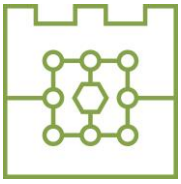
Politechnika Krakowska
im. Tadeusza Kościuszki

Wydział Informatyki i Telekomunikacji



Bartosz Szymański

Numer albumu: 148976

Praktyczny przykład implementacji bezpiecznego przeniesienia aplikacji na architekturę mikroserwisów w chmurze AWS

A practical example of an implementation of a monolithic application secure migration to AWS cloud microservices architecture

Praca magisterska
na kierunku Informatyka
specjalność Cyberbezpieczeństwo

Praca wykonana pod kierunkiem:
dr Barbara Borowik

Kraków, 2024

Spis treści

1. WSTĘP

1.1 Cel pracy

Celem niniejszej pracy dyplomowej jest zbadanie procesu migracji systemu monolitycznego do architektury mikroserwisowej opartej o usługi chmury obliczeniowej dostawcy Amazon Web Services (w skrócie AWS) pod kątem ulepszenia zabezpieczeń omawianego systemu. System objęty badaniem, to: „System do internetowego wspomagania pacjenta i lekarza", który został w całości zaprojektowany i zaimplementowany w formie monolitycznej przez autora pracy dyplomowej jako projekt inżynierski w celu ukończenia studiów inżynierskich pierwszego stopnia. Wybór systemu do przeprowadzenia analizy motywowany jest znajomością jego architektury, wpasowującą się doskonale w temat pracy oraz motywacja do jego dalszego rozwijania i ulepszania w zakresie cyberbezpieczeństwa.

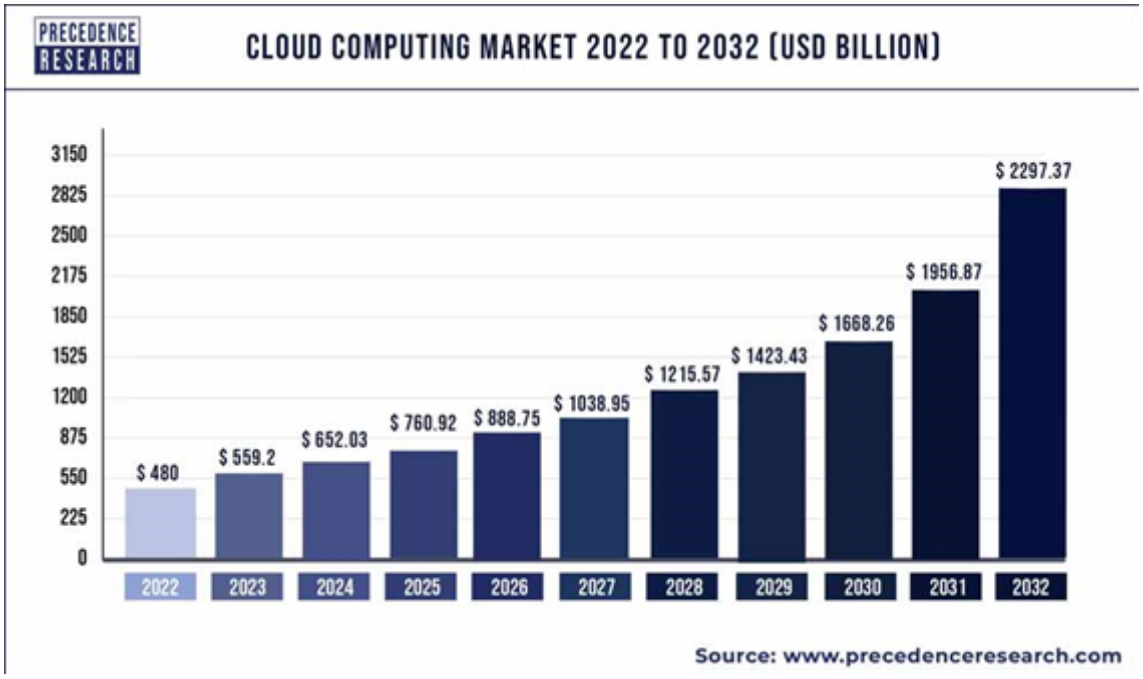
Analiza procesu migracji oraz rozwinięcia systemu zabezpieczeń systemu ma wykazać jak dobrze usługi chmurowe ochraniają swoich usługobiorców i ich oprogramowania przed popularnymi atakami hakerskimi oraz jak duży wpływ na bezpieczeństwo systemu ma jego architektura. Narzędzia używane przez autor pracy to przede wszystkim oprogramowanie wirtualizujące systemy operacyjne Docker, język programowania PHP, język programowania Java Script, zestaw narzędzi programistycznych Symfony oraz Vue.js, dodatkowo usługi AWS, takie jak: Elastic Compute Cloud (w skrócie EC2), Relational Databases (w skrócie RDS), Route 53, Code Pipeline, Code Build, Code Deploy, Elastic Cache, Simple Storage Service, Elastic Container Registry, Lambda. Również narzędzia do analizy ruchu sieciowego, takie jak: Wireshark.

Praca ta omówia aspekty takie jak: ewaluacja zastanej architektury aplikacji, planowanie architektury mikroserwisowej w chmurze AWS, ocena zagadnień bezpieczeństwa podczas migracji, implementacja i wdrażanie mikroserwisów w chmurze AWS, ocena i analiza wyników. Każdy wymieniony etap pracy jest szczegółowo omówiony, odzwierciedlając wiedzę i doświadczenie akademickie jak i zawodowe autora.

1.2 Motywacja

Motywacją do podjęcia tematu pracy są doświadczenia autora na tle zawodowym, które przyspieszyły naukę w zakresie obliczeń chmurowych. Jednak największym moty-

watorem do wykonania badania jest kontynuacja pracy akademickiej, dążenie do ciągłego rozwoju wyprodukowanego przez siebie oprogramowania i wdrażanie wiedzy pozyskanej w czasie studiów drugiego stopnia, tak by umiejętnie zabezpieczać oprogramowanie. Dodatkowym motywatorem jest również fakt, iż technologia chmurowa każdego roku zdobywa coraz większy udział na rynku usług hostingowych, a także staje się dzięki wzrastającej konkurencyjności przystępniejsza dla mniejszych firm lub prywatnych odbiorców. Serwis Precedence Research przewiduje, iż w nadchodzących ośmiu latach udział rynkowy technologii chmurowych zwiększy się około pięciokrotnie do wartości 2297 miliardów dolarów [1]. Patrz Rysunek 1.1.



Rys. 1.1. Wykres przedstawiający prognozę wzrostu udziału rynkowego technologii chmurowej

1.3 Założenia i spodziewane wyniki

Rezultatem niniejszej pracy dyplomowej jest rozebranie monolitycznego systemu na mikroserwisy z pojedynczą odpowiedzialnością logiczną. Zasada działa aplikacji nie ulega zmianie, dla użytkownika zewnętrznego wprowadzenie zmiany powinno odbyć się bez odczuwalnej różnicy. Samo rozczłonkowanie systemu odbywa jak najbardziej atomicznie się da przy rozsądnym nakładzie pracy pozwalającym ukończyć tę pracę w terminie jej obrony. Autor zakłada, iż poprawie ulegnie bezpieczeństwo danych przechowy-

wanych w bazie danych poprzez ukrycie bazy danych przed dostępem z zewnątrz przy wykorzystaniu wirtualnej sieci prywatnej, także większa granularność aplikacji poprawi bezpieczeństwo wdrożenia zmian poprzez ustystematyzowanie procesu jaki za tym stoi poprzez wykorzystanie technologii „Pipeline” (ciąg zautomatyzowanych procesów dostarczania nowego oprogramowania). Również autor odświeżył wersję zależności z jakich składa się projekt, tak by rozwiązać problemy luk bezpieczeństwa wynikających z użycia przestarzałych bibliotek. Dodatkowym poziomem ulepszenia zabezpieczeń aplikacji będzie przeszukanie kodu w pod kątem znalezienia luk logicznych lub twardo zakodowanych dostępów, takich jak hasła do bazy danych lub inne, z których aplikacja korzysta.

2. EWALUACJA ZASTANEJ ARCHITEKTURY APLIKACJI

Zgodnie z stanem aplikacji z dnia 11.06.2022r. projekt “System do internetowego wspomagania pacjenta i lekarza” oparty jest o kilka składowych jakie należy wyróżnić, a są nimi:

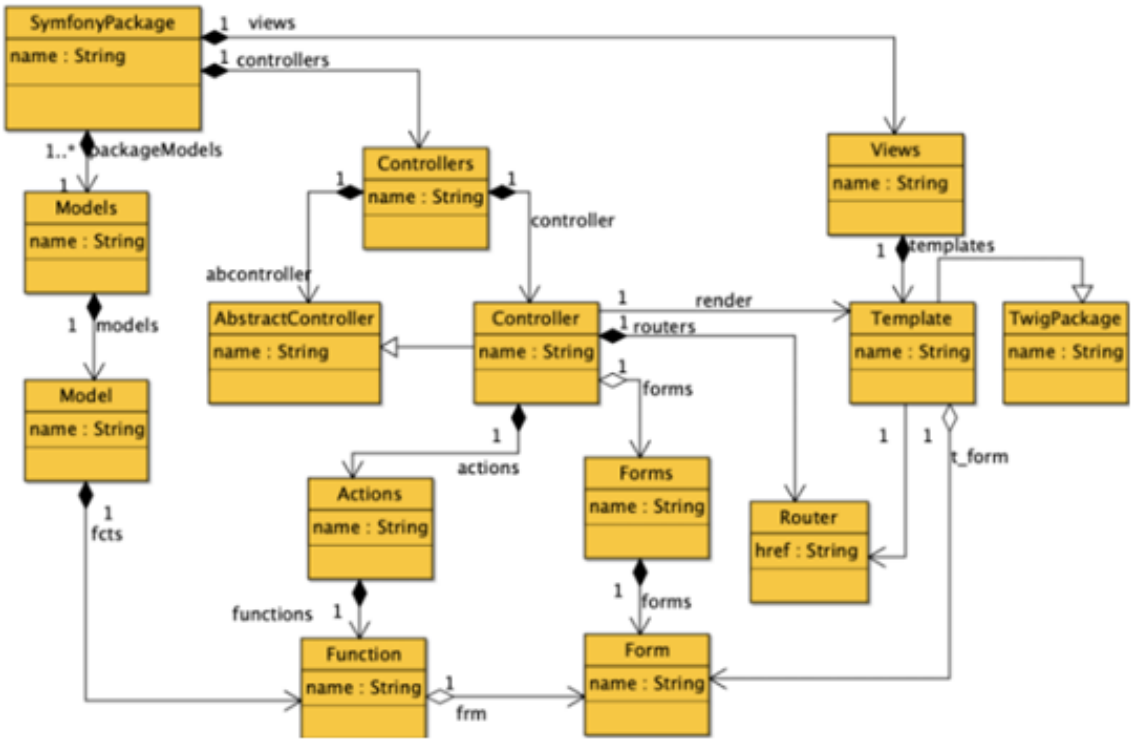
- baza danych oprata o silnik MySQL,
- trzon logiki biznesowej oparty o zestaw narzędzi Symfony w wersji 5.4,
- warstwa widoku aplikacji oparta o silnik Twig oraz Vue.js,
- silnik nginx jako narzędzie do przekazania zapytania klienckiego do logiki biznesowej.

Wyróżnione komponenty są ściśle ze sobą połączone. Logika biznesowa oraz baza danych są nierozerwalne, logika biznesowa posiada klasy encji, które bezpośrednio przekładają się na tabele w bazie danych. Logika biznesowa również decyduje o tym jaki widok jest obecnie wyświetlany, a warstwa widoku bazuje na tym, co dostarczył kontroler Symfony. Cały zestaw wykorzystuje narzędzie nginx do tego, by otrzymać zapytanie od klienta. W niniejszym rodziale zostaną przedstawione kolejno wyżej wymienione komponenty z szerszym opisem co do każdego z nich.

2.1 Logika biznesowa i baza danych

Symfony jako framework, który za zadanie ma ułatwić rozwój oprogramowania internetowego w projekcie z pracy inżynierskiej autora niniejszej pracy spełnił swoją rolę niejako przyczyniając się do skrócenia czasu potrzebnego na rzecz implementacji założeń diagramów użyć, odwzwierciedlenie domen każdego z aktorów systemu, a także diagramów przepływu danych. Narzędzia takie jak wbudowane kontrolery z dostępnym API do zapytań, czy formularze do walidacji danych wejściowych do systemu, router odpowiadający na zapytania, czy szablony wspierane przez silnik Twig to tylko niektóre z narzędzi jakie zostały wykorzystane przy okazji implementacji. Wyczerpujący schemat udogodnień oferowanych przez Symfony znajduje się na Rysunku 2.1.

W celu odzwierciedlenia domen aktorów, w projekcie została wykorzystana biblioteka Doctrine, wspierana przez Symfony. Samo Doctrine to narzędzie do manipulacji bazą danych przy wykorzystaniu obiektów PHP [3]. By zainicjować dane w systemie bez



Rys. 2.1. Schemat metamodelu Symfony [2]

konieczności manualnego wprowadzania ich autor wykorzystał dedykowaną ku temu bibliotekę wykorzystującą mechanizm fabryki danych testowych (ang. Fixtures Factory). Sam mechanizm fabryki danych testowych w procesie tworzenia oprogramowania umożliwia uruchomienie systemu z wypełnioną bazą danych, tak by móc przeprowadzać wymagane akcje i sprawdzić poprawność logiki biznesowej [4].

3. PLANOWANIE ARCHITEKTURY MIKROSERWISOWEJ W CHMURZE AWS

Niniejszy rozdział pokrywa zagadnienie planowania architektury mikroserwisowej oraz wykorzystanych usług potrzebnych do realizacji celu uruchomienia platformy. Nie istnieje mapa działań jakie należy wykonać by zgodnie z sztuką podzielić aplikację monolityczną na zbiór mikroserwisów. Istnieją natomiast wskazówki, którymi można się posługiwać by ułatwić doprowadzenie tego przedsięwzięcia do końca, a są nimi:

- identyfikacja mikroserwisów,
- dbanie o szczególną troskę w procesie ekstrakcji modułów, które są kandydatami na mikroserwisy,
- wprowadzenie modelu heksagonalnego aplikacji. [5]

3.1 Wybór odpowiednich usług do utrzymania aplikacji

W wyborze usług potrzebnych do użycia dla architektury mikroserwisowej, autor korzystał z własnych doświadczeń w pracy z platformą AWS. W związku z powyższym powody wyboru konkretnej usługi zostaną przedstawione w podrozdziale poświęconym na rzecz opisania motywacji doboru.

3.1.1 VPC - Virtual Private Cloud (ang. prywatna, wirtualna chmura)

Powody wyboru VPC dla celu realizacji architektury sieciowej:

- możliwość tworzenia wyizolowanych sieci w chmurze, co zapewnia wysoki standard bezpieczeństwa dla systemu, pozwala kontrolować ruch sieciowy przy pomocy reguł zapory sieciowej,
- możliwość kontrolowania adresacji IP, podsieci, tabel routingu i bram internetowych, co pozwala na dostosowanie sieci do indywidualnych potrzeb budowanego przez autora systemu. [6]

3.1.2 ECS - Elastic Container Service (ang. elastyczny serwis kontenerów)

Wybór ECS jako usługi pozwalającej na uruchomienie mikroserwisów jest uargumentowany niniejszymi powodami:

- ułatwione zarządzanie kontenerami poprzez wbudowane narzędzia do orkiestracji nimi,
- możliwość uruchomienia kontenerów bez konieczności zarządzania architekturą serwerową. [7]

Bibliografia

[1] “Cloud computing market,” 2023, data dostępu: 20 Styczeń 2024.

[2] M. Rahmouni, C. Talbi, and S. Ziti, “Model-driven architecture; generating models from symfony,” 2023, data dostępu: 6 Kwiecień 2024.

[3] F. Potencier, *Symfony 5: The Fast Track*, 2020, data dostępu: 6 Kwiecień 2024.

[4] L. da Silva and P. Vilain, “Execution and code reuse between test classes.” Los Alamitos, CA, USA: IEEE Computer Society, Czerwiec 2016, pp. 99–106, data dostępu: 11 Kwiecień 2024. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SERA.2016.7516134

[5] R. Rocha, P. A. Simoes, and J. C. Purificação, *Java EE 8 Design Patterns and Best Practices*, 2018, data dostępu: 1 Sierpień 2024.

[6] “What is amazon vpc? - amazon virtual private cloud,” https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html, data dostępu: 1 Sierpień 2024.

[7] “What is amazon elastic container service? - amazon elastic container service,” https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html#welcome-terminology, data dostępu: 1 Sierpień 2024.

Spis rysunków