Projektowanie i wdrażanie systemów informatycznych

Karpiński Maciej
Krysa Marcin
Kuczma Łukasz
Mertuszka Adam

31 grudnia 2020

Spis treści

1 Prezentacja firmy

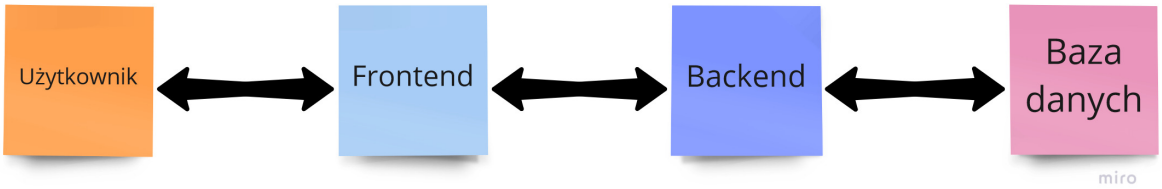
Opis firmy

2 Analiza SWOT

Analiza SWOT - Tabela

3 Analiza potrzeb informacyjnych

Celem projektowanego systemu informatycznego jest usprawnienie realizacji umów z kontrahentami poprzez skuteczne gromadzenie informacji na temat zamówienia, tak aby dział realizujący zamówienie posiadał kompletną listę zamówionych elementów i inny rzeczy wymaganych do montażu u klienta, bez konieczności przeglądania umów przez pracownika magazynu w dziale sprzedaży. Model działania systemu informatycznego przedstawia rysunek nr. 1



Rysunek 1: Model systemu infromatycznego

Dobór oraz zakup oprogramowania i podzespołów komputerowych, na którym będzie funkcjonował system informatyczny.

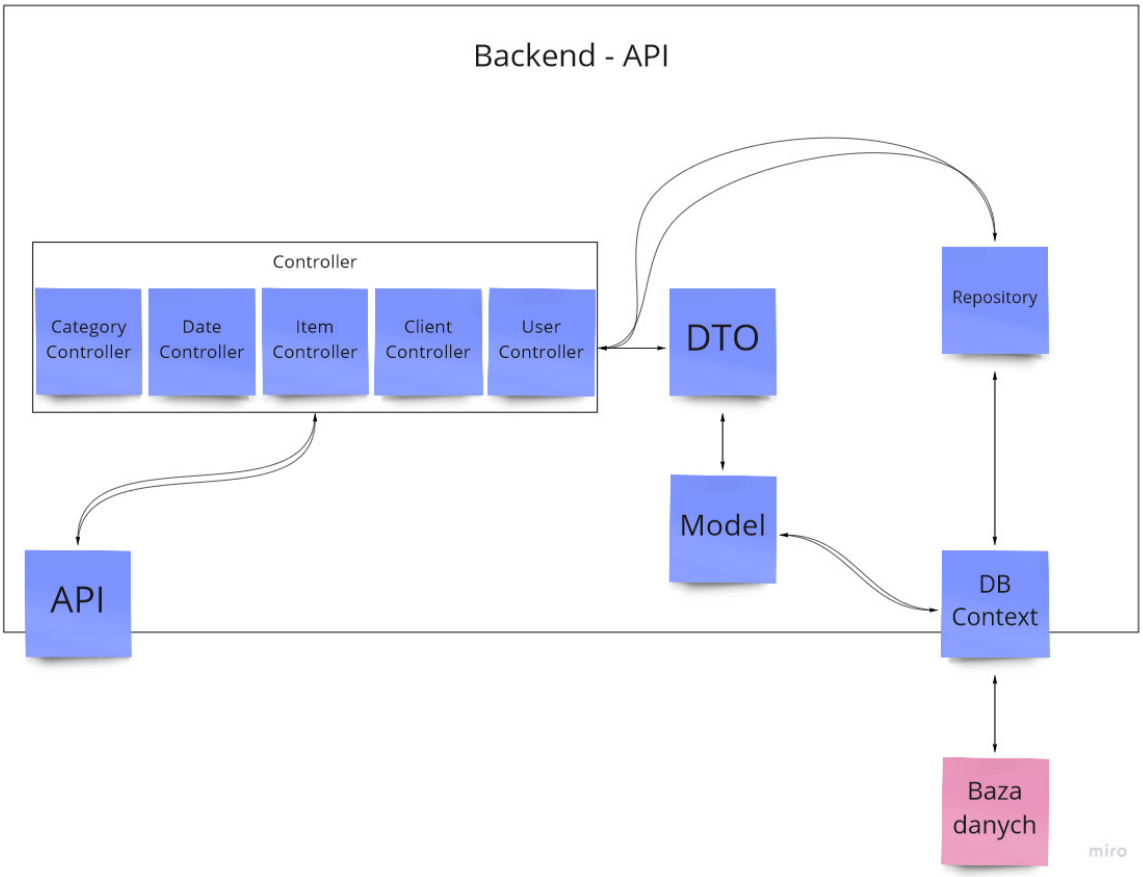
Dostosowanie infrastruktury sieciowej - rozszerzenie i konfiguracja dotychczas istniejącej infrastruktury sieciowej o możliwość podłączenia serwera udostępniającego system informatyczny.

System informatyczny składa się z trzech głównych modułów tj:

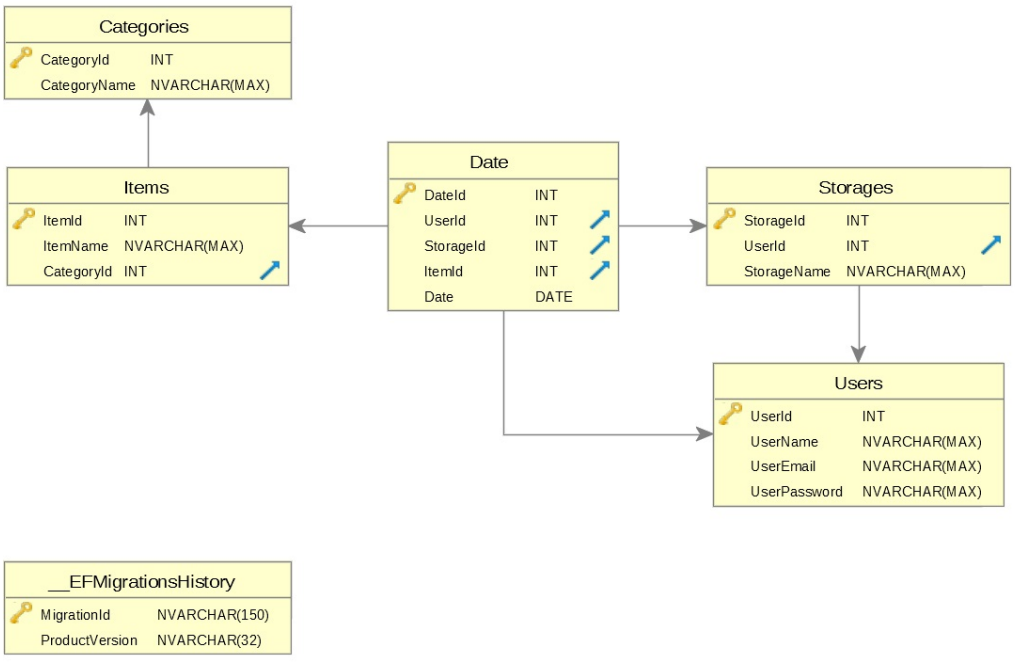
- Backend - API
- Baza danych
- Frontend

Backend - API, realizuje łączność między modułem frontendu a bazą danych, jest odpowiedzialny za przetwarzanie i udostępnianie danych. Backend udostępnia API dzięki, któremu moduł frontendu ma możliwość pobierania i wysyłania niezbędnych danych z bazy danych. Zastosowanie API pozwala w przyszłości rozbudować system o nową funkcjonalność lub alternatywne moduły komunikacyjne z użytkownikiem. Budowa backendu uniemożliwia pobieranie zbędnych danych, dzięki czemu ruch sieciowy oraz wymagana moc obliczeniowa jest ograniczona do niezbędnego minimum. Rysunek nr. 2 przedstawia model budowy Backendu - API.

Baza danych realizuje funkcję gromadzenia danych wprowadzanych przez użytkowników. Dane wprowadzone do bazy danych są raz dziennie automatycznie archiwizowane w kopii zapasowej na twardym dysku zgodnie z ustawionym harmonogramem. Model bazy danych przedstawia rysunek nr. 3



Rysunek 2: Model Backend - API



Rysunek 3: Model bazy danych

Frontend realizuje funkcje komunikacji pomiędzy użytkownikiem a modułem backendu. Udostępnia prosty i przejrzysty interfejs użytkownika, przetwarza dane pobrane z API w sposób przyjazny dla użytkownika, oraz umożliwia w łatwy sposób wprowadzanie nowych danych do systemu informatycznego.

System informacyjny realizuje następującą funkcjonalność:

- Przyjazny interfejs użytkownika - interfejs użytkownika był projektowany i konsultowany wraz z użytkownikami przedsiębiorstwa, którzy będą go codziennie wykorzystywać do pracy;
- Tworzenie i logowanie użytkowników systemu - po otworzeniu okna systemu informatycznego użytkownik będzie miał możliwość zalogowania się na swoje indywidualne konto lub w przypadku nowego użytkownika jego utworzenie;
- Nadawanie uprawnień użytkownikom - administrator systemu może przypisywać uprawnienia dla kont użytkowników, ograniczając im funkcjonalność jedynie do niezbędnych przy wykonywaniu pracy;
- Użytkownik działu sprzedaży może tworzyć i modyfikować „kontener" zawierający listę zamówień, numer umowy oraz termin i adres realizacji zamówienia;
- Użytkownik działu magazynowego może przeglądać listę utworzonych „kontenerów" dzięki czemu będzie mógł uzupełnić magazyn o brakujące elementy;
- Użytkownik działu realizacji zamówienia może przeglądać listę utworzonych „kontenerów" dzięki czemu wie kiedy i gdzie będzie musiał przystąpić do realizacji zamówienia;
- Logowanie akcji użytkowników - system informatyczny loguje wszystkie podejmowane przez użytkowników akcje. Administrator posiada możliwość przeglądania wygenerowanych logów, dzięki czemu ma wgląd w działanie systemu informatycznego;
- Zabezpieczenie przed utratą danych - system operacyjny serwera został skonfigurowany w taki sposób aby była automatycznie tworzona kopia zapasowa danych, kontrola jej integralności oraz aby możliwe było odtworzenie stanu sprzed awarii maksymalnie jednego twardego dysku. Funkcja realizująca archiwizację danych uruchamia się automatycznie codziennie o godzinie 19:00 oraz powinna zakończyć się maksymalnie o godzinie 5:00, pozostałe funkcje zabezpieczające działają ciągle w tle jako część systemu plików i macierzy RAID-Z;
- Zabezpieczenie przed nieautoryzowanym dostępem do danych - dostęp do systemu informatycznego wymaga logowania, wszystkie dane przesyłane pomiędzy systemem informatycznym a użytkownikiem są szyfrowane asynchronicznie. Dostęp do serwera wymaga autoryzacji, konto „root" wymaga podania specjalnego hasła, wszystkie dane na dysku są automatycznie szyfrowane. Dostęp do bazy danych wymaga autoryzacji, system nie jest dostępny poprzez sieć internetową - działa jedynie w lokalnej sieci Ethernet. Zapora sieciowa na serwerze automatycznie blokuje wszelki ruch na portach, które nie są wykorzystywane w realizacji zadań systemu informatycznego, systemu operacyjnego oraz usługi konteneryzacji Docker;

4 Wybór narzędzi

4.1 ASP.NET Core 3.1

ASP.Net Core jest wysokowydajnym frameworkiem, do budowania nowoczesnych aplikacji internetowych wykorzystujących moc obliczeniową chmur. ASP.Net Core jest technologią open - source, wykorzystującą silnik html Razor, dzięki której możliwe jest tworzenie aplikacji mulitplaformowych, które mogą być używane na każdym urządzeniu wyposażonym w przeglądarkę internetową.

4.2 AutoMapper

AutoMapper jest biblioteką służącą do mapowania między obiektami, dzięki czemu można automatycznie mapować właściwości dwóch różnych obiektów, przekształcając obiekt wejściowy jednego typu na obiekt wyjściowy innego typu.

4.3 C#

C# jest obiektowym językiem programowania, zaprojektowanym w latach 1998 – 2001 dla firmy Microsoft. Napisany program jest kompilowany do Common Intermediate Language(CLI), który następnie wykonywany jest w środowisku uruchomieniowym takim jak .NET Framework, .NET Core, Mono lub DotGNU. Wykorzystanie CLI sprawia, że kod programu jest wieleplatformowy (dopóki istnieje odpowiednie środowisko uruchomieniowe). C# posiada wiele wspólnych cech z językami Object Pascal, Delphi, C++ i Java a najważniejszymi cechami C# są:

- Obiektowość z hierarchią o jednym elemencie nadrzędnym (podobnie jak w Javie);
- Zarządzaniem pamięcią zajmuje się środowisko uruchomieniowe;
- Właściwości i indeksery;
- Delegaty i zdarzenia – rozwinięcie wskaźników C++;
- Typy ogólne, generyczne, częściowe, Nullable, domniemane, anonimowe;
- Dynamiczne tworzenie kodu;
- Metody anonimowe;
- Wyrażenia lambda.

4.4 Coverlet

Coverlet to projekt typu open source, który zapewnia wieloplatformowy framework pokrywający kod. Coverlet zbiera dane dotyczące przebiegu testu pokrycia, które są używane do generowania raportów.

4.5 Docker

Docker jest otwarto źródłowym oprogramowaniem służącym do realizacji „konteneryzacji" aplikacji, służąca jako platforma dla programistów i administratorów do tworzenia, wdrażania i uruchamiania aplikacji rozproszonych. Pozwala umieścić program oraz jego zależności (biblioteki, pliki konfiguracyjne, lokalne bazy danych itp.) w lekkim, przenośnym, wirtualnym kontenerze, który można uruchomić na prawie każdym serwerze z systemem Linux. Kontenery wraz z zawartością działają niezależnie od siebie i nie wiedzą o swoim istnieniu. Mogą się jednak ze sobą komunikować w ramach ściśle zdefiniowanych kanałów wymiany informacji. Dzięki uruchamianiu na jednym wspólnym systemie operacyjnym, konteneryzacja jest lżejszym sposobem wirtualizacji niż pełna wirtualizacja lub parawirtualizacja za pomocą wirtualnych systemów operacyjnych.

4.6 Entity Framework

Entity Framework jest technologią open - source do mapowania obiektowo – relacyjnego (ORM), które wspierają rozwój aplikacji zorientowanych na dane. Entity Framework umożliwia programistom pracę z danymi w postaci obiektów i właściwości specyficznych dla domeny, bez konieczności przejmowania się bazowymi tabelami i kolumnami baz danych, w których dane są przechowywane.

4.7 Fluent Assertions

Fluent Assertions to zestaw metod rozszerzających .NET, które pozwalają w bardziej naturalny sposób określić oczekiwany wynik testu jednostkowego. Umożliwia to prostą, intuicyjną budowę testu oraz szybsze diagnozowanie przyczyn niepowodzenia testu dzięki czytelniejszym błędom.

4.8 MailKit

MailKit jest multiplatformową otwarto źródłową biblioteką .NET klienta pocztowego opartą o MimeKit, która została zoptymalizowana pod kątem urządzeń mobilnych. MailKit oferuje następującą funkcjonalność:

- Obsługa proxy HTTP, Socks4, Socks4a i Socks5;
- Uwierzytelnianie SASL;
- Kompletny klient SMTP;
- Kompletny klient POP3;
- Kompletny klient IMAP;
- Sortowanie i wątkowanie wiadomości po stronie klienta;
- Asynchroniczne wersje wszystkich metod sieciowych;
- Obsługa S/MIME, OpenPGP, DKIM i ARC;
- Obsługa Microsoft TNEF.

4.9 Microsoft SQL Server

Microsoft SQL Server jest systemem zarządzania relacyjnymi bazami danych opracowany przez firmę Microsoft. Cechą charakterystyczną jest głównie wykorzystywanie języka zapytań Transact-SQL, który jest rozwinięciem standardu ANSI/ISO. W projekcie wykorzystano wersje 2019 Express, która jest bezpłatną edycją programu Microsoft SQL Server, oferująca podstawowy silnik bazy danych, nieposiadający ograniczenia ilości obsługiwanych baz lub użytkowników. Ograniczenia, występujące w wersji Express to m.in.: korzystanie z jednego procesora, 1 GB pamięci RAM, 10GB plików bazy danych czy brak SQL Agent.

4.10 Quartz.NET

Quartz.NET jest otwarto źródłową biblioteką do planowania zadań. Quartz.NET może być używany do tworzenia prostych lub złożonych harmonogramów wykonywania dziesiątek, setek, a nawet dziesiątek tysięcy zadań. Quartz.NET jest portem biblioteki Quartz dla środowiska Java.

4.11 RAID-Z

Zamiast sprzętowej macierzy RAID, ZFS wykorzystuje „miękki” RAID, oferując RAID-Z(oparty na parzystości, taki jak RAID 5 i podobne) oraz dublowanie dysku (podobne do RAID 1). RAID-Z to schemat dystrybucji danych/parzystości, taki jak RAID 5, ale wykorzystuje dynamiczną szerokość paska: każdy blok jest własnym paskiem RAID, niezależnie od rozmiaru bloku, co powoduje, że każdy zapis RAID-Z jest zapisem pełnym paskiem. To w połączeniu z transakcyjną semantyką kopiowania przy zapisie ZFS, eliminuje błąd dziury w zapisie. RAID-Z jest również szybszy niż tradycyjny RAID 5, ponieważ nie wymaga wykonywania zwykłej sekwencji odczytu-modyfikacji-zapisu. Ponieważ wszystkie paski mają różne rozmiary, rekonstrukcja RAID-Z musi przejść przez metadane systemu plików, aby określić rzeczywistą geometrię RAID-Z. Przeglądanie metadanych oznacza, że ZFS może na bieżąco weryfikować każdy blok względem jego 256-bitowej sumy kontrolnej, podczas gdy tradycyjne produkty RAID zwykle nie mogą tego zrobić. Oprócz obsługi awarii całego dysku, RAID-Z może również wykrywać i korygować ciche uszkodzenie danych, oferując „samo-naprawiające się dane”: podczas odczytu bloku RAID-Z ZFS porównuje go z jego sumą kontrolną, a jeśli dyski nie zwraca prawidłowej odpowiedzi, ZFS odczytuje parzystość, a następnie sprawdza, który dysk zwrócił złe dane. Następnie naprawia uszkodzone dane i zwraca dobre dane. RAID-Z i dublowanie nie wymagają żadnego specjalnego sprzętu: nie potrzebują NVRAM dla niezawodności i nie potrzebują buforowania zapisu dla dobrej wydajności lub ochrony danych. Dzięki RAID-Z ZFS zapewnia szybkie i niezawodne przechowywanie danych przy użyciu tanich, standardowych dysków.

4.12 Swashbuckle

Swashbuckle jest biblioteką, która dodaje zestaw narzędzi „Swagger"generujących automatycznie dokumentację API aplikacji, wyposażoną w przejrzysty interfejs użytkownika. Swashbuckle umożliwia również testowanie API. Dokumentacja jest dostępna pod adresem: „/swagger"

4.13 Ubuntu 20.04

Ubuntu 20.04 LTS (Focal Fossa) server jest kompletną dystrybucją systemu operacyjnego GNU/Linux, przeznaczoną dla serwerów. Ubuntu bazuje na niestabilnej gałęzi „Sid" dystrybucji Debian. Projekt rozwijany jest przed przedsiębiorstwo Canonical Ltd. oraz fundację Ubuntu Foundation. Najważniejszymi cechami dystrybucji jest:

- Domyślne ustawienia i konfiguracja sprzętu;
- Uproszczona administracja;
- Bogaty wybór oprogramowania;
- Wsparcie techniczne;

Producent zapewnia wsparcie techniczne i rozwój dystrybucji do kwietnia 2025.

4.14 xUnit.net

xUnit.net to darmowe narzędzie typu open source służące do testowania jednostkowego przeznaczone dla platformy .NET Framework, napisane przez oryginalnego autora NUnit. xUnit.net współpracuje z platformami Xamarin, ReSharper, CodeRush i TestDriven.NET.

4.15 ZFS

ZFS łączy system plików z menedżerem woluminów. Został opublikowany w 2001 roku jako część systemu operacyjnego Sun Microsystems Solaris, na licencji typu open source. W latach 2005 - 2010 otwarta wersja ZFS została przeniesiona na Linux, Mac OS X (kontynuowany jako MacZFS) i FreeBSD. Zarządzanie przechowywanymi danymi zazwyczaj obejmuje dwa aspekty: zarządzanie woluminami fizycznymi jednego lub większej liczby blokowych urządzeń magazynujących, takich jak dyski twarde i karty SD, oraz ich organizację w logiczne urządzenia blokowe widziane przez system operacyjny(często z udziałem menedżera woluminów, kontrolera RAID, menedżera macierzy lub odpowiedni sterownik urządzenia) oraz zarządzanie danymi i plikami, które są przechowywane na tych logicznych urządzeniach blokowych np. systemie plików). ZFS w przeciwieństwie do większości innych systemów pamięci masowej ujednolica obie te role i działa zarówno jako menedżer woluminów, jak i system plików. W związku z tym ma pełną wiedzę zarówno o dyskach fizycznych i woluminach, jak i o wszystkich przechowywanych na nich plikach. ZFS został zaprojektowany w celu zapewnienia, że dane przechowywane na dyskach nie zostaną utracone z powodu błędów fizycznych lub niewłaściwego przetwarzania przez sprzęt, system operacyjny lub zdarzeń rotacji bitów oraz uszkodzenia danych, które mogą się zdarzyć w czasie, a także posiada pełną kontrolę nad system pamięci masowej jest używany w celu zapewnienia, że każdy krok, niezależnie od tego, czy jest związany z zarządzaniem plikami czy zarządzaniem dyskami, jest weryfikowany, potwierdzany, korygowany w razie potrzeby i optymalizowany w sposób, którego nie są w stanie osiągnąć karty kontrolera pamięci oraz osobne menedżery woluminów i plików. ZFS zawiera także mechanizm tworzenia migawek na poziomie zbioru danych i puli oraz replikacji, w tym klonowanie migawek, które jest opisane w dokumentacji FreeBSD jako jedna z jego „najpotężniejszych funkcji", posiadając funkcje, których „brakuje nawet innym systemom plików z funkcją migawki". Można wykonać bardzo dużą liczbę migawek bez obniżania wydajności, co pozwala na użycie migawek przed ryzykownymi operacjami systemowymi i zmianami w oprogramowaniu lub

wykonanie pełnej migawki całego produkcyjnego systemu plików(„na żywo”) kilka razy na godzinę w celu ograniczenia utraty danych z powodu błędu użytkownika lub złośliwej aktywności. Migawki można przywrócić „na żywo” lub wyświetlać poprzednie stany systemu plików, nawet w bardzo dużych systemach plików, co prowadzi do oszczędności w porównaniu z formalnymi procesami tworzenia kopii zapasowych i przywracania. Migawki można również klonować w celu utworzenia nowych niezależnych systemów plików. Dostępna jest migawka na poziomie puli (nazywana „punktem kontrolnym”), która umożliwia wycofanie operacji, które mogą wpłynąć na strukturę całej puli lub które dodają lub usuwają całe zestawy danych.

5 Tworzenie projektu z uwzględnieniem bezpieczeństwa

Aplikacja została stworzona z uwzględnieniem wszystkich obowiązujących standardów bezpieczeństwa oraz obowiązujących przepisów RODO.

Przy projektowaniu zabezpieczeń aplikacji pomocna była książka *Bezpieczeństwo aplikacji webowych*[1] opisująca liczne mechanizmy i metody zabezpieczenia aplikacji takich jak asynchroniczne szyfrowanie, autoryzacja, uwierzytelnianie, logowanie akcji użytkownika itd. oraz liczne wpisy opisujące wpadki i ataki na serwisy www i aplikacje webowe opisane na blogach internetowych *Niebezpiecznik*[2], *Sekurak*[3] i *Zaufana Trzecia Strona*[4] dzięki, którym udało się wyeliminować dużą ilość potencjalnych wektorów ataków.

Przy zabezpieczaniu serwera oraz bazy danych pomocna była książka *Unix i Linux Przewodnik administratora systemów*[5] oraz artykuły na blogu internetowym *Chris Titus Tech*[6] przy pomocy, których udało się zabezpieczyć i zaszyfrować dane znajdujące się na serwerze przed nieautoryzowanym dostępem. Na serwerze jest automatycznie tworzona kopia bezpieczeństwa, a dzięki wykorzystaniu macierzy RAID-Z, serwer jest odporny na całkowite uszkodzenie jednego z trzech podłączonych twardych dysków. W przeciwieństwie do tradycyjnych macierzy RAID, RAID-Z jest macierzą opartą o system plików ZFS dzięki czemu rozszerzenie macierzy o dodatkowe dyski nie stanowi żadnego problemu a przeniesie dysków do innego serwera nie jest uzależnione od obecności identycznego fizycznego kontrolera macierzy RAID, wadą macierzy RAID-Z jest minimalna obecność 3 nośników danych, w przeciwieństwie do zastosowania macierzy RAID 1, gdzie minimalna ilość nośników danych wynosi 2.

Ogólne rozporządzenie o ochronie danych[7], *poradniki i wskazówki - UODO*[8] wraz z rozmowami z pracownikami wskazały zakres danych, które są niezbędne do przetworzenia zlecenia oraz, które w przypadku złamania i/lub ominięcia zabezpieczeń będą jak najmniej narażać klientów firmy na wszelkiego rodzaju potencjalne straty.

6 Oszacowanie kosztów

Łączny koszt wdrożenia aplikacji w firmie wyniósł: 13 068zł, na którą składały się poszczególne pozycje:

- Projektowanie i programowanie aplikacji: 10 055zł
- Wdrożenie na produkcję: 1 300zł
- Serwer: 1 713zł

Szczegóły dotyczące kosztów przedstawiają tabele 1, 2 i 3

Nazwa:	Ilość godzin:	Cena za godzinę:	Cena:
Analiza zapotrzebowania	3	50	150zł
Projekt aplikacji	5	50zł	250zł
Programowanie frontendu	80	30zł	2 400zł
Programowanie backendu	100	60zł	6 000zł
Programowanie testów	10	30zł	300zł
DevOps	5	35zł	175zł
Poprawki 1	10	40zł	400zł
Poprawki 2	5	40zł	200zł
Dokumentacja	6	30zł	180zł
Razem:			10 055zł

Tablica 1: Koszt projektu i programowania aplikacji

Dobór podzespołów	50zł
Montaż serwera	150zł
Instalacja i konfiguracja systemu operacyjnego	300zł
Konfiguracja sieci komputerowej	100zł
Instalacja i konfiguracji aplikacji	100zł
Szkolenie pracowników	500zł
Inne	100zł
Razem	1 300zł

Tablica 2: Koszt wdrożenia na produkcję

Nazwa:		Sztuk:	Cena:
Procesor	AMD Athlon 3000G	1	229zł
Płyta główna	ASRock B450M-HDV R4	1	259zł
Pamięć RAM	Patriot Viper 4 Blackout	1	179zł
Twardy dysk	Crucial BX 500	3	687zł
Zasilacz	SilentiumPC Elementum E2 450W	1	150zł
Obudowa	SilentiumPC Armis AR1	1	109zł
Razem			1 713zł

Tablica 3: Koszt podzespołów serwera

7 Wdrożenie

Po uprzednim przygotowaniu miejsca w instalację sieci przewodowej Ethernet oraz sieci energetycznej, w którym będzie znajdować się serwer przystąpiono do poskładania wszystkich zakupionych podzespołów serwera a następnie zainstalowania wybranej dystrybucji systemu operacyjnego GNU/Linux. Proces złożenia serwera i instalacji systemu operacyjnego, zajęło około dwóch godzin. Następnie przystąpiono do konfiguracji dostępu SSH, stworzenia i skonfigurowania macierzy dyskowej RAID-Z, zainstalowania i skonfigurowania zapory sieciowej, uruchomienia pełnego szyfrowania dysków. Po wstępnym przygotowaniu serwera został on podłączony w docelowej lokalizacji, a kolejne etapy wdrażania zostały przeprowadzone zdalnie poprzez konsolę SSH. Po podłączeniu serwera do sieci Ethernet, został skonfigurowany znajdujący się w przedsiębiorstwie serwer DHCP oraz switch. Kolejnymi krokami w wdrażaniu aplikacji było zainstalowanie i skonfigurowanie usługi konteneryzacji Docker oraz pobranie i instalacja stworzonej aplikacji. Dzięki modułowej budowie aplikacji, po kolei były uruchamiane i konfigurowane poszczególne elementy systemu aplikacji tj. baza danych MS SQL, Backend-API i Frontend. Instalacja, zabezpieczenie bazy danych i konfiguracja aplikacji potrwała około jednej godziny. Po zakończeniu instalacji i konfiguracji aplikacji, został skonfigurowany harmonogram systemu *cron* w celu automatycznego uruchamiania skryptu tworzącego kopię bezpieczeństwa a następnie weryfikującego poprawność utworzonej kopii. Ze względu na wykorzystanie systemu plików ZFS, nie było potrzeby tworzenia skryptu sprawdzających integralność wcześniej wykonanych kopii zapasowych - system plików automatycznie sprawdza integralność wszystkich plików i w razie potrzeby dokonuje stosownej korekty. Po zakończeniu konfiguracji wszystkich usług i aplikacji, nastąpił etap testowania integralności całego systemu tj. sprawdzono poziomy dostępu do systemu operacyjnego, bazy danych i aplikacji, autoryzację, wszystkie funkcjonalności aplikacji, przeprowadzono test jakości kopii zapasowej oraz przeprowadzono symulację uszkodzenia twardego dysku oraz odbudowę macierzy dyskowej, sprawdzono konfigurację infrastruktury sieciowej. W przypadku wykrycia nieprawidłowości od razu były wprowadzane niezbędne poprawki do konfiguracji, a następnie ponownie sprawdzano czy wszystko działa prawidłowo. Po zakończeniu testów rozpoczęliśmy szkolenie pracowników z wykorzystania aplikacji.

8 Źródła

Literatura

[1] M. Bentkowi, G. Coldwind, A. Czyż, R. Janicki, J. Kamiński, A. Michalczyk, M. Niezabitowski, M. Piosek, M. Sajdak, G. Trawiński, B. Widła: *Bezpieczeństwo aplikacji webowych*, SECURITUM Szkolenia sp. z o.o. sp.k., 2019. ISBN: 978-83-954853-0-5

[2] https://niebezpiecznik.pl

[3] https://sekurak.pl

[4] https://zaufanatrzeciastrona.pl

[5] E. Nemeth, G. Snyder, T. R. Hein, B. Whaley, D. Mackin: *Unix i Linux Przewodnik administratora systemów Wydanie V*, Helion S.A., 2018. ISBN: 978-83-283-4176-0

[6] https://christitus.com

[7] https://uodo.gov.pl/pl/404

[8] https://uodo.gov.pl/7

9 Spis tabel i obrazków

Spis tablic

Spis rysunków