komponencie Keycloak Application, będącym centralnym serwerem uwierzytelniania i autoryzacji, a sama aplikacja License dostarcza usługi sieciowe do zarządzania licencjami. Oznacza to, że zadaniem aplikacji License jest udostępnienie usług sieciowych spójnych z pozostałymi komponentami udostępniającymi dane (Model Application i Survey Application), które mogą być konsumowane w komponencie Front Application.

Komponent Survey Application (aplikacja Survey) służy do zarządzania danymi pomiarowymi zebranymi podczas wypraw rybackich. Komplet danych zebranych podczas pojedynczej wyprawy składa się z trzech plików: danych źródłowych zebranych sondą CTD (*Conductivity, Temperature, Depth*) (podczas realizacji projektu wykorzystano sondę Midas CTD+ 300), ścieżki GPS (Global Positioning System) (Blewitt, 2016) kutra rybackiego w formacie XML – Extensible Markup Language (World Wide Web Consortium (W3C), 2008) z wykorzystaniem schematu GPX – GPS Exchange Format (TopoGrafix, 2004) oraz ankiety rybackiej opisującej warunki pogodowe, stan morza, lokalizację i wielkość połowu w formacie DOCX – Word Extensions to the Office Open XML (Microsoft Corporation, 2022b). Dane zebrane podczas wypraw mogą być wprowadzone za pomocą graficznego interfejsu użytkownika z poziomu aplikacji Front.

Po wprowadzeniu i przetworzeniu przez aplikację dane są przechowywane w relacyjnej bazie danych. Dostęp do bazy danych jest realizowany z użyciem technologii JPA (Java Persistence API) (VMware, Inc., 2022c) umożliwiającej automatyczne odwzorowanie obiektów języka Java na wiersze w bazie danych i przesyłanie ich za pomocą sterownika JDBC (Java Database Connectivity) (Oracle Corporation, 2017). Rola sterownika JDBC polega na tłumaczeniu wywołań programistycznych na natywne zapytania odpowiednie dla konkretnego serwera baz danych. Do implementacji repozytoriów przesyłających żądania do bazy danych wykorzystano bibliotekę Spring Data (VMware, Inc., 2022b). Dane przechowywane w bazie danych są dostępne za pomocą usług sieciowych zgodnych ze stylem architektonicznym REST.

Komponent Model Application (aplikacja Model) służy do zarządzania danymi pochodzącymi z modeli dostępnych w ramach platformy (model Fish dla każdego poławianego gatunku oraz model EcoFish złożony z modeli hydrodynamicznego i biochemicznego). Dane modelowe są przechowywane w systemie plików w plikach binarnych w formacie NetCDF (Network Common Data Form) (Unidata, 2022a). Na podstawie plików modelowych aplikacja Model za pomocą usług sieciowych zgodnych ze stylem architektonicznym REST udostępnia informacje o dostępnych modelach, zmiennych, zakresach dat oraz głębokości. Dla każdej zmiennej możliwe jest pobranie danych ze wskazanego punktu (współrzędnych geograficznych) oraz okresu w celu zaprezentowania ich za pomocą tabeli lub wykresu. Do odczytywania danych z plików w formacie NetCDF wykorzystano bibliotekę programistyczną netCDF-Java (Unidata, 2022b).

Główną funkcjonalnością aplikacji jest udostępnianie warstw mapy za pomocą usługi sieciowej WMS (Web Map Service). Sama implementacja usługi WMS znajduje się w komponencie GeoSever Application a aplikacja Model pełni rolę pośrednika w przekazywaniu żądań. W przypadku gdy warstwa żądana przez klienta istnieje w aplikacji GeoServer, jest ona zwraca. W przeciwnym wypadku aplikacja Model na podstawie odpowiedniego pliku w formacie NetCDF tworzy obraz w formacie GeoTiff i rejestruje go w aplikacji GeoServer. Do generowania obrazów w formacie GeoTiff wykorzystano bibliotekę GeoTools (Open Source Geospatial Foundation, 2022b).

11.1.3. Gotowe rozwiązania

Komponent Survey Database (baza Survey) jest relacyjną bazą danych uruchomioną na otwartym serwerze baz danych PostgreSQL (The PostgreSQL Global Development Group, 2022) dostępnym na licencji PostgreSQL License (The PostgreSQL Global Development Group, 2010). Serwer PostgreSQL to powszechnie wykorzystywany serwer baz danych wyposażony w wydajne wsparcie

dla danych przestrzennych dzięki rozszerzeniu PostGIS 3.3.2dev Manual (Open Source Geospatial Foundation, 2022d).

Komponent Model File Storage reprezentuje przestrzeń dyskową, na której dostępne są dane modelowe w postaci plików binarnych w formacie NetCDF. Przestrzeń dyskowa może się znajdować na tym samym urządzeniu (serwerze) co aplikacja Model albo w przestrzeni masowej (np. macierz dyskowa) i być podmontowana jako zasób w systemie plików systemu operacyjnego, na którym jest uruchomiona aplikacja Model.

Komponent GeoServer Application (aplikacja GeoServer) służy do udostępniania warstw mapy za pomocą usługi WMS. Wykorzystano tutaj gotowe rozwiązanie w postaci aplikacji GeoServer (Open Source Geospatial Foundation, 2022a), dostępne na otwartej licencji GNU General Public License 2.0 (Stallman, 1991) i napisane w języku Java. Podobnie jak aplikacje License, Survey i Model, wymaga ona do działania środowiska wykonawczego w postaci kontenera webowego. Analogicznie jak w aplikacjach wytworzonych przez zespół projektowy, zastosowano tutaj kontener webowy Apache Tomcat.

Aplikacja GeoServer nie jest udostępniona klientom bezpośrednio, ale za pośrednictwem aplikacji Model. Oznacza to, że aplikacja Front nie łączy się bezpośrednio z aplikacją GeoServer, a zamiast tego wysyła za pośrednictwem protokołu HTTP żądania zgodne z usługą WMS do aplikacji Model. Dopiero aplikacja Model przekazuje żądanie do aplikacji GeoServer, a w przypadku gdy żądana warstwa mapy nie istnieje, najpierw tworzy obraz w formacie GeoTiff i rejestruje go jako nową warstwę w aplikacji GeoServer.

Komponent Keycloak Application (aplikacja Keycloak) jest centralnym modułem jednokrotnego logowania (Single Sign On, SSO). Wykorzystano tutaj gotowe rozwiązanie w postaci aplikacji Keycloak (RedHat, 2022), dostępne na otwartej licencji Apache License 2.0 (The Apache Software Foundation, 2004) i napisane w języku Java. Aplikacja Keycloak ze względu na wykorzystanie technologii Jakarta EE Platform (Jakarta Enterprise Edition), wcześniej Java EE (Java Enterprise Edition) (Eclipse Foundation, 2019), wymaga pełnego serwera aplikacji, w przeciwieństwie do wcześniej omawianych aplikacji, które wymagały jedynie kontenera webowego. Jako serwer aplikacji został wykorzystany serwer WildFly zintegrowany z aplikacją Keycloak.

Aplikacja Keycloak jest odpowiedzialna za proces rejestracji i uwierzytelniania użytkowników. W celu zachowania kompatybilności z aplikacją Front interfejs użytkownika wykonany w technologii Apache FreeMarker (The Apache Software Foundation, 2022a) został odpowiednio zmodyfikowany przez zespół projektowy. Dzięki integracji aplikacji Keycloak z technologiami Spring Boot i Angular wystarczy, że użytkownik zaloguje się raz, a przez czas trwania sesji użytkownika będzie miał dostęp do wszystkich komponentów systemu. Aplikacja może być

również zintegrowana z aplikacją GeoServer, w sytuacji gdyby konieczne było udostępnienie jej bezpośrednio użytkownikom.

11.2. UDOSTĘPNIANIE DANYCH

System udostępniania danych modelowych i pomiarowych realizuje udostępnianie danych na kilka sposobów:

- licencje użytkowników – zarządzanie za pomocą zaprojektowanej usługi typu REST,
- dane pomiarowe – udostępnianie za pomocą zaprojektowanej usługi typu REST;
- dane modelowe – udostępnianie za pomocą zaprojektowanej usługi typu REST;
- warstwy mapy – udostępnianie za pomocą usługi Web Map Service.

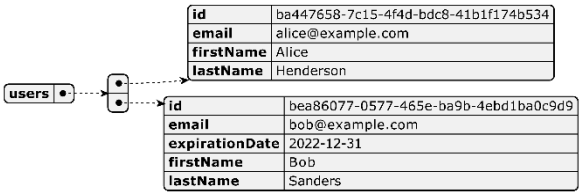
Dzięki zastosowaniu usług sieciowych zgodnych ze stylem architektonicznym REST, dostępnych za pośrednictwem protokołu http, nie ma konieczności korzystania z graficznego interfejsu użytkownika udostępnianego przez aplikację Front do pobierania danych udostępnianych przez system. Po uwierzytelnieniu w aplikacji Keycloak zgodnie ze standardowym protokołem uwierzytelniania OAuth 2.0 (Parecki, 2020) użytkownik może skorzystać z dowolnej usługi sieciowej udostępnianej w ramach systemu. W szczególności użytkownik może przygotować własne skrypty pobierające dane lub własny interfejs graficzny wyświetlający wyłącznie interesującą go prognozę dla konkretnego modelu i zmiennej w formie warstwy mapy.

Aplikacja License udostępnia usługi sieciowe pozwalające na pobranie wszystkich użytkowników zarejestrowanych w systemie i skonfigurowanie dla każdego z osobna daty wygaśnięcia licencji dostępu do danych. Dane wysyłane i pobierane do i z usługi sieciowej są w formacie JSON – JavaScript Object Notation (Bray, 2017). Opis usług udostępnionych przez aplikację License znajduje się w tabeli 11.1. Struktury żądania i odpowiedzi zostały przedstawione na rysunkach wskazanych w tabeli.

Tabela 11.1

Usługi sieciowe udostępnione przez aplikację License

Metoda	Zasób	Opis	Struktura
GET	/users	Pobranie listy wszystkich użytkowników	Rys. 11.2
PUT	/users/{uuid}/expirationDate	Ustawienie ważności licencji dostępu do danych dla konkretnego użytkownika, przy czym {uuid} to identyfikator konkretnego użytkownika	Rys. 11.3



Rys. 11.2. Struktura odpowiedzi z listą użytkowników

Źródło: opracowanie własne.

Rys. 11.3. Struktura żądania wydłużenia licencji

Źródło: opracowanie własne.

Aplikacja Survey udostępnia usługi sieciowe pozwalające na przesyłanie nowych oraz pobieranie istniejących danych pomiarowych. Opis usług udostępnionych przez aplikację Survey znajduje się w tabeli 11.2. Struktury żądania i odpowiedzi zostały przedstawione na rysunkach wskazanych w tabeli.

Tabela 11.2

Usługi sieciowe udostępnione przez aplikację Survey

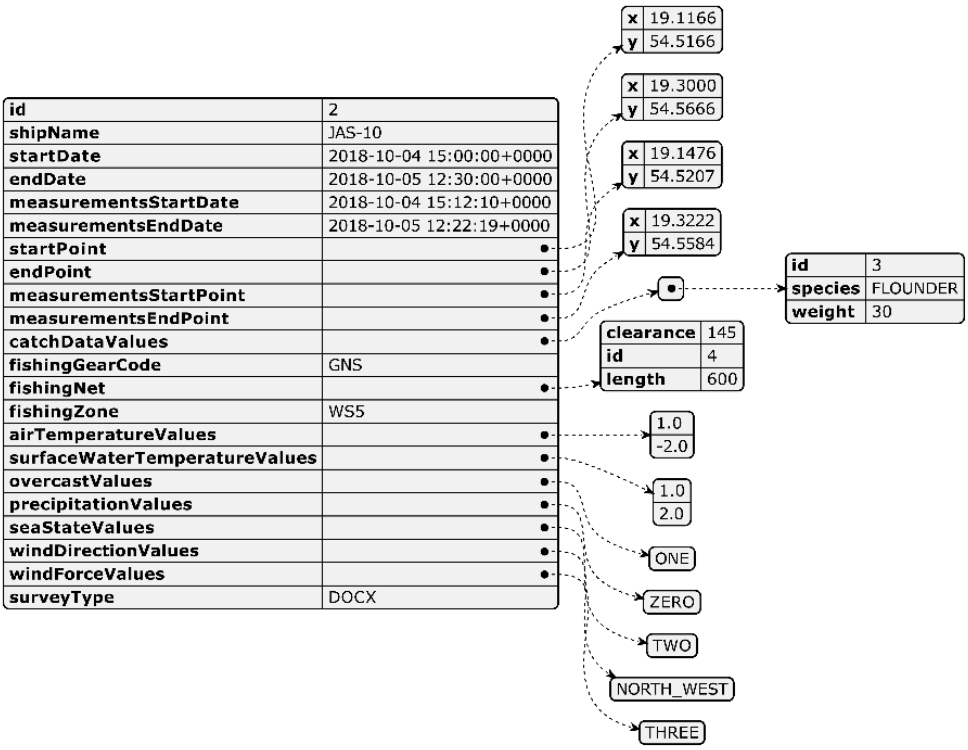
Metoda	Zasób	Opis	Struktura
GET	/voyages/count	Pobranie liczby wszystkich wypraw rybackich	Rys. 11.4
GET	/voyages	Pobranie identyfikatorów wypraw rybackich. Zasób wspiera stronicowanie (parametry żądania limit i offset)	Rys. 11.5
GET	/voyages/{id}	Pobranie informacji o konkretnej wyprawie, przy czym {id} to identyfikator wyprawy. Odpowiedź zawiera zawartość wypełnionej ankiety rybackiej, a także daty i współrzędne odczytane z plików przesłanych razem z ankietą	Rys. 11.6
GET	/voyages/{id}/ surveys	Pobranie danych zebranych przez sondę CTD dla konkretnej wyprawy	Rys. 11.7
GET	/voyages/{id}/ track	Pobranie śladu GPS dla konkretnej wyprawy rybackiej	Rys. 11.8
POST	/voyages	Wgranie nowych danych w postaci pliku źródłowego z sondy CTD, pliku XML ze śladem GPS oraz pliku DOCX z ankietą rybacką	

Rys. 11.4. Struktura odpowiedzi z liczbą wypraw rybackich

Źródło: opracowanie własne.

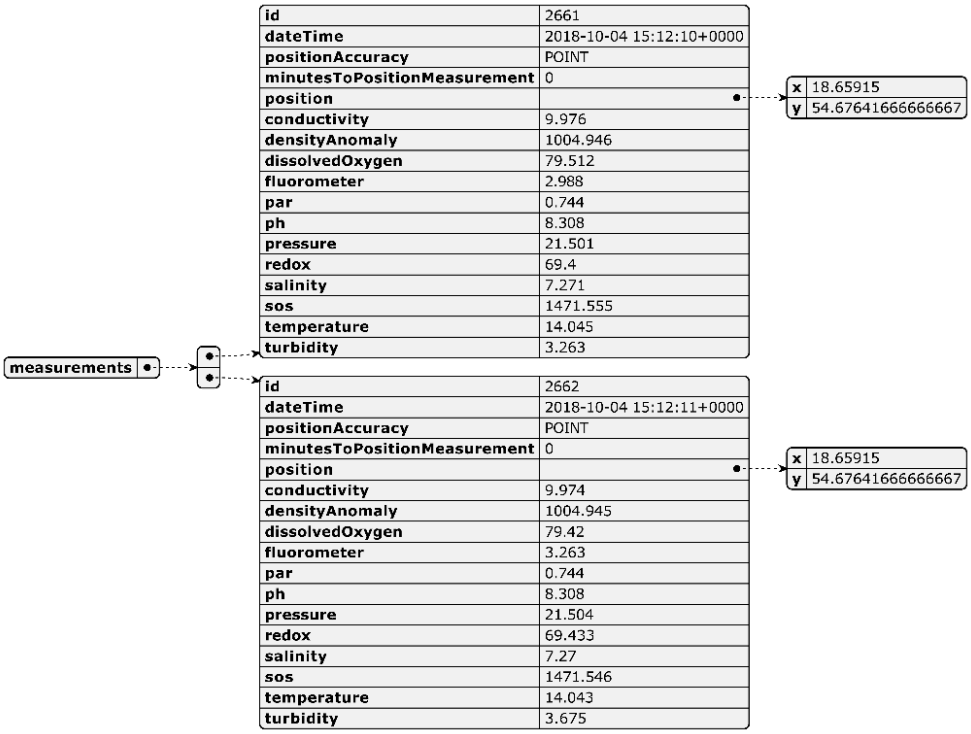
Rys. 11.5. Struktura odpowiedzi z listą identyfikatorów wypraw rybackich

Źródło: opracowanie własne.



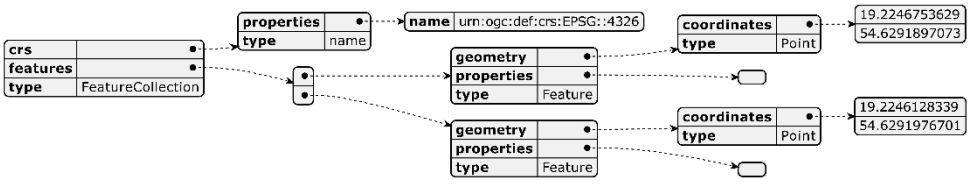
Rys. 11.6. Struktura odpowiedzi z informacjami o konkretnej wyprawie rybackiej

Źródło: opracowanie własne.



Rys. 11.7. Struktura odpowiedzi z danymi zebranymi przez sondę CTD

Źródło: opracowanie własne.



Rys. 11.8. Struktura odpowiedzi ze śladem GPS

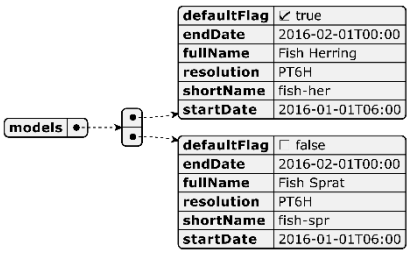
Źródło: opracowanie własne.

Aplikacja Model udostępnia usługi sieciowe pozwalające na pobieranie danych modelowych. Opis usług udostępnionych przez aplikację Model znajduje się w tabeli 11.3. Struktury żądania i odpowiedzi zostały przedstawione na rysunkach wskazanych w tabeli.

Tabela 11.3

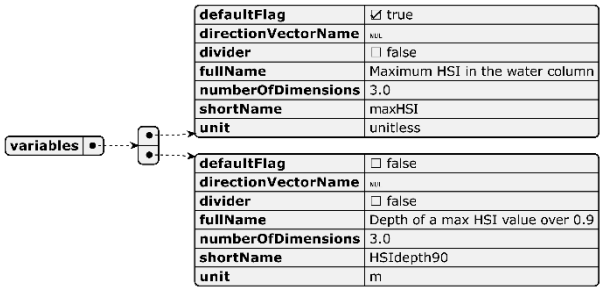
Usługi sieciowe udostępnione przez aplikację Model

Metoda	Zasób	Opis	Struktura
GET	/models	Pobranie listy wszystkich dostępnych modeli. Odpowiedź zawiera informacje o zakresie dat i rozdzielczości czasowej modelu	Rys. 11.9
GET	/models/{model}/variables	Pobranie listy zmiennych wybranego modelu, przy czym {model} to skrócona nazwa modelu. Odpowiedź zawiera informację o wymiarach i jednostce parametru	Rys. 11.10
GET	/models/{model}/variables/{var}/data	Wartości dla wskazanego parametru modelu we wskazanym punkcie, przy czym {model} to skrócona nazwa modelu, a {var} to skrócona nazwa parametru. Parametry żądania: kat, lon, from, to, elevation	Rys. 11.11
GET	/models/{model}/wms	Adres usługi WMS dla wybranego modelu. Parametry żądania zgodnie z dokumentacją usługi OGC WMS. Nazwa zmiennej modelu przekazywana jako nazwa warstwy	



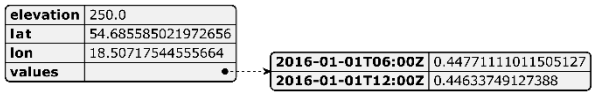
Rys. 11.9. Struktura odpowiedzi z listą dostępnych modeli

Źródło: opracowanie własne.



Rys. 11.10. Struktura odpowiedzi z listą dostępnych zmiennych modelu

Źródło: opracowanie własne.



Rys. 11.11. Struktura odpowiedzi z danymi modelowymi dla konkretnego punktu

Źródło: opracowanie własne.

Powyżej zostały opisane usługi sieciowe udostępniane przez system udostępniania danych modelowych i pomiarowych w ramach platformy FindFISH. Pokazuje to, że system został przygotowany zgodnie z architekturą mikrousług – cały system składa się z wielu luźno powiązanych, mniejszych usług, które można wdrażać niezależnie od siebie (Richardson, 2018). Pozwala to na niezależne rozwijanie konkretnych funkcjonalności bez ingerencji w pozostałe elementy systemu. Ponadto zastosowanie usług sieciowych (zarówno zaprojektowanych usług typu REST, jak i standardowych usług OGC WMS) umożliwia zintegrowanie systemu z innymi systemami, a także pozwala użytkownikom na wykorzystanie własnych narzędzi pobierania i prezentacji danych.

11.3. KONTENERYZACJA

W celu usprawnienia procesów zarówno wytwarzania, jak i wdrażania systemu wykorzystano mechanizm konteneryzacji. Kontenerami nazywa się wykonywalne jednostki oprogramowania. Aplikacja wraz ze swoimi zależnościami (bibliotekami programistycznymi) jest umieszczona w kontenerze w taki sposób, aby umożliwić uruchomienie w dowolnym środowisku (komputer klasy PC, serwer, chmura itd.) (IBM Cloud Education, 2021). Do tworzenia i uruchamiania kontenerów wykorzystano darmowe narzędzie Docker CE (Docker Inc., 2022). Dla każdego komponentu systemu (Front Application, License Application, Survey Application, Model Application, Keycloak Application, GeoServer Application i Survey Database) tworzony jest obraz kontenera, który następnie jest umieszczany w repozytorium obrazów. Jako repozytorium obrazów wykorzystano aplikację Nexus (Sonatype, 2022). W celu uruchomienia konkretnego modułu wystarczy za pomocą poleceń narzędzia Docker CE pobrać odpowiedni obraz z repozytorium i uruchomić kontener.

Oprócz kontenerów przygotowano także konfigurację wdrożenia całego systemu za pomocą pliku YAML (YAML Ain't Markup Language) (Ben-Kiki i in., 2009), korzystając ze składni narzędzia Docker Compose (Docker Inc., 2022). Dzięki temu cała konfiguracja aplikacji tworzących system oraz zależności między nimi zawarta jest w pojedynczym pliku konfigurującym całe wdrożenie. Taka konfiguracja jest wykorzystywana w środowisku zarówno deweloperskim, jak i produkcyjnym.

PODSUMOWANIE

W rozdziale zaprezentowano szczegółowy opis architektury systemu odpowiedzialnego za udostępnianie danych pomiarowych i modelowych w ramach platformy transferu wiedzy FindFISH. W szczególności zostały opisane wszystkie komponenty systemu, zarówno te zaimplementowane w całości przez zespół projektowy, jak i te oparte na gotowych rozwiązaniach. Dla każdego komponentu podano wykorzystane języki programowania, gotowe aplikacje oraz najistotniejsze biblioteki programistyczne.

Mimo że system jest dostępny przede wszystkim z poziomu przeglądarki internetowej, to dane mogą być też pobierane automatycznie za pomocą odpowiednich usług sieciowych. W rozdziale opisano zarówno wykorzystanie standardowych usług OGC, jak i dokładną strukturę usług zaimplementowanych na potrzeby systemu przez zespół projektowy.

Zastosowanie gotowych rozwiązań oraz bibliotek programistycznych dostępnych bezpłatnie na licencjach otwartego oprogramowania umożliwi łatwy rozwój systemu w przyszłości bez konieczności inwestowania środków finansowych w zakup odpowiednich licencji oprogramowania. Dodatkowo zastosowanie architektury opartej na mikrousługach pozwoli na niezależne rozwijanie poszczególnych komponentów, a także dodawanie nowych. Ponadto zastosowanie mechanizmów konteneryzacji zapewnia łatwe uruchomienie systemu w dowolnej infrastrukturze informatycznej.

LITERATURA

1. Ben-Kiki O., Evans C., döt Net I., *YAML Ain't Markup Language (YAML™) version 1.2* [online], 2009, https://yaml.org/spec/1.2/spec.html [30.09.2022].
2. Blewitt G., *GPS, reference systems* [online], Springer International Publishing, Switzerland 2016, https://nbmg.unr.edu/staff/pdfs/Blewitt_Encyclopedia_of_Geodesy.html [30.09.2022].
3. Bray T., *The JavaScript Object Notation (JSON) Data Interchange Format* [online], RFC Editor, 2017, https://rfc-editor.org/rfc/rfc8259.txt [30.09.2022].
4. Docker Inc., *Docker Reference documentation* [online], 2022, https://docs.docker.com/reference/ [30.09.2022].
5. Eclipse Foundation, *Jakarta EE Platform, version 8* [online], 2019, https://jakarta.ee/specifications/platform/8/platform-spec-8.html [30.09.2022].
6. Fielding R.T., Reschke J., *Hypertext Transfer Protocol (HTTP/1.1): semantics and content* [online], RFC Editor, 2014, https://rfc-editor.org/rfc/rfc7231.txt [30.09.2022].
7. Google LLC, *Angular Docs* [online], 2022a, https://angular.io/docs [30.09.2022].
8. Google LLC, *Angular Material* [online], 2022b, https://material.angular.io/ [30.09.2022].

9. Gosling J., Joy B., Steele G., Bracha G., Buckley A., Smith D., *The Java® Language Specification Java SE 11 Edition* [online], 2018, https://docs.oracle.com/javase/specs/jls/se11/html/index.html [30.09.2022].
10. IBM Cloud Education, *Containerization* [online], 2021, https://www.ibm.com/cloud/learn/containerization [30.09.2022].
11. Lindholm T., Yellin F., Bracha G., Buckley A., Smith D., *The Java® Virtual Machine Specification Java SE 11 Edition* [online], 2018, https://docs.oracle.com/javase/specs/jvms/se11/html/ [30.09.2022].
12. Massachusetts Institute of Technology, *MIT License* [online], 1980, https://choosealicense.com/licenses/mit/ [30.09.2022].
13. Microsoft Corporation, *TypeScript Documentation* [online], 2022a, https://www.typescriptlang.org/docs/ [30.09.2022].
14. Microsoft Corporation, *Word Extensions to the Office Open XML (.docx) File Format* [online], 2022b, https://interoperability.blob.core.windows.net/files/MS-DOCX/%5bMS-DOCX%5d.pdf [30.09.2022].
15. Mozilla Corporation, *JavaScript* [online], 2022, https://developer.mozilla.org/en-US/docs/Web/JavaScript [30.09.2022].
16. Nginx, Inc., *nginx documentation* [online], 2022, https://nginx.org/en/docs/ [30.09.2022].
17. npm, Inc., *npm Docs* [online], 2022, https://docs.npmjs.com// [30.09.2022].
18. Open Geospatial Consortium (OGC), *About OGC* [online], 2022, https://www.ogc.org/about [30.09.2022].
19. Open Geospatial Consortium (OGC), *Web Map Service* [online], 2006, https://www.ogc.org/standards/wms [30.09.2022].
20. Open Source Geospatial Foundation, *GeoServer documentation* [online], 2022a, https://docs.geoserver.org/ [30.09.2022].
21. Open Source Geospatial Foundation, *GeoTools documentation* [online], 2022b, https://docs.geotools.org/ [30.09.2022].
22. Open Source Geospatial Foundation, *OpenLayers documentation* [online], 2022c, https://openlayers.org/doc/ [30.09.2022].
23. Open Source Geospatial Foundation, *PostGIS 3.3.2dev Manual* [online], 2022d, https://postgis.net/docs/manual-3.3/ [30.09.2022].
24. Open Source Initiative, *The Open Source definition* [online], 2007, https://opensource.org/osd [30.09.2022].
25. Oracle Corporation, *JDBC™ 4.3 Specification* [online], 2017, https://download.oracle.com/otn-pub/jcp/jdbc-4_3-mrel3-spec/jdbc4.3-fr-spec.pdf [30.09.2022].
26. Oracle Corporation, *OpenJDK* [online], 2022, https://openjdk.org/ [30.09.2022].
27. Parecki A., *The Little Book of OAuth 2.0 RFCs*, 2020.
28. RedHat, *Keycloak 12.0 documentation* [online], 2020, https://www.keycloak.org/archive/documentation-12.0.html [30.09.2022].
29. Richardson C., *Microservices patterns with examples in Java*, Manning, 2018.
30. Richardson L., Ruby S., *Restful Web Services*, O'Reilly, 2007.
31. Sonatype, *Repository Manager 3* [online], 2022, https://help.sonatype.com/repomanager3 [30.09.2022].

32. Stallman R., *GNU General public license v2.0* [online], 1991, https://choosealicense.com/licenses/gpl-2.0/ [30.09.2022].
33. The Apache Software Foundation, *Apache FreeMarker tutorial* [online], 2022a, https://freemarker.apache.org/docs/index.html [30.09.2022].
34. The Apache Software Foundation, *Apache license 2.0* [online], 2004, https://choosealicense.com/licenses/apache-2.0/ [30.09.2022].
35. The Apache Software Foundation, *Apache Maven Project – documentation* [online], 2022b, https://maven.apache.org/guides/index.html [30.09.2022].
36. The Apache Software Foundation, *Apache Tomcat 9 – documentation* [online], 2022c, https://tomcat.apache.org/tomcat-9.0-doc/index.html [30.09.2022].
37. The PostgreSQL Global Development Group, *PostgreSQL 13.8 documentation* [online], 2022, https://www.postgresql.org/docs/13/index.html [30.09.2022].
38. The PostgreSQL Global Development Group, *PostgreSQL license* [online], 2010, https://choosealicense.com/licenses/postgresql/ [30.09.2022].
39. TopoGrafix, *GPX 1.1 Schema documentation* [online], 2004, https://www.topografix.com/GPX/1/1/ [30.09.2022].
40. Unidata, *NetCDF 4.9.0* [online], 2022a, https://docs.unidata.ucar.edu/netcdf-c/current/ [30.09.2022].
41. Unidata, *NetCDF-Java tutorial and reference documentation 5.5* [online], 2022b, https://docs.unidata.ucar.edu/netcdf-java/current/userguide/index.html [30.09.2022].
42. Valor Software, *ng2-charts* [online], 2022, https://valor-software.com/ng2-charts/ [30.09.2022].
43. VMware, Inc., *Spring Boot reference documentation* [online], 2022a, https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/ [30.09.2022].
44. VMware, Inc., *Spring Data Commons – reference documentation* [online], 2022b, https://docs.spring.io/spring-data/commons/docs/current/reference/html/ [30.09.2022].
45. VMware, Inc., *Spring Data JPA – reference documentation* [online], 2022c, https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ [30.09.2022].
46. VMware, Inc., *Spring Security* [online], 2022d, https://docs.spring.io/spring-security/reference/index.html [30.09.2022].
47. VMware, Inc., *Web on Servlet Stack* [online], 2022e, https://docs.spring.io/spring-framework/docs/current/reference/html/web.html [30.09.2022].
48. Web Hypertext Application Technology Working Group, *HTML living standard* [online], 2022, https://html.spec.whatwg.org/multipage/ [30.09.2022].
49. World Wide Web Consortium (W3C), *CSS Snapshot 2021 W3C Working Group Note* [online], 2021, https://www.w3.org/TR/css-2021/ [30.09.2022].
50. World Wide Web Consortium (W3C), *Extensible Markup Language (XML) 1.0 (Fifth Edition)* [online], 2008, https://www.w3.org/TR/xml/ [30.09.2022].

Praca wykonana w ramach projektu „Platforma transferu wiedzy FindFISH – Numeryczny System Prognozowania warunków środowiska morskiego Zatoki Gdańskiej dla Rybołówstwa” (nr RPPM.01.01.01-22-0025/16-00) współfinansowanego ze środków Europejskiego Funduszu Rozwoju Regionalnego w ramach Regionalnego Programu Operacyjnego Województwa Pomorskiego na lata 2014–2020.