Grzegorz RYBAK[1], Krzysztof StRZECHA[2], Łukasz STURGULEWSKI[3], Marek KRAKOS[4], Dominik SANKOWSKI[5]

[1,2,3,5] Instytut Informatyki Stosowanej, Politechnika Łódzka
[4] Oddział Chirurgii i Urologii Dziecięcej Szpital im. J. Korczaka w Łodzi
ORCID: [1] 0000-0003-1250-2115, [2] 0000-0002-5648-0942, [3,] 0000-0003-1163-7165, [4] 0000-0001-6408-4328 [5] 0000-0003-2223-6690

doi:10.15199/48.2021.12.29

Aspekty architektoniczne tworzenia cyfrowej platformy gromadzenia danych medycznych

Streszczenie. *Artykuł porusza zagadnienia związane z projektowaniem komputerowej platformy gromadzenia danych medycznych dedykowanej pomiarom uzyskanym przy zastosowaniu nowych metod diagnostycznych. Do wskazanych technik diagnostycznych należą Tomografia Ultradźwiękowa oraz Tomografia Impedancyjna. W artykule przedstawione zostały główne problemy architektoniczne i sposoby realizacji oprogramowania uwzględniając opis koncepcji oraz opis stosu technologicznego.*

Abstract*. The article discusses issues related to the design of a computer platform for collecting medical data dedicated to measurements obtained with the use of new diagnostic methods. The indicated diagnostic techniques include Ultrasound Tomography and Impedance Tomography. The article presents the main architectural problems and methods of software implementation, taking into account the description of the concept and the description of the technological stack (**Architectural aspects of creating a digital medical data collection platform**).*

Słowa kluczowe: układ moczowy, tomografia, rezonans magnetyczny, DICOM, mikroserwisy, IoT, broker wiadomości.
Keywords: urinary system, tomography, magnetic resonance, DICOM, microservices, IoT, message broker.

Wstęp

Zastosowanie cyfrowych technik pomiarowych, takich jak tomografia komputerowa i rezonans magnetyczny, pozwala wcześnie wykryć zmiany chorobowe, a przez co uchronić pacjenta przed rozwojem schorzeń mogących zagrażać jego życiu. Metody te często są podstawowymi narzędziami obrazowania w diagnostyce medycznej. Występują jednak schorzenia, do diagnozowania których nie istnieją wystarczające rozwiązania techniczne. Metody badań lekarskich schorzeń urologicznych często są inwazyjne oraz nie pozwalają na długotrwały monitoring układu moczowego. Problem może dotyczyć nawet 20% populacji dzieci do 5 roku życia.

Zaproponowano utworzenie komputerowego systemu diagnostycznego opartego na dotychczas niestosowanych w urologii rozwiązaniach nieinwazyjnych, do których należą Tomografia Ultradźwiękowa i Elektryczna Tomografia Impedancyjna. Dostarczenie infrastruktury, dzięki której działalność medyczna w zakresie opisanym powyżej, będzie zarówno możliwa jak i efektywna, wiązało się z potrzebą opracowania cyfrowej platformy zarządzania danymi medycznymi. Niniejszy artykuł podejmuje tematykę kształtowania architektury rozwiązania, propozycji stosu technologicznego, struktury danych oraz interfejsu użytkownika.

W celu utworzenia platformy cyfrowej gromadzenia danych medycznych Autorzy zaproponowali zastosowanie architektury mikroserwisów. Rozwiązania tego typu mają liczne zalety, takie jak: uniwersalność, elastyczność, skalowalność czy mobilność. Stanowiło to podstawę wyboru stosu technologicznego rozwiązania. Opisane podejście wraz z zastosowaniem brokerów wiadomości wpada również w nowatorską ścieżkę tworzenia sieci komunikujących się ze sobą obiektów w przestrzeni cyfrowej dając w rezultacie realizację wizji IoT (Internet of Things).

Styl architektoniczny mikroserwisów

Architektura mikroserwisów (mikrousług) stanowi strukturalny wariant architektury zorientowanej na usługi (ang. *service-oriented architecture* - SOA) aranżując aplikację jako zbiór luźno powiązanych usług. Cechy charakterystyczne stylu architektonicznego mikroserwisów to: usługi w architekturze mikroserwisów (ang. microservice architecture - MSA) to często procesy, które komunikują się w sieci, w celu osiągnięcia celu przy użyciu protokołów niezależnych od technologii, takich jak http [1,2]. Usługi są zorganizowane wokół własności biznesowych [3]. Usługi mogą być wdrażane przy użyciu różnych języków programowania, baz danych, sprzętu i środowiska oprogramowania, w zależności od tego, jaka technologia pozwala na najefektywniejszą implementację [4]. Usługi są: małe, obsługujące wiadomości, ograniczone kontekstami, rozwijane autonomicznie, niezależnie wdrażane [4,5], zdecentralizowane, budowane i udostępniane za pomocą zautomatyzowanych procesów [5]. Ze względu na dużą liczbę usług w porównaniu z rozwiązaniami monolitycznymi, zdecentralizowane ciągłe dostarczanie oraz wdrożenie metodyki DevOps (ang. *development and operations* – metodyka zespolenia rozwoju i eksploatacji oraz zapewnienia jakości) z holistycznym monitorowaniem usług są niezbędne do skutecznego tworzenia, utrzymywania i obsługi takich aplikacji [6]. Istnieje wiele korzyści z dekompozycji aplikacji na wiele mniejszych usług, do których należą:

- modułowość [4],
- skalowalność [7],
- integracja heterogenicznych i starszych systemów [1,2,8] przy zastosowaniu podejścia przyrostowego [9],
- rozproszony rozwój poprzez faktoryzację [10] oraz usprawnienie procesów dostarczania [11].

Podejście mikroserwisów ma też wady takie jak:

- usługi tworzą bariery informacyjne.
- połączenia między usługami za pośrednictwem sieci mają wyższy koszt pod względem opóźnienia transmisji i czasu przetwarzania wiadomości niż wywołania w ramach monolitycznego procesu usługowego.
- testowanie oraz wdrażanie są bardziej skomplikowane. Przenoszenie funkcjonalności między serwisami jest trudniejsze [4].
- postrzeganie rozmiaru usług jako głównego mechanizmu strukturyzacji może prowadzić do zbyt wielu usług, gdy alternatywa polegająca na wewnętrznej modularyzacji może prowadzić do prostszego projektu. Wymaga to użycia narzędzi pomagających zrozumieć

ogólną architekturę aplikacji i współzależności między komponentami [12].

– protokół zwykle używany z mikroserwisami (HTTP) został zaprojektowany dla usług publicznych i jako taki nie nadaje się do pracy z wewnętrznymi mikroserwisami, które często muszą wykazywać wysoki poziom niezawodności [13].

Mikrousługi komputerowe mogą być implementowane w różnych językach programowania i mogą wykorzystywać różne infrastruktury. Dlatego najważniejszym wyborem technologicznymi jest sposób, w jaki mikrousługi komunikują się ze sobą (synchroniczna, asynchroniczna, integracja UI) oraz wybór protokołów używanych do komunikacji (RESTful HTTP, messaging, GraphQL). W tradycyjnym systemie większość wyborów technologicznych, takich jak język programowania, ma wpływ na cały system. Dlatego podejście do wyboru technologii jest zupełnie inne [14].

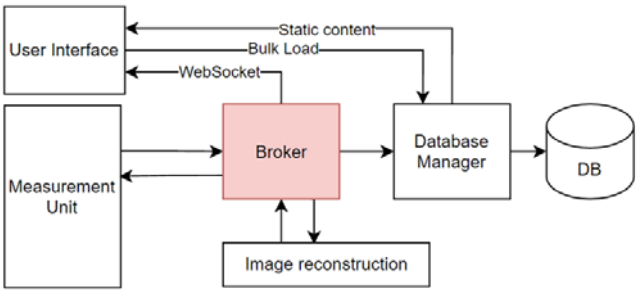
W architekturze z brokerem wiadomości cała komunikacja jest przekierowana przez grupę brokerów. Są to programy, które działają w oparciu o zaawansowane algorytmy routingu. Każdy mikroserwis jest podłączony do brokera. Mikroserwis może wysyłać i otrzymywać wiadomości przez to samo połączenie. Serwis wysyłający informacje to wydawca (ang. *publisher*), a odbiorcę określa się mianem subskrybenta (ang. *subscriber*). Wiadomości są publikowane w konkretnym kanale tematycznym. Subskrybent otrzymuje takie wiadomości z kanałów, które subskrybował.

Obecnie na rynku dostępnych jest szereg brokerów wiadomości zapewniających różne spektrum możliwości i funkcjonalności. Systemy brokerów wiadomości spełniające kryteria darmowej licencji i dużej popularności to: Apache Kafka, Mosquito, Hive, RabbitMQ, ActiveMQ, Apollo JMS oraz wiele innych.

Zastosowanie brokerów wpada również w nowatorską ścieżkę tworzenia sieci komunikujących się ze sobą obiektów w przestrzeni wirtualnej dając w rezultacie realizację wizji IoT (Internet of Things). Jej ideą jest zapewnienie możliwości wymiany komunikatów pomiędzy fizycznymi obiektami przy kontroli lub bez nadzoru człowieka (również przy zastosowaniu innych technologii jak LTE).

Architektura i stos technologiczny

W trakcie realizacji prac, zaproponowano koncepcję architektury cyfrowej platformy gromadzenia danych medycznych. Do głównych elementów rozwiązania zaliczyć można jednostkę pomiarową (ang. measurement unit), broker wiadomości, menedżer bazy danych, komponent obrazowania i wizualizacji danych pomiarowych oraz bazę danych. Powyższa propozycja stanowi podstawę kształtowania komputerowego mechanizmu gromadzenia danych medycznych. Koncepcję zaprezentowano na rysunku 1.



Rys. 1. Schemat koncepcyjny rozwiązania architektonicznego cyfrowej platformy gromadzenia danych medycznych.

Zaproponowana architektura nie wprowadza ograniczeń przy określeniu stosu technologicznego poza wymogiem dostosowania warstwy komunikacyjnej. Możliwe jest utworzenie wielu komponentów współpracujących ze sobą przy zastosowaniu protokołu WebSocket. Przygotowane rozwiązanie komputerowe może być wdrożone zarówno na systemie Windows jak i Linux.

Rozwiązanie architektoniczne cyfrowej platformy gromadzenia danych medycznych bazuje na wzorcu MVVC. Podejście takie pozwala na odseparowanie warstwy logiki aplikacji od warstwy prezentacji. Dodatkowo zapewnia podział pomiędzy strukturą danych komunikatów wewnętrznych jak i wiadomości wysyłanych i pozyskiwanych warstwy prezentacji danych.

Stos technologiczny obejmuje środowisko programistyczne i uruchomieniowe Java JDK w wersji 16.0.1. Jako kontener aplikacji wykorzystano Apache Tomcat w wersji 9.0. W celu gromadzenia danych wybrana została baza danych PostgreSQL (wersja 13) wraz z narzędziem administracyjnym PgAdmin (wersja 4.13). Wykorzystanie bazy danych wiąże się z przygotowaniem oraz utrzymaniem schematów bazodanowych. Aby usprawnić pracę ze schematami oraz ich organizację użyto narzędzia Liquibase 3.4.2.

Kolejnym elementem systemu jest broker wiadomości stanowiący główną część architektury. Dzięki możliwości zastosowania wtyczki (ang. plugin) pozwalającej na wymianę wiadomości za pomocą protokołu WebSocket, oraz dzięki potwierdzonej efektywności wybrano profesjonalne oprogramowanie Mosquitto 2.0.11.

Dane wysyłane pomiędzy komponentami platformy zgodne są ze standardem JSON. Zastosowano również liczne biblioteki webowego interfejsu graficznego (jquery.1.1.0.js, angular-1.5.8.js, angular-material.min-1.1.0.js).

Tabela 1. Stos technologiczny platformy gromadzenia danych medycznych

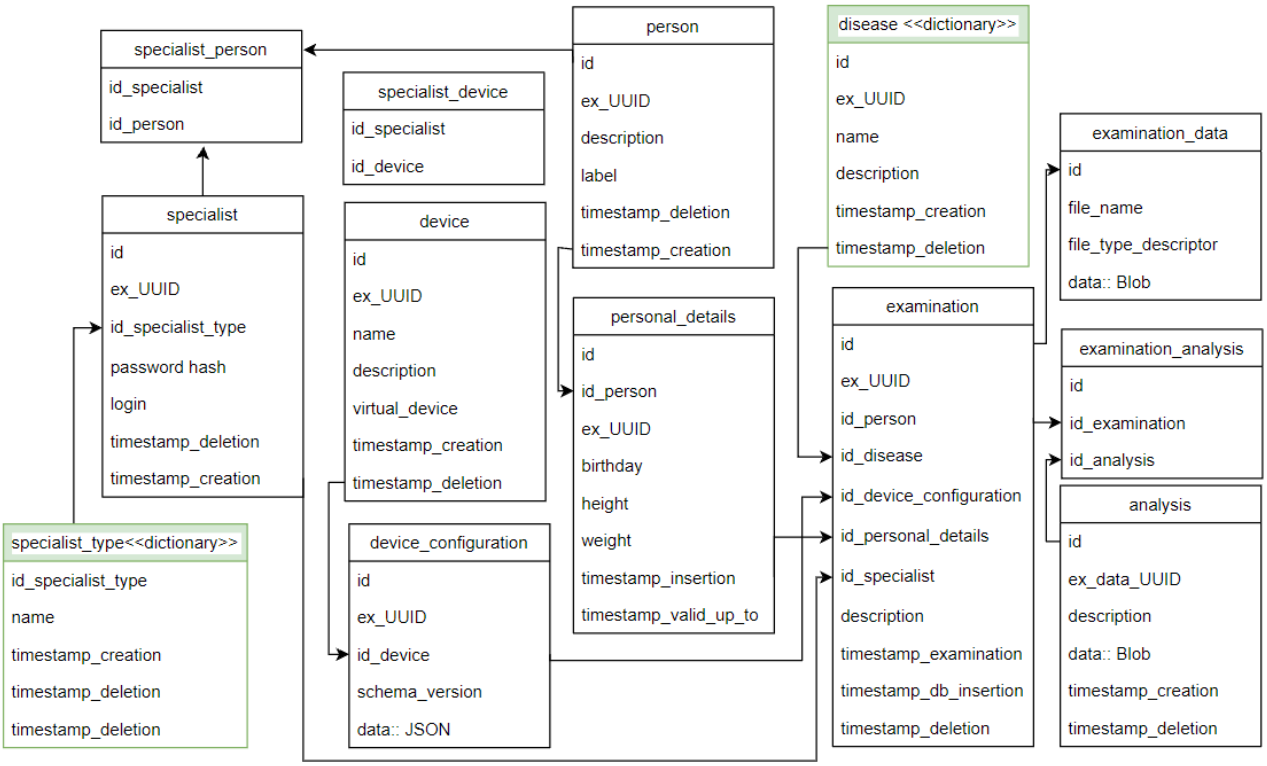
Lp.	Opis	Nazwa	Wersja
1	System operacyjny	Windows/Linux	
2	Środowisko uruchomieniowe	Java JDK	16.0.1
3	Kontener aplikacji	Tomcat	9.0
4	Baza danych	PostgreSQL	13
5	Narzędzie administracji bazy danych	PgAdmin	4.13
6	Narzędzie wspierające zarządzanie schematem bazy danych	Liquibase	liquibase-3.4.2
7	Broker wiadomości	Mosquitto	2.0.11
8	Format wymiany danych	JSON	-
9	Biblioteki tworzenia frontend'u (JavaScript)	jquery angular angular-material.min	1.1.0 1.5.8 1.1.0

Struktura bazy danych

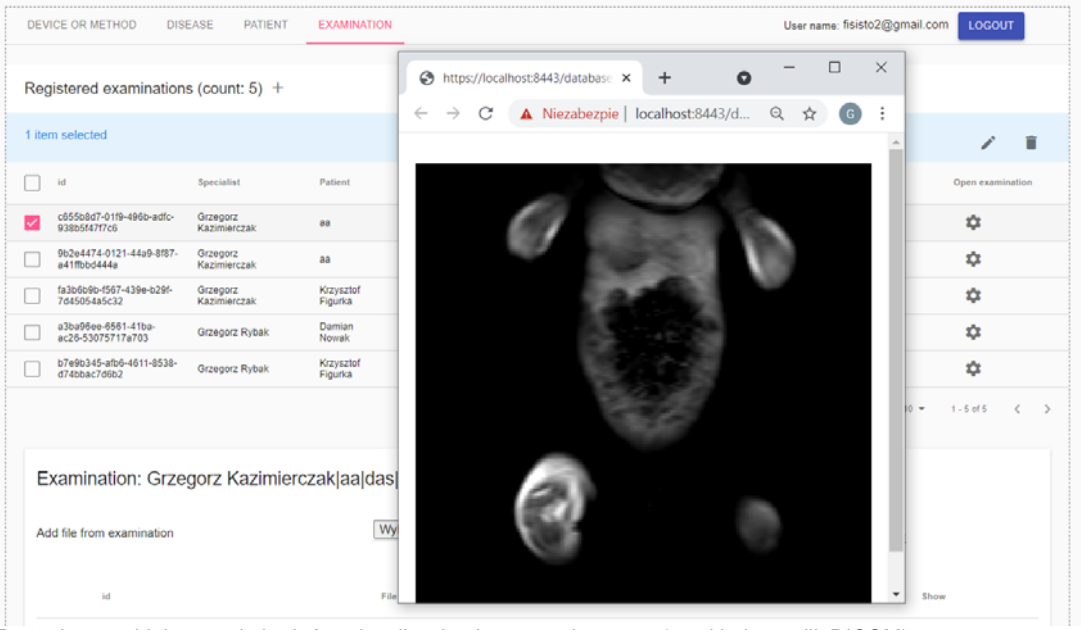
Struktura bazy danych została przedstawiona na rysunku 2. Diagram opisuje między innymi tabele danych: specjalisty, urządzenia pomiarowego, choroby, pacjenta, karty pacjenta oraz badań. Zaproponowany sposób gromadzenia danych medycznych pozwala na wykonanie badań w czasie rzeczywistym, strumieniując dane poprzez brokera wiadomości jak i pozwala na wysłanie pliku z pomiarami do systemu.

Możliwe jest przejście podstawowego scenariusza zapisu danych pochodzących z badania medycznego. Specjalista posiada konto w systemie, na które loguje się przed przystąpieniem do wykonania procedury. Na pierwszej karcie wyświetlone zostają informacje dotyczące dostępnych urządzeń pomiarowych. Specjalista

ma możliwość wyboru pożądanego narzędzia lub ma również możliwość wprowadzenia nowego rekordu do bazy danych. Po wybraniu urządzenia, wskazane zostaje schorzenie oraz wybrany zostaje pacjent, którego dotyczy usługa medyczna. Specjalista wykonuje badanie i umieszcza wyniki, przy zastosowaniu ostatniej zakładki (examination), w bazie danych w formacie DICOM.



Rysunek 2. Schemat bazy danych systemu gromadzenia informacji medycznych.



Rysunek 3. Zrzut ekranu widoku panelu badań z wizualizacją obrazu medycznego (przykładowy plik DICOM)

Wynik działania opracowanego systemu

Przykładowy widok panelu badań medycznych zaprezentowany został na rysunku 3. Interfejs użytkownika składa się z szeregu zakładek dzięki którym możliwe jest efektywne zarządzanie informacją dotyczącą pacjenta oraz danymi pomiarowymi. Zakładki odpowiadają tabelom w prezentowanym wyżej schemacie agregując: panel urządzenia pomiarowego, panel choroby, panel pacjenta oraz panel badania medycznego. Obraz medyczny pozyskano za pomocą metody rezonansu magnetycznego. Dane przechowywane są w pliku typu DICOM, który jest standardem gromadzenia tomograficznych danych medycznych.

Podsumowanie

W artykule przedstawiono celowość stosowania cyfrowych platform gromadzenia i przetwarzania danych

medycznych. Dane te pochodzą z rezonansu magnetycznego oraz docelowo nowoczesnych metod obrazowania układu moczowego.

W szczególności, uwagę skupiono na pomiarach z zastosowaniem technik tomograficznych. Wskazano główne zalety stosowanego stylu architektonicznego mikroserwisów wraz z opisem stosu technologicznego wybranego do realizacji części eksperymentalnej.

Finalnie przedstawiono wynik działania opracowanej, zaimplementowanej i wdrożonej cyfrowej platformy przetwarzania i gromadzenia danych medycznych w zastosowaniu do diagnostyki urologicznej.

Badania realizowane w ramach Programu Operacyjnego Inteligentny Rozwój Unii Europejskiej POIR.04.01.04-00-0045/20.

***Autorzy**: dr inż. Grzegorz Rybak, Politechnika Łódzka, Instytut Informatyki Stosowanej, 90-537 Łódź, ul. Stefanowskiego 18/22, E-mail: grzegorz.rybak@p.lodz.pl; dr inż. Krzysztof Strzecha, Politechnika Łódzka, Instytut Informatyki Stosowanej, 90-537 Łódź, ul. Stefanowskiego 18/22, E-mail: krzysztof.strzecha@p.lodz.pl; dr inż. Łukasz Sturgulewski, Politechnika Łódzka, Instytut Informatyki Stosowanej, 90-537 Łódź, ul. Stefanowskiego 18/22, E-mail: lukasz.sturgulewski@p.lodz.pl; dr n. m. Marek Krakós, Instytut Centrum Zdrowia Matki Polki; E-mail: marek.krakos@p.lodz.pl; prof. dr hab. inż. Dominik Sankowski, Politechnika Łódzka, Instytut Informatyki Stosowanej, 90-537 Łódź, ul. Stefanowskiego 18/22, E-mail: dominik.sankowski@p.lodz.pl;*

LITERATURA

[1] Newman S., Building microservices : designing fine-grained systems, O'Reilly Media, 2015
[2] Wolff E., Microservices : flexible software architecture, 2019
[3] Pautasso C, Zimmermann O, Amundsen M, Lewis J and Josuttis N 2017 Microservices in Practice, Part 1: Reality Check and Service Design IEEE Software, 2017
[4] Chen L., Microservices: Architecting for Continuous Delivery and DevOps 2018 IEEE International Conference on Software Architecture (ICSA) (IEEE), 2018
[5] Nadareishvili I., Mitra R., McLarty M. and Amundsen M. Microservice architecture : Aligning principles, practices, and culture, ISBN 1491956259, 2016
[6] Richardson C. and Safari an O. M. C. Microservices Patterns Video Edition, Manning Publications, 2018
[7] Dragoni N., Lanese I., Larsen S. T., Mazzara M, Mustafin R and Safina L. 2018 Microservices: How To Make Your Application Scale (Springer, Cham) pp 95–104, 2018
[8] Knoche H. and Hasselbring W. 2019 Drivers and Barriers for Microservice Adoption – A Survey among Professionals in Germany Enterp. Model. Inf. Syst. Archit. 14 1:1–35-1:1–35, 2019
[9] Taibi D, Lenarduzzi V., Pahl C. and Janes A 2017 Microservices in agile software development Proceedings of the XP2017 Scientific Workshops (New York, NY, USA: ACM) pp 1–5, 2017
[10] Chen L., Babar M. A. 2014 Towards an Evidence-Based Understanding of Emergence of Architecture through Continuous Refactoring in Agile Software Development 2014 IEEE/IFIP Conference on Software Architecture (IEEE), 2014
[11] Balalaie A., Heydarnoori A. and Jamshidi P. 2016 Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture IEEE Software, 2016
[12] Lanza M. and Ducasse S. 2002 Understanding software evolution using a combination of software visualization and software metrics L'objet 8, 2002
[13] Löwy J. Righting software : a method for system and project design, 2020
[14] Wolff E. Microservices - a practical guide : principles, concepts, and recipes, 2019