Politechnika Warszawska

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH

Praca dyplomowa inżynierska

na kierunku Informatyka i Systemy Informacyjne

System do zdalnej pracy w środowisku graficznym wykorzystujący maszyny wirtualne QEMU z akceleracja sprzętową

Krzysztof Smogór

Numer albumu 298906

Piotr Widomski

Numer albumu 298919

promotor

dr inż. Marek Kozłowski

WARSZAWA 2022

....

podpis promotora

....

podpis autora

Streszczenie

System do zdalnej pracy w środowisku graficznym wykorzystujący maszyny wirtualne QEMU z akceleracja sprzętową

Streszczam.

Lorem ipsum dolor sit amet, consetetur sadipscing elit, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Słowa kluczowe: slowo1, slowo2, ...

Abstract

Environment for remote work with Graphical User Interface using QEMU virtual machines with hardware acceleration

Konieczne jest załączenie wypełnionego oświadczenia o autorstwie pracy. By tego dokonać, skan (w formacie PDF) należy umieścić w folderze *scans* i nazwać go, np. oswiadczenie_o_autorstwie_pracy.pdf (w przypadku innej nazwy lub umieszczenia w innym folderze, konieczne jest adekwatne zmodyfikowanie ścieżki w komendzie je załączającej — patrz fragment kodu OŚWIADCZENIA.

Keywords: keyword1, keyword2, ...

Załącznik nr 1 do zarządzenia nr 109 /2021
Rektora PW z dnia 9 listopada 2021 r.

załącznik nr 5 do zarządzenia nr 42 /2020 Rektora PW

Politechnika Warszawska

.....
miejscowość i data

.....
imię i nazwisko studenta

.....
numer albumu

.....
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.
Jednocześnie oświadczam, że:

— niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,

— niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,

— niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,

— wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,

— znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

.....
czytelny podpis studenta

Załącznik nr 3 do zarządzenia nr 109 /2021
Rektora PW z dnia 9 listopada 2021 r.

załącznik nr 9 do zarządzenia nr 42 /2020 Rektora PW

Politechnika Warszawska

.....
miejscowość i data

.....
imię i nazwisko studenta

.....
numer albumu

.....
Wydział i kierunek studiów

Oświadczenie studenta w przedmiocie udzielenia licencji
Politechnice Warszawskiej

Oświadczam, że jako autor/współautor* pracy dyplomowej pt. .....
..... udzielam/nie udzielam* Politechnice Warszawskiej nieodpłatnej licencji na niewyłączne, nieograniczone w czasie, umieszczenie pracy dyplomowej w elektronicznych bazach danych oraz udostępnianie pracy dyplomowej w zamkniętym systemie bibliotecznym Politechniki Warszawskiej osobom zainteresowanym.

Licencja na udostępnienie pracy dyplomowej nie obejmuje wyrażenia zgody na wykorzystywanie pracy dyplomowej na żadnym innym polu eksploatacji, w szczególności kopiowania pracy dyplomowej w całości lub w części, utrwalania w innej formie czy zwielokrotniania.

.....
czytelny podpis studenta

* niepotrzebne skreślić

Spis treści

1. Wstęp

1.1. Opis problemu

Można aktualnie zaobserwować dużą zmianę w rynku pracy. Z powodu globalnej epidemii wiele firm zdecydowało się na zmianę pracy stacjonarnej na zdalną. Nawet po złagodzeniu obostrzeń, znaczna część miejsc pracy pozostała przy takim trybie, lub przyjęło hybrydową formę pracy. Taka forma pracy prowadzi jednak do pewnych utrudnień. Pracownicy mogą musieć łączyć się za pomocą funkcji zdalnego pulpitu z komputerami znajdującymi się w biurze. Może to wynikać z niewystarczającej wydajności sprzętu pracownika, lub dostępu do specyficznych programów lub zasobów. W takim wypadku komputer, z którym łączy się pracownik, musi być uruchomiony, a w przypadku awarii - zrestartowany. Dodatkowo taki dostęp może być wymagany przez ograniczony czas, co powoduje, że dużą część czasu spędza włączony, ale nieużywany.

Możliwym sposobem na złagodzenie tego problemu jest użycie zmniejszonej liczby komputerów, które mogą być używane przez większą liczbę pracowników jednocześnie, za pośrednictwem maszyn wirtualnych. Tym zmniejszamy liczbę działających maszyn, a zarządzanie może być rozwiązane za za pomocą zdalnego operowania komputerem, na którym działają.

System stworzony w ramach tej pracy adresuje opisany problem. Rozwiązanie opiera się na tym wcześniej opisanym, jednocześnie rozbudowując je w sposób ułatwiający użytkowanie oraz zarządzanie.

1.2. Podobne rozwiązania

1.3. Wizja systemu

Tworzony system ma za zadanie umożliwiać zdalną pracę za pomocą protokołu zdalnego pulpitu. System skierowany jest w stronę firm zatrudniających wielu pracowników, które chcą scentralizować sprzęt używany do pracy zdalnej.

Użytkownikami końcowym są pracownicy, którzy za pomocą okienkowej aplikacji klienckiej

mogą uzyskać sesję do pracy zdalnej. Użytkownik podczas łączy się za pomocą protokołu zdalnego pulpitu z maszyną wirtualną uruchamiającą obraz systemu GNU/Linux. Uruchamianie i zarządzanie maszynami jest zadaniem aplikacji działającej na rzeczywistej maszynie, która udostępnia swoje zasoby maszynom wirtualnym. Aplikacja ta, oraz rzeczywista maszyna uruchamiająca ją, nazywana jest dalej serwerem wirtualizacji. Aplikacje te działają niezależnie od siebie i nie ma teoretycznego ograniczenia na ich liczbę w systemie. Komunikacją z użytkownikami oraz zarządzaniem systemem zajmuje się aplikacja nadzorcza. Ilość jej instancji również jest teoretycznie nieograniczona, co umożliwia balansowanie obciążeniem.

Wyróżniamy dwa typy maszyn wirtualnych: maszyny wykorzystujące jedynie procesor maszyny, na której pracuje, oraz takie, które mają bezpośredni dostęp do karty graficznej maszyny. Do używania systemu użytkownik musi posiadać konto w systemie katalogowym, który umożliwia użytkownikom dostęp do własnego folderu domowego na każdej maszynie. System katalogowy nie jest ujęty w obrębie systemu, ale jego poprawna konfiguracja jest wymagana do użytkowania systemu.

System udostępnia panel administracyjny w postaci strony WWW umożliwiający podgląd obciążenia i stanu systemu przez upoważnione osoby. Komunikacja aplikacji klienckiej z aplikacją nadzorczą oraz panel administratora wykorzystują komunikację za pomocą protokołu HTTP. Możliwe jest użycie szyfrowanego protokołu HTTPS, pod warunkiem użycia poprawnych certyfikatów SSL/TSL.

1.4. Istotne pojęcia

- Aplikacja kliencka - aplikacja okienkowa uruchamiana na komputerze użytkownika, która umożliwi komunikację z systemem oraz uruchomienie zewnętrznego programu implementującego protokół RDP.

- Aplikacja nadzorcza (Nadzorca) - aplikacja, która przetwarza zapytania od aplikacji klienckiej oraz komunikuje się ze wszystkimi serwerami wirtualizacji. Na podstawie tych informacji buduje model zajętości każdego z serwerów wirtualizacji oraz decyduje kiedy, i na którym serwerze, trzeba uruchomić nowe maszyny wirtualne. Decyduje również, do której wirtualnej maszyny ma podłączyć się użytkownik proszący o utworzenie sesji.

- Serwer wirtualizacji - komputer, który udostępnia swoje zasoby (rdzenie procesora, karty graficzne, pamięć RAM oraz przestrzeń dyskową) w postaci uruchamianych na nim maszyn wirtualnych. Komputer ten uruchamia aplikację, która odpowiada na zapytania aplikacji

nadzorczej oraz wykonuje operacje na maszynach wirtualnych (uruchamianie i wyłączanie). Komputer może uruchamiać co najwyżej jedną aplikację, dlatego zarówno komputer, jak i aplikację, nazywamy serwerem wirtualizacji.

- Maszyna wirtualna CPU - maszyna systemowa emulująca, lub para-emulująca, sprzęt i służąca do uruchamiania systemu operacyjnego. Udostępnia użytkownikowi podstawowe zasoby (procesor, pamięć RAM i przestrzeń dyskowa). Uruchamiana jest na serwerze wirtualizacji z liczbą zasobów określoną w konfiguracji. Maszyna wirtualna uruchamia system operacyjny GNU/Linux (ArchLinux).

- Maszyna wirtualna GPU - maszyna analogiczna do maszyny wirtualnej CPU. Wyróżnia się przekazaną na wyłączność, za pośrednictwem mechanizmu GPU Passthrough, kartą graficzną podłączona do serwera wirtualizacji.

- RDP - protokół zdalnego dostępu do pulpitu od firmy Microsoft[1]. Maszyny wirtualne uruchamiają serwer RDP(XRDP[2]), który umożliwia zdalną pracę za pośrednictwem protokołu RDP.

- Sesja - jednorazowy dostępu użytkownika do systemu oraz maszyny wirtualnej. Utworzenie sesji wiąże się z przypisaniem do użytkownika konkretnej maszyny wirtualnej, na której będzie pracować. Sesja kończy się w przypadku, gdy użytkownik poinformuje system o zakończeniu pracy lub gdy minie czas oczekiwania na odzyskanie połączenia jego utracie.

- Vagrant-box[3] - przygotowany wcześniej obraz maszyny wirtualnej, który umożliwia zmianę dostępnych zasoby. Uruchamiają się bardzo powtarzalnie w środowisku programu Vagrant. Obrazy te używane są do tworzenia maszyn wirtualnych.

- Ansible playbook[4] - skrypt konfiguracyjny dla systemu operacyjnego, który umożliwia parametryzację oraz wykonywanie podczas uruchamiania Vagrant-boxa.

- Panel administratora - aplikacja przeglądarkowa, która umożliwia administratorowi systemu podgląd listy serwerów wirtualizacji znajdujących się w systemie oraz zajętości zasobów.

- Konto użytkownika - profil użytkownika w systemie, do którego ma dostęp na każdej maszynie wirtualnej. Używając przygotowanych wcześniej danych logowania może za ich pomocą

[1] Dokumnetacja protokołu RDP od Microsoft
[2] Strona projektu XRDP
[3] Dokumentacja i opis na stronie Vagranta
[4] Dokumentacja i opis na stronie Ansible'a

logować się do maszyn wirtualnych. Przechowywane są w zewnętrznym (poza opisanym systemem) systemie katalogowym.

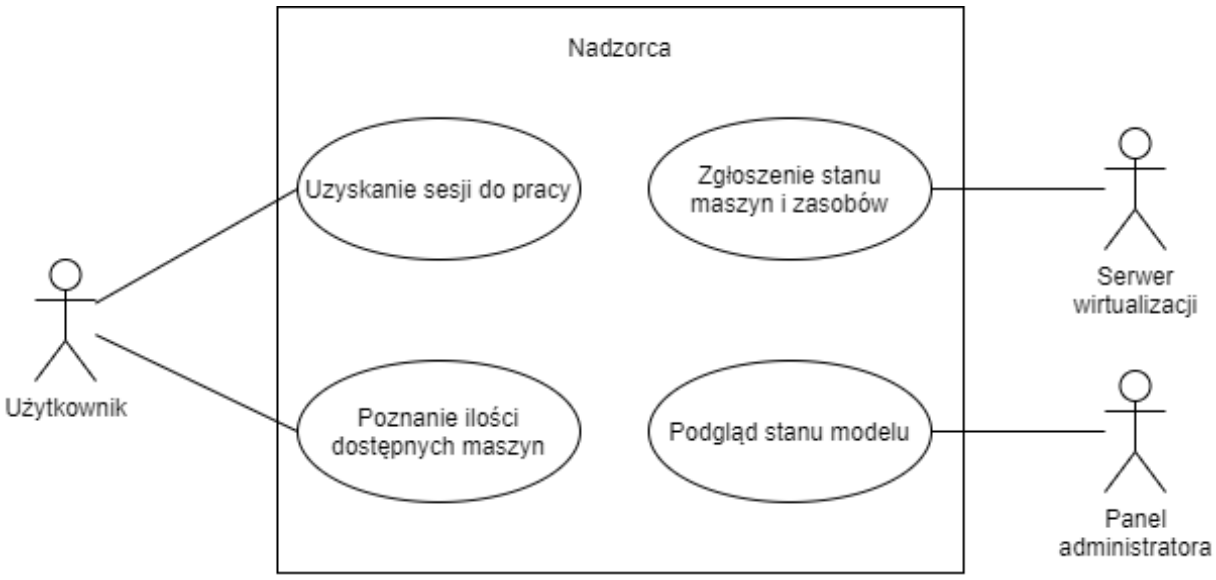
- Katalog użytkownika - prywatny folder dostępny dla użytkownika na każdej maszynie wirtualnej. Przechowywany na zewnętrznym (poza opisanym systemem) dysku sieciowym.

- Konfiguracja stała - konfiguracja maszyny wirtualnej, która nie zmienia się w zależności od miejsca uruchomienia. Docelowo ta konfiguracja ma być zapisana w Vagrant-boxie. W razie potrzeby można ja także zdefiniować w odpowiednim Ansible playbooku.

- Konfiguracja zmienna - konfiguracja maszyny wirtualnej, która zmienia się w zależności od miejsca uruchomienia. Jest definiowana w odpowiednim Ansible playbooku uruchamianym przy każdym włączeniu maszyny.

1.5. Wymaganie funkcjonalne

1.5.1. Nadzorca



Rysunek 1.1: Przypadki użycia aplikacji nadzorczej

Tablica 1.1: Przypadki użycia aplikacji nadzorczej

Aktor	Nazwa	Opis	Odpowiedź systemu
Użytkownik	Uzyskanie sesji do pracy	Uzyskanie sesji do pracy na maszynie wirtualnej CPU lub GPU	Do użytkownika zostaje przydzielona maszyna wirtualna oraz zestawione połączenie RDP. W przypadku, gdy utracił on połączenie, to przydzielana jest do niego poprzednio używana maszyna, jeżeli jego sesja nie została jeszcze umorzona.
	Poznanie ilości dostępnych maszyn	Wyświetlanie szacowanej ilości dostępnych maszyn każdego typu	Użytkownikowi zostaje wyświetlona szacowana liczba dostępnych maszyn obliczona na podstawie informacji o dostępnych zasobach każdego z serwerów wirtualizacji
Serwer wirtualizacji	Zgłoszenie dostępnych zasobów	Serwer zgłasza nadzorcy dostępne zasoby	Nadzorca wykorzystuje zgłoszone zasoby do wyliczania szacowanej liczny dostępnych maszyn oraz do balansowania obciążenia serwerów wirtualizacji
Panel administratora	Podgląd stanu modelu	Nadzorca udostępnia panelowi administratora stan zasobów systemu.	Panel administratora wykorzystuje uzyskane dane to wygenerowania raportu o stanie systemu dla administratora. wirtualizacji

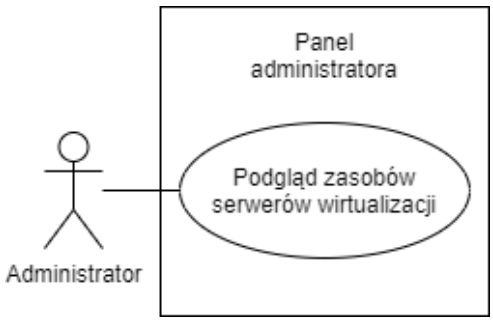
1.5.2. Serwer wirtualizacji

Rysunek 1.2: Przypadki użycia aplikacji nadzorczej

Tablica 1.2: Przypadki użycia serwera wirtualizacji

Aktor	Nazwa	Opis	Odpowiedź systemu
Użytkownik	Nawiązanie połączenia z maszyną	Użytkownik nawiązuje połączenie z maszyną wirtualną	Maszyna wirtualna zostaje zajęta przez użytkownika; serwer wirtualizacji rozpoczyna monitorowanie, czy sesja wciąż trwa
Nadzorca	Poproś o zgłoszenie zasobów	Nadzorca wysyła do wszystkich serwerów wirtualizacji prośbę o zgłoszenie swoich używanych i wolnych zasobów	Serwer wirtualizacji informuje nadzorcę o stanie swoich zasobów
	Stwórz nową wirtualna maszynę	Nadzorca prosi serwer wirtualizacji o stworzenie nowej wirtualnej maszyny dla danego użytkownika na wybranym typie maszyny	Serwer wirtualizacji tworzy wirtualna maszynę i udostępnia możliwość połączenia się z nią
	Wyłącz konkretna wirtualna maszynę	Nadzorca prosi serwer wirtualizacji aby wyłączył konkretna wirtualna maszynę.	Serwer wirtualizacji wyłącza konkretna wirtualna maszynę oraz pilnuje aby na pewno się wyłączyła.
Administrator	Zmień obraz maszyn wirtualnych	Zmiana obrazu źródłowego maszyn wirtualnych	Zdefiniowany przez administratora vagrant-box jest używany przez serwery wirtualizacji
	Zmień konfigurację maszyn wirtualnych	Zmiana zmiennej konfiguracji maszyn wirtualnych	Zmodyfikowany ansible playbook jest używany przez serwery wirtualizacji
	Zdefiniuj zasoby maszyn wirtualnych	Zmiana ilości zasobów przydzielanych na każdy z typów maszyn wirtualnych oraz łączną ilość zasobów przeznaczonych na maszyny	Zmodyfikowana konfiguracja zasobów będzie wykorzystywana przez serwer wirtualizacji przy kolejnym uruchomieniu

1.5.3. Panel administratora



Rysunek 1.3: Przypadki użycia aplikacji nadzorczej

Tablica 1.3: Przypadki użycia panelu administratora

Aktor	Nazwa	Opis	Odpowiedź systemu
Administrator	Podgląd zasobów serwerów wirtualizacji	Wyświetlanie wolnych oraz zajętych zasobów serwerów wirtualizacji	Wyświetlenie zasobów poszczególnych serwerów wirtualizacji, liczby zajętych maszyn oraz szacowanej liczby wolnych maszyn

1.6. Wymaganie niefunkcjonalne

Tablica 1.4: Wymagania niefunkcjonalne

Grupa wymagań	Nr wymagania	Opis
Użytkowanie (Usability)	1	Aplikacja kliencka ma działać na systemach operacyjnych MS Windows (Windows 10) oraz GNU/Linux (ArchLinux). Aplikacja na systemach GNU/Linux wymaga zainstalowanego klienta RDP zgodnego z XRDP[5].
	2	Aplikacja kliencka musi udostępniać możliwość użycia własnego klienta RDP do nawiązania połączenia z maszyną wirtualną
	3	Maszyny wirtualne muszą mieć dostęp do systemu przechowującego konta użytkowników wraz z ich katalogami domowymi
Niezawodność (Reliability)	4	System musi być odporny na awarie poszczególnych serwerów wirtualizacji i kontynuować działanie w sposób niezauważalny dla użytkowników nie używających danego serwera.
	5	Awaria nadzorcy może spowodować uniemożliwienie rozpoczęcia nowych sesji, ale nie może przerwać istniejących sesji
Wydajność (Performance)	6	Łącznie zużywane zasoby przez maszyny wirtualne na poszczególnym serwerze wirtualizacji nie mogą przekroczyć wcześniej zdefiniowanych limitów
	7	Nadzorca musi balansować obciążenie serwerów wirtualizacji
	8	W systemie zawsze musi istnieć jedna działająca maszyna wirtualna nie połączona z żadną sesją, aby można było ją szybko przydzielić użytkownikowi
	9	Zwolnione maszyny wirtualne, które nie są wykorzystywane jako zapas, muszą być wyłączane
Utrzymanie (Supportability)	10	Możliwe jest działanie więcej niż jednego nadzorcy w systemie, w celu zwiększenie dostępności lub przeprowadzenia prac utrzymaniowych

1.7. Analiza ryzyka

Tablica 1.5: Analiza ryzyka

Mocne strony	Słabości
• Łatwa skalowalność pod względem liczby sesji w systemie • Wiele rozwiązań Open Source • Elastyczność pod względem konfiguracji • Tańsze rozwiązanie niż kupno stacji roboczych	• System trudny w konfiguracji • Potrzeba wymiany sprzętu komputerowego • Krótki czas rozwoju systemu • Ograniczenie doświadczenie twórców systemu • Małe prawdopodobieństwo wsparcia projektu po zakończeniu prac
Okazje	**Zagrożenia**
• Grupa docelowa to firmy z dużą ilością stacji roboczych • Zwiększenie zapotrzebowania na prace zdalną na rynku pracy	• Istnienie konkurencji ugruntowanej na rynku • System w dużej mierze oparty o oprogramowanie rozwijane przez inne organizacje

1.7.1. Omówienie zagrożeń

- System trudny w konfiguracji - wysoko prawdopodobne

  Można temu zaradzić poprzez udostępnienie dokładnej dokumentacji lub ścisłą współprace z klientem przy wdrażaniu systemu.
  Wartość: duża

- Potrzeba wymiany sprzętu komputerowego - średnio prawdopodobne

  Klient może potrzebować wymienić aktualne stacje robocze na terminale oraz zainwestować w sprzęt serwerowy. Jednak gdy klientami będą firmy, które mają dużo pracowników pracujących spoza biura, lub dopiero tych pracowników pozyskują, to kupno terminali i

serwerów powinno być bardziej zachęcające niż kupno stacji roboczych.
Wartość: średnia.

- Krótki czas rozwoju systemu - wysoko prawdopodobne

  Czas rozwoju systemu jest bardzo ograniczony. Aby pomimo tego ograniczenia działał on w sposób akceptowalny powinniśmy skupić się na dobrym przedyskutowaniu i opisaniu kluczowych modułów systemu. W czasie projektu należy pilnować aby nie dodawać nadmiarowych funkcjonalności do systemu. W czasie implementacji krytyczne będzie dokładne zaplanowanie aplikacji pod kątem testowania automatycznego. Ułatwi to wyłapywanie prostych błędów jeszcze we wczesnej fazie projektu.
  Wartość: wysoka

- Ograniczone doświadczenie twórców systemu - pewne

  Jedynym sposobem na ograniczenie ryzyka jest rozważna implementacja.
  Wartość: średnia

- Małe prawdopodobieństwo wsparcia projektu po zakończeniu prac - wysoko prawdopodobne

  Trudno teraz przewidzieć co się stanie z projektem po zakończeniu prac. Jednak prawdopodobnie twórcy systemu zajmą się innymi projektami. Można jedynie dokładnie komentować kod i pokrywać jak najwięcej jego części testami. Wtedy inne osoby będą w stanie szukać błędów albo próbować w taki sposób uzupełnić brakującą wiedzę o systemie.
  Wartość: niska

- Istnienie konkurencji ugruntowanej na rynku - bardzo prawdopodobne

  Konkurencyjne systemy oferujące podobne rozwiązania są już dobrze ugruntowane na rynku i przetestowane. Nasz system może spróbować konkurować jedynie z nimi ceną implementacji oraz elastycznością.
  Wartość: średnia

- System w dużej mierze oparty o oprogramowanie rozwijane przez inne organizacje - nisko prawdopodobne

  W czasie życia systemu mogą pojawić się błędy w oprogramowaniu nie rozwijanym w ramach naszego systemu. naprawa takich błędów może trwać bardzo długo. Pewnym sposobem wsparcia takiego systemu jest własnoręczne poprawiania błędów w zewnętrznym oprogramowaniu i zgłaszanie ich do odpowiedniej organizacji. Do czasu zastosowania poprawki jest możliwość korzystania z wersji, na którą nanieśliśmy własną poprawkę.
  Wartość: wysoka

1.8. Podział pracy

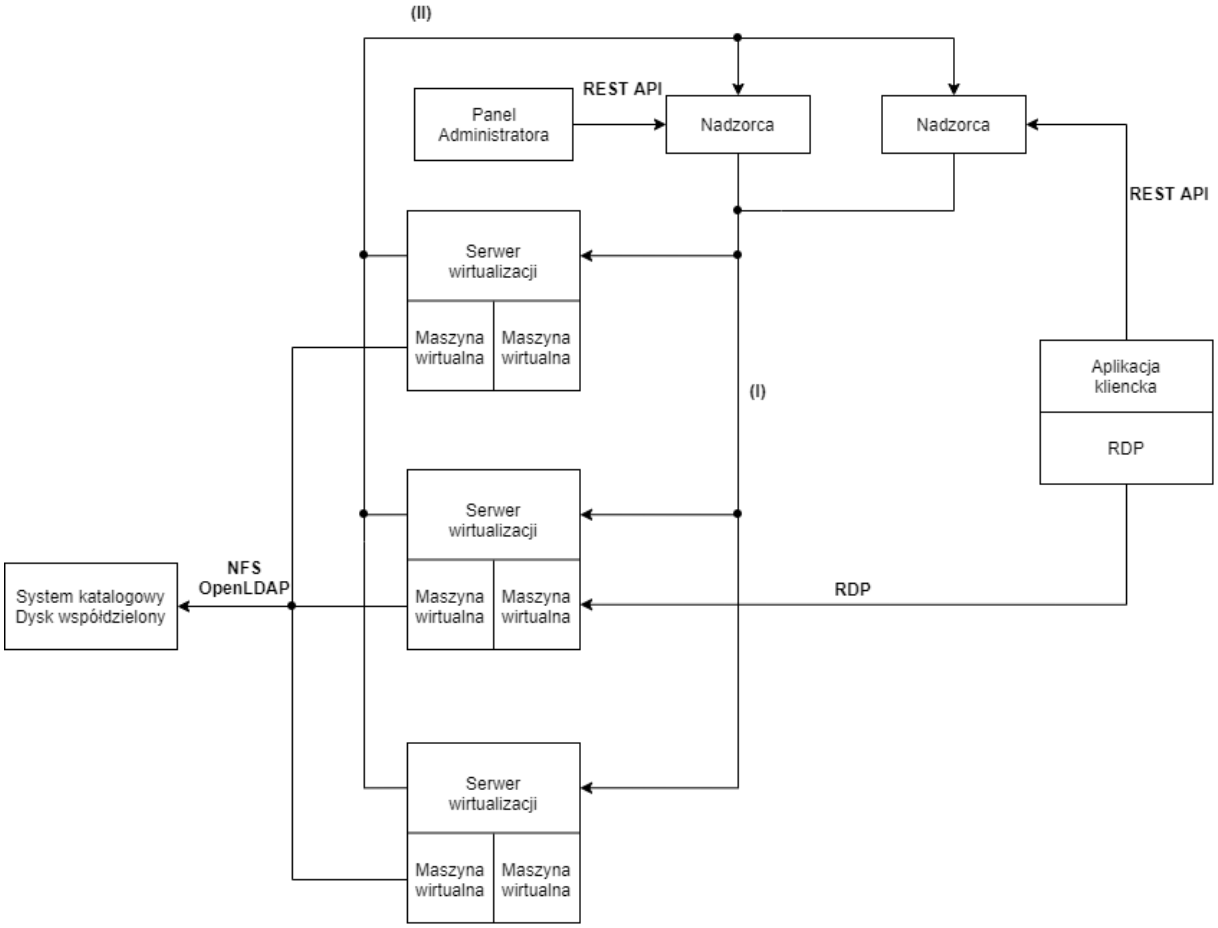
2. Opis rozwiązania

2.1. Architektura systemu

Opracowywany system składa się z następujących modułów:

- nadzorcy,
- serwera wirtualizacji,
- aplikacji klienckiej,
- panelu administratora,
- brokera wiadomości,
- systemu katalogowego,
- dysku współdzielonego.

Schematyczny obraz systemu przedstawia poniższy rysunek.



Rysunek 2.1: Schematyczna architektura systemu

Połączenia oznaczone liczbami rzymskimi oznaczają kolejki komunikacji za pośrednictwem brokera wiadomości, które opisane zostały w punkcie jemu poświęconym. Z założenia system powinien móc skalować się w dwóch wymiarach, to znaczy:

1. Zwiększanie liczby serwerów wirtualnych - zwiększenie liczby istniejących jednocześnie sesji.
2. Zwiększenie liczby nadzorców - zwiększenie liczby obsługiwanych klientów jednocześnie oraz niezawodności systemu.

2.1.1. Nadzorca

Aplikacja mająca za zadanie obsługiwać komunikację z aplikacjami klienckimi oraz wysyłać polecenia do serwerów wirtualizacji. Udostępnia REST API służące do komunikacji z aplikacjami klienckimi. Do komunikacji z serwerami wirtualizacji wykorzystuje kolejki.

Nadzorca przechowuje wewnętrznie model systemu zawierający informację o działających serwerach wirtualizacji i stanie ich maszyn. Na podstawie tego modelu moduł stwierdza, do której maszyny przypisać nowo utworzoną sesję. Wewnętrzne procesy skupione są wokół zmian modelu.

Jeżeli proces wysłał do serwera wirtualizacji prośbę o zmianę stanu, to dalsze procesowanie odbywa się, gdy stan modelu został zaktualizowany, i na podstawie jego stanu podejmowane są decyzje.

Dzięki zastosowaniu kolejek oraz zasad komunikacji w systemie może istnieć więcej niż jeden nadzorca. Instancje nadzorców działają niezależnie od siebie i przechowują identyczny model systemu. Dzięki temu uzyskujemy retencję i możemy zmniejszyć obciążenie poszczególnych nadzorców.

2.1.2. Serwer wirtualizacji

Zadaniem serwera wirtualizacji jest uruchamianie i zarządzanie maszynami wirtualnymi, z którymi łączy się użytkownik systemu. Komunikuje się on z nadzorcami i wykonuje operacje na maszynach wirtualnych zgodnie z żądaniami.

Moduł ten nie jest w stanie funkcjonować samodzielnie. Z tego powodu aplikacja nie uruchomi się, jeżeli nie jest w stanie nawiązać połączenia z aplikacją nadzorczą, a w przypadku ostatni nadzorca w systemie zakończy działanie, aplikacja również je zakończy, pod warunkiem że nie ma żadnych działających sesji.

Serwer wirtualizacji jest częścią systemu, która przechowuje realne zasoby udostępniane użytkownikom. System zaprojektowany jest w taki sposób aby teoretycznie nie było ograniczenia na liczbę serwerów wirtualizacji działających jednocześnie.

2.1.3. Aplikacja kliencka

Aplikacja okienkowa umożliwiająca użytkownikowi autoryzację, uzyskanie sesji oraz automatyczne rozpoczęcie połączenia. Komunikuje się z nadzorcą za pomocą REST API.

Proces uzyskania sesji z perspektywy aplikacji klienckiej zawiera:

1. Uzyskanie informacji o dostępnych typach i liczbie maszyn
2. Wybór typu maszyny
3. Oczekiwanie na utworzenie sesji
4. Nawiązanie połączenia RDP
5. Utrzymanie i monitorowanie stanu połączenia.

2.1.4. Panel administratora

Prosta aplikacja internetowa umożliwiająca administratorowi systemu podgląd stanu zużycia zasobów serwerów wirtualizacji.

2.1.5. Broker wiadomości

Komunikacje wewnątrz systemu, czyli pomiędzy serwerami wirtualizacji oraz nadzorcami, będzie realizowali poprzez kolejki wiadomości. W tym celu użyty został system RabbitMQ, który zajmuje się transportem wiadomości wewnątrz systemu oraz niezawodnością komunikacji między modułami.

Zdefiniowane zostały następujące kolejki wiadomości:

(I) Kolejka kończąca się na każdym z serwerów wirtualizacji powielająca wiadomości między nich. Służy ona do wysyłania nie spersonalizowanych próśb od nadzorców do serwerów wirtualizacji.

(II) Kolejka kończąca się na każdym z nadzorców powielająca wiadomości między nich. Służy ona do przesyłania informacji do nadzorców o zmianach wewnątrz serwera wirtualizacji.

(III) Kolejka kończąca się wyłącznie na pojedynczym serwerze wirtualizacji. Liczba kolejek zgadza się z liczbą serwerów wirtualizacji aktywnych w systemie. Służą one do przesyłania spersonalizowanych wiadomości oraz sprawdzania, czy serwer wirtualizacji nadal pracuje po drugiej stronie. Skorzystamy z funkcjonalności kolejek na wyłączność(Exclusive Queue[1]).

(IV) Kolejka kończąca się na aktualnie podłączonym do maszyny wirtualnej kliencie. Podobnie jak powyżej kolejek istnieje tyle ile aktywnych użytkowników. Celem kolejki jest sprawdzenie, czy aplikacja kliencka nadal jest podłączona do wirtualnej maszyny (mechanizm Exclusive Queue). W celach bezpieczeństwa będą one definiowane na oddzielnym procesie brokera, który będzie można w razie potrzeby udostępnić poza sieć lokalną.

Powyższe 4 grupy kolejek umożliwią prawidłowe działanie systemu. Każdy z modułów tworzy w trakcie uruchamiania kolejki, z których odbiera wiadomości. Jedynym wymogiem prawidłowego uruchomienia komunikacji jest dostępny dla wszystkich serwerów wirtualizacji oraz nadzorców proces brokera.

2.2. Zewnętrzne narzędzia

2.2.1. Ansible

Ansible został wykorzystany w systemie do zaaplikowania zmiennej konfiguracji do każdej uruchamianej wirtualnej maszyny. Podstawowo playbook będzie zawierać informacje o:

[1]Opis zachowania kolejek na wyłączność

1. danych dostępowych do dysku sieciowego oraz wykorzystanym protokole,
2. danych dostępowych do usługi katalogowej.

Playbook można rozszerzać o potrzebne dane zależne od użycia.

2.2.2. Vagrant

Vagrant został wykorzystany w celu łatwej parametryzacji oraz powtarzalnego tworzenia maszyn wirtualnych z przygotowanego wcześniej obrazu systemu.

Wykorzystywany jest głównie mechanizm Vagrant-boxów, które są obrazami wcześniej przygotowanego systemu operacyjnego. Aby system działał prawidłowo obraz systemu zamknięty w Vagrantboxie musi spełniać następujące warunki:

1. Użytkownicy muszą być pobierani z usługi katalogowej.
2. Katalogi domowe użytkowników muszą być na dysku sieciowym.
3. Musi istnieć serwer RDP

2.2.3. Libvirt z QEMU

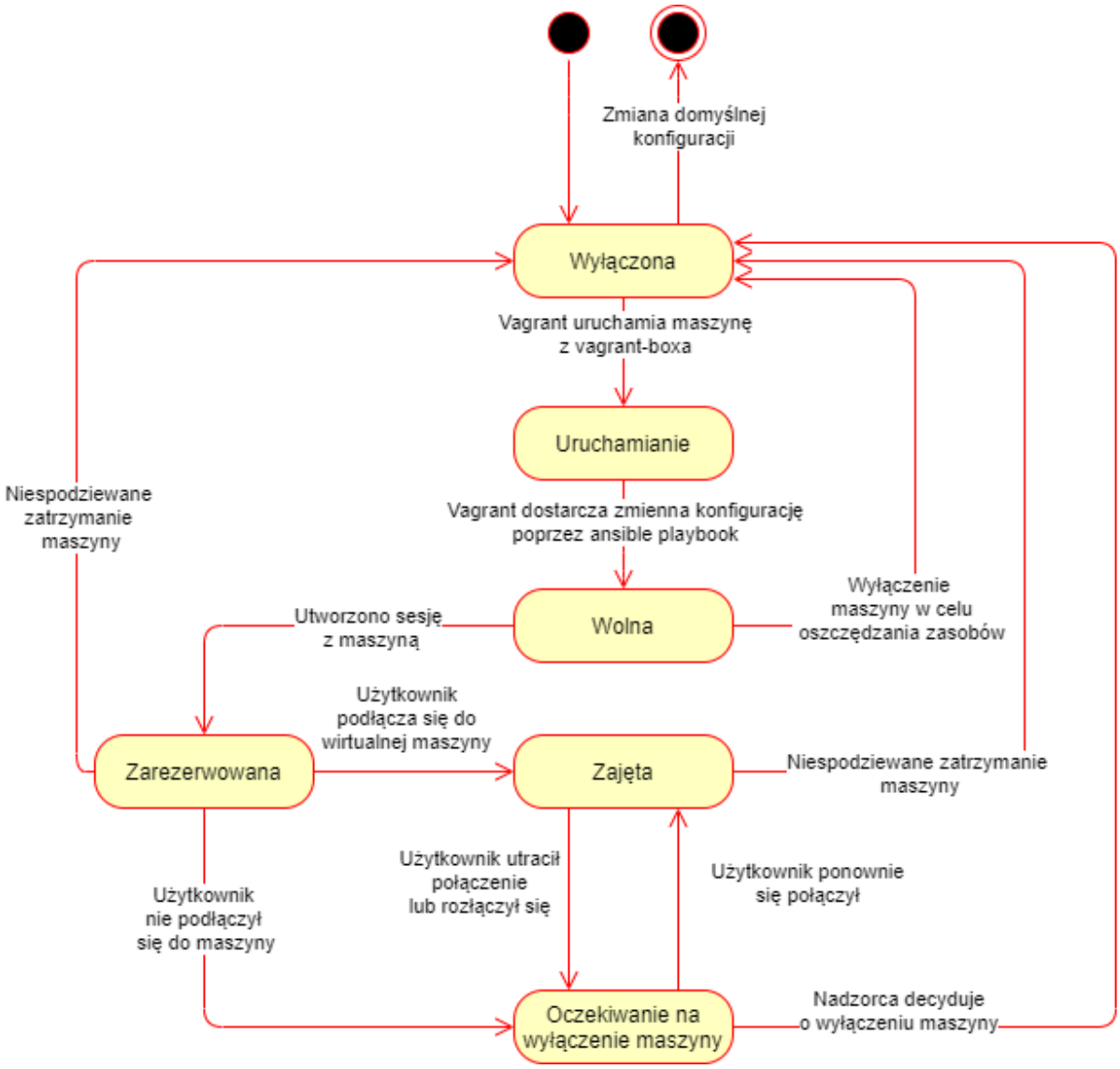
Libvirt połączony z QEMU jest wykorzystany do zarządzania maszynami wirtualnymi uruchamianymi na serwerze wirtualizacji. Umożliwia on:

1. Tworzenie maszyn wirtualnych.
2. Uruchamianie maszyn wirtualnych.
3. Przyporządkowanie zasobów maszynom wirtualnym (w tym krat graficznych).
4. Wyłączanie maszyn wirtualnych.
5. Sprawdzanie, czy maszyna o danej nazwie już działa.

2.3. Stany biznesowe

2.3.1. Maszyna wirtualna

Najważniejszym obiektem biznesowym w systemie jest maszyna wirtualna, do której będą podłączać się użytkownicy.

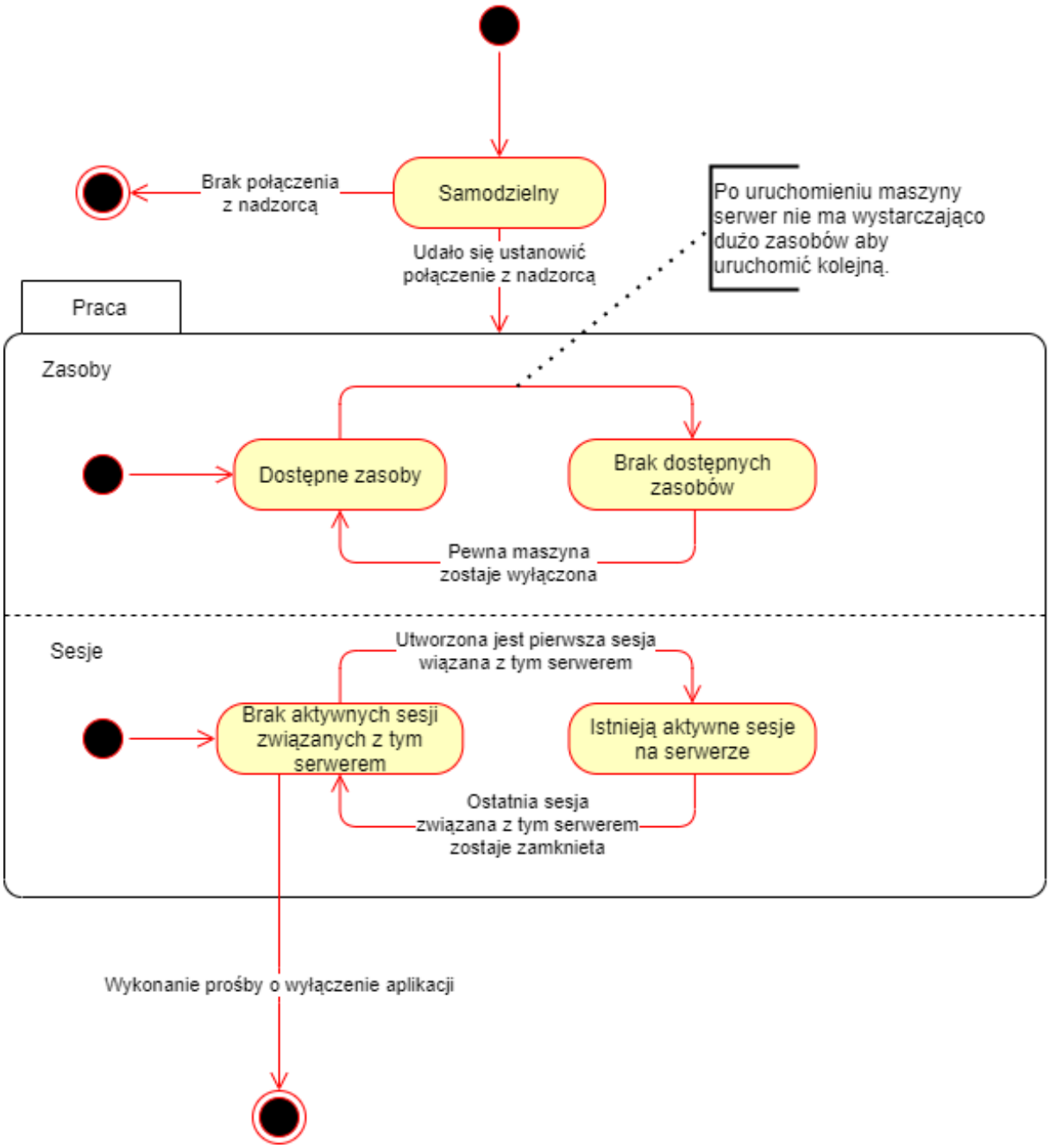


Rysunek 2.2: Diagram stanów dla maszyny wirtualnej

Maszyna, aby być całkowicie uruchomiona, musi zostać zaopatrzona we wszystkie konfiguracje. Stan "Wolnaóznacza możliwość przypisania sesji do tej maszyny. Po przypisaniu maszyny do sesji przechodzi ona w stan oczekiwania na użytkownika. Czas oczekiwania jest konfigurowalny, a po jego upłynięciu maszyna przechodzi w stan oczekiwania na wyłączenie. W tym stanie oczekuje ona na ponowne połączenie, pozwalając użytkownikowi na bezproblemowy powrót do sesji w przypadku nieoczekiwanego utracenia połączenia. Jeżeli użytkownik nie powróci do sesji, system wyłącza maszynę. Maszyna jest w stanie Źajęta", gdy aktualnie pracuje na niej użytkownik. Monitorowanie zajętości realizowane jest przy użyciu odpowiedniej kolejki wiadomości.)

2.3.2. Serwer wirtualizacji

Serwer wirtualizacji monitoruje zasoby zużywane przez uruchamiane na nim maszyny wirtualne oraz fakt podłączenia do niego użytkowników.



Rysunek 2.3: Diagram stanów dla serwera wirtualizacji

Przy starcie serwer wirtualizacji oczekuje na działającego nadzorcy w sieci. W przypadku jego braku serwer kończy działanie zwracając błąd. Pod względem zasobów, może on mieć wolne zasoby aby utworzyć nowe maszyny, lub też nie. Jednak ważniejszym stanem z perspektywy działania serwera są podłączeni do niego użytkownicy. W przypadku gdy podłączony jest do niego

przynajmniej jeden użytkownik, serwer nie może się poprawnie zakończyć pracy aż użytkownik nie skończy używać maszyny.

2.3.3. Użytkownik

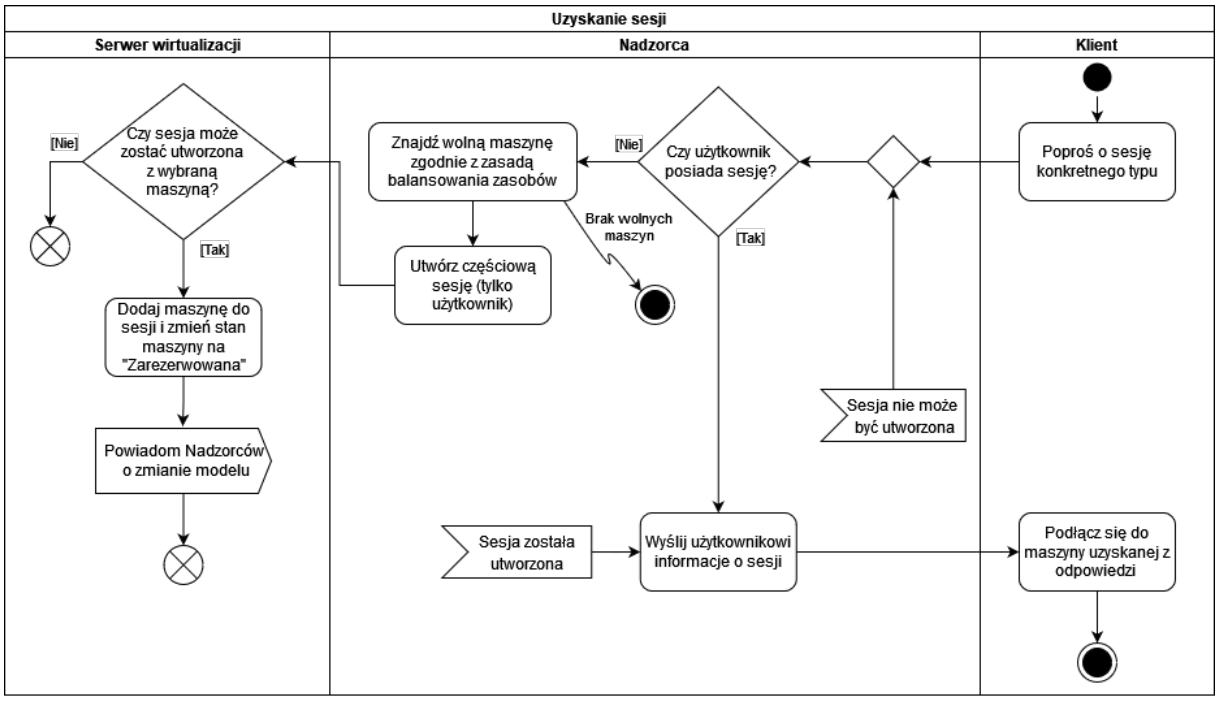
Rysunek 2.4: Diagram stanów dla użytkownika

Użytkownik z perspektywy systemu po zalogowaniu może być 2 dwóch stanach: pracuje w ramach swojej sesji lub też nie. W stanie "Połączonyąplikacja kliencka powiadamia serwer wirtualizacji, że ciągle jest obecny. Przy zmianie stanu do innego informowanie musi ustać, aby serwer mógł wykryć odłączenie się użytkownika.

2.4. Procesy biznesowe

2.4.1. Uzyskanie sesji

Proces opisuje prośbę klienta o ustanowienie dla niego sesji. Sesja może już istnieć lub zostać utworzona.



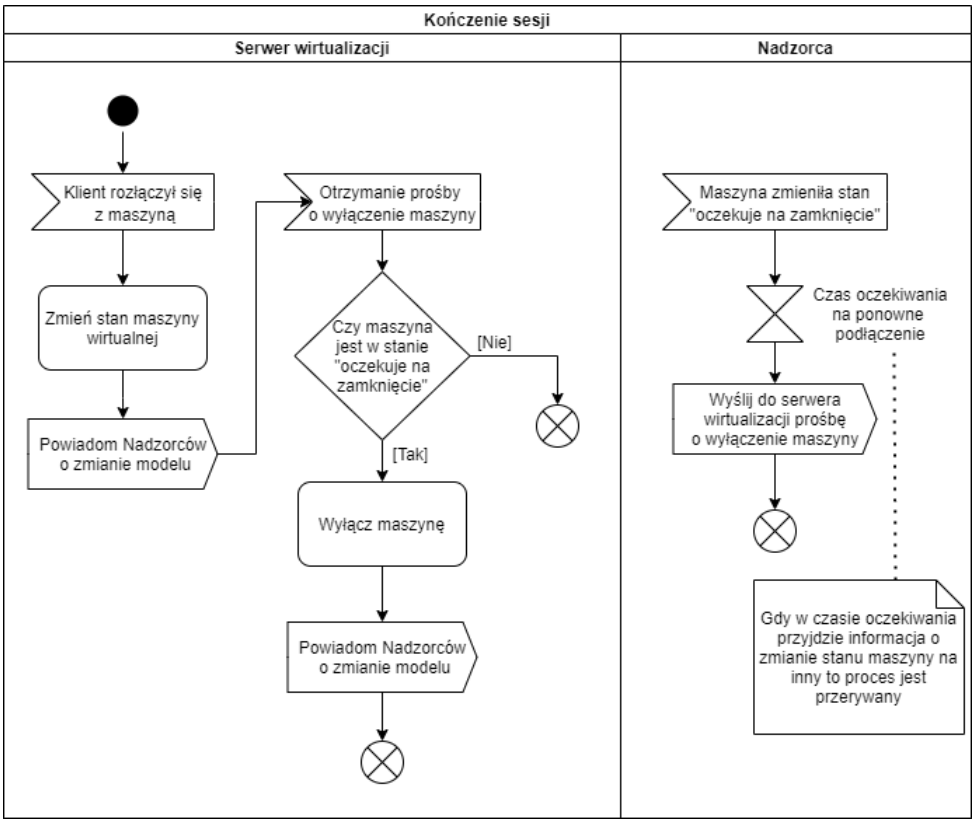
Rysunek 2.5: Diagram aktywności dla uzyskania sesji

W przypadku gdy sesja już istnieje zostaje one zwrócona użytkownikowi. W przeciwnym razie nadzorca, na podstawie modelu systemu, wybiera pewną wolna maszynę i wysyła do serwera wirtualizacji, na którym działa wybrana maszyna, prośbę o utworzenie sesji. Możliwość działanie wielu nadzorców wymaga, aby proces ten był powtarzalny, czyli dla konkretnego stanu modelu wybrana musi zostać ta sama maszyna. Gdy nie znajdzie wolnej maszyny, to zgłasza błąd do użytkownika. Z założenia taka sytuacja może zajść jedynie, gdy wszystkie maszyny zostały zajęte i brakuje zasobów na utworzenie nowych. Wynika to z tego, że system zawsze powinien trzymać pewien zapas wolnych maszyn. Może się zdarzyć, że model jest nieaktualny i nie można utworzyć sesji z wcześniej wybraną maszyną. Taka prośba zostaje odrzucona przez serwer wirtualizacji, ale zmiana modelu w nadzorcy, wywołana odświeżeniem modelu, spowoduje powtórzenie procesu, tym razem wybierając inną maszynę.

Uzyskanie sesji przez użytkownika zrealizowane jest asynchronicznie. Użytkownik oddzielnym zapytaniem prosi o uzyskanie sesji, po czym używając otrzymanego identyfikatora sesji prosi o jej dane. Obiekt jest w pełni utworzona, gdy odpowiedź nadzorcy sesję w stanie gotowym oraz adres maszyny do połączenia.

2.4.2. Kończenie sesji

Proces ma za zadanie zakończyć sesję oraz wyłączyć skojarzoną z nią wirtualna maszynę w celu zwolnienia zasobów.

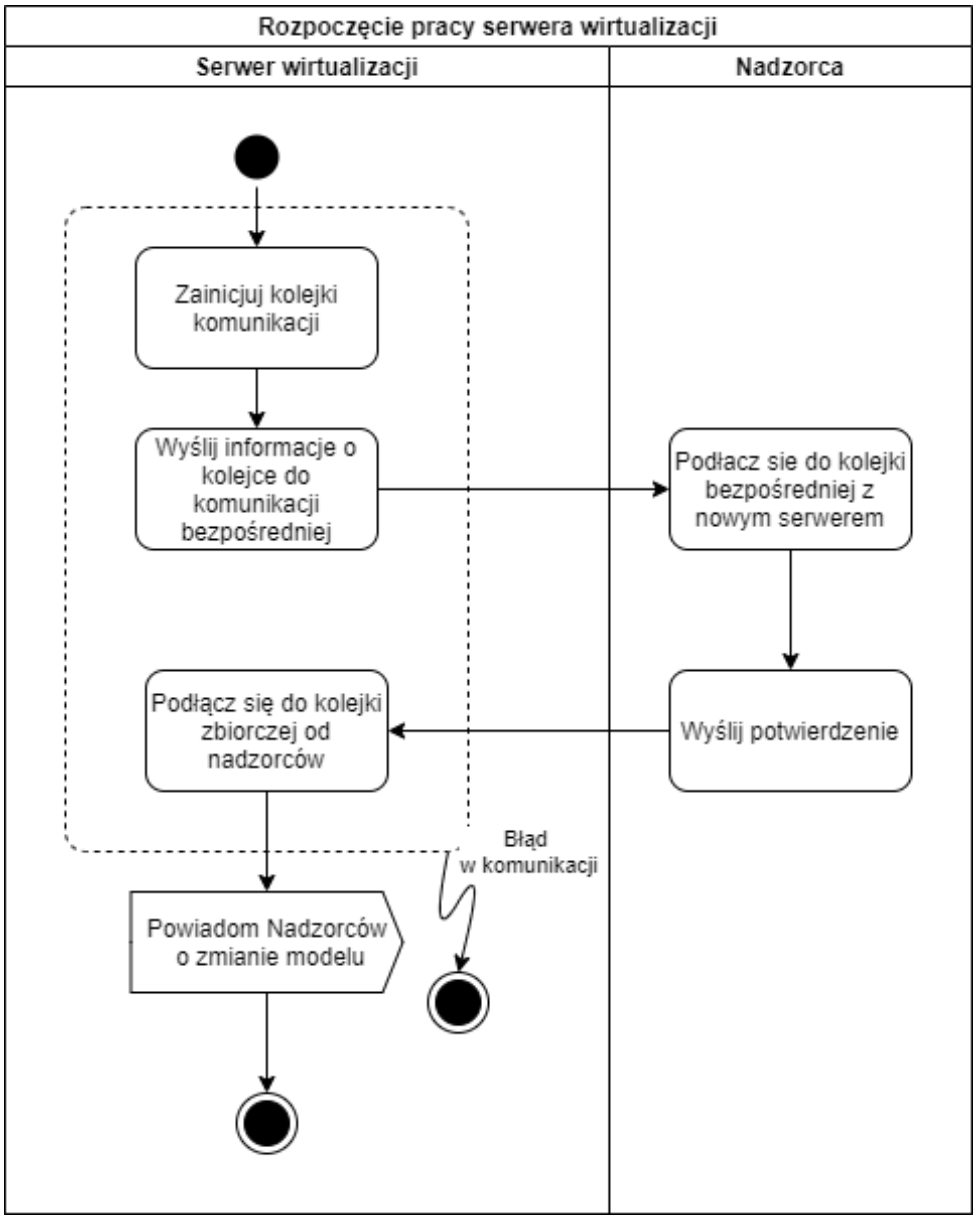


Rysunek 2.6: Diagram aktywności dla zakończenia sesji

Proces rozpoczyna się w momencie, gdy użytkownik odłączy się od systemu lub utraci połączenie. Po upłynięciu ustalonego czasu, jeżeli użytkownik nie podłączył się ponownie, maszyna zostaje wyłączona. serwer wirtualizacji informuję nadzorcę o utracie połączenia lub odłączeniu się użytkownika poprzez zmianę model. Decyzję o wyłączeniu maszyny podejmuje nadzorca. Powoduje to, że zmiana konfiguracji nadzorców będzie oznaczać spójną reakcje całego systemu. Dodatkowo umożliwia to w perspektywie czasu utworzenie bardziej złożonego algorytmu zarządzania zasobami.

2.4.3. Rozpoczęcie pracy serwera wirtualizacji

Proces opisuje przyjęcie nowego serwera wirtualizacji do systemu.



Rysunek 2.7: Diagram aktywności dla rozpoczęcia pracy serwera wirtualizacji

Serwer wirtualizacji bez działającego nadzorcy nie jest w stanie obsługiwać użytkowników. Oznacza to, że jeśli przy starcie nie wykryje brokera wiadomości lub nadzorcy po drugiej stronie kolejek[2] to się wyłączy. Jeżeli jednak komunikacja z nadzorcą jest możliwa, to serwer podłączy się do wspólnych kolejek oraz przekaże informacje nadzorcom o kolejce bezpośredniej. Gdy komunikacja będzie ustanowiona bezwarunkowo wyśle stan swojego modelu do nadzorców.

[2]Mechanizm potwierdzenia wykonania zadania

2.5. Komunikacja

2.5.1. Komunikacja wewnętrzna

Komunikacja wewnątrz systemu opiera się na kolejkach opisanych w opisie modułów. W celu uniknięcia wyścigów i utrzymania spójności modelu systemu pomiędzy nadzorcami ustalone są następujące zasady:

- Nadzorca może zmienić stan systemu jedynie w reakcji na odpowiedź serwera wirtualizacji. Odpowiedzi te wysyłane są do wszystkich nadzorców, dzięki czemu każdy nadzorca ma taki sam model systemu.

- Wiadomości przetwarzane są przez serwer wirtualizacji w sposób atomowy. Pojedyncza wiadomość musi zostać w pełni obsłużona zanim program przejdzie do obsługi kolejnej.

- Serwer wirtualizacji odpowiada na wiadomości wysyłając nowy stan maszyn. Jeżeli żądanie nie może być spełnione z powodu błędnego żądania, to serwer nie odpowiada na żądanie. Wyjątkiem jest żądanie o wysłanie aktualnego stanu maszyn.

- Z powodu asynchroniczności wiadomości moduły nie oczekują na odpowiedź. W przypadku nadzorcy przetwarzanie ódpowiedziźostanie uruchomione przez zmianę modelu.

- Do monitorowania utrzymania połączenia z brokerem użyty jest mechanizm zwracania wiadomości, które nie mogą zostać dostarczone[3]. Używając tego nadzorcy mogą wykryć, kiedy poszczególne serwery wirtualizacji przestaną działać, a serwery wirtualizacji - kiedy wszyscy nadzorcy przestaną działać.

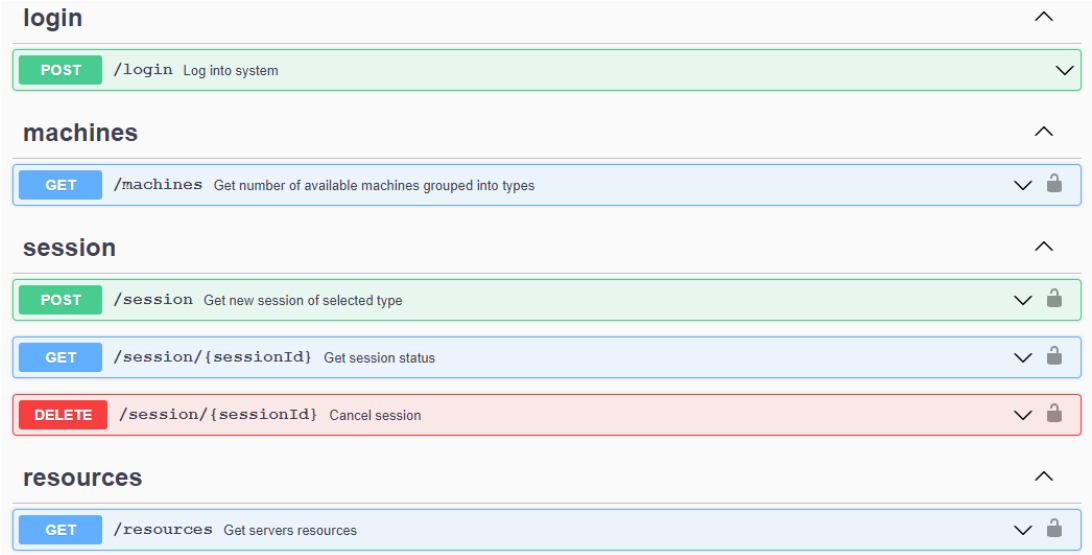
Opisane wyżej założenia pozwalają uniknąć problemu hazardów i wyścigów. Jeżeli wiele nadzorców wyśle do serwera wirtualizacji tą samą prośbę, np. o stworzenie sesji na konkretnej maszynie, to z atomowości obsługi sesja zostanie stworzona tylko dla pierwszego z nich. Serwer wirtualizacji wyśle wiadomość o aktualizacji stanu maszyn i zignoruje pozostałe prośby. Nadzorcy otrzymają zmianę stanów, co spowoduje wywołanie odpowiednich procedur. Dla pierwszego będzie to dalsza część procesu tworzenia sesji, a pozostali nadzorcy pozostaną w procesie wyszukiwania maszyny do sesji.

[3]Mechanizm zwracania wiadomości

2.5.2. Komunikacja zewnętrzna

Komunikacja aplikacji klienckiej oraz panelu administratora z systemem - nadzorcą - rozwiązana jest za pomocą REST API[4]. W zależności od konfiguracji nadzorcy wiadomości mogą być wysyłane za pomocą protokołu HTTPS[5], który zapewnia ich szyfrowanie. W tym celu wymagane jest, aby na adres, pod którym udostępniony będzie system, wystawiony był odpowiedni certyfikat[6], gwarantujący jego tożsamość.

Całość specyfikacji API umieszczona jest w załączniku. Poniżej znajduje się zestawienie oraz krótki opis endpointów.



Rysunek 2.8: Endpointy API

- Login - służy do logowania do systemu; współdzielony przez aplikację kliencką oraz panel administracyjny. Poprawne zalogowanie zwraca token do dalszej autoryzacji.
- Machines - służy do pobierania przez aplikację informacji o typach i ilości dostępnych maszyn. Ten endpoint, oraz wszystkie następne wymagają autoryzacji poprzez umieszczenie tokenu otrzymanego podczas logowania w odpowiednim nagłówku wiadomości, oraz dostępne są tylko dla użytkownika.
- Session - pozwala na wysłanie prośby o uzyskanie sesji, pobranie stanu sesji oraz jej anulowanie. Utworzenie sesji jest możliwe poprzez `POST` z typem maszyny. W odpowiedzi użytkownik dostaje częściowo wypełniony obiekt sesji zawierający id umożliwiające dalsze zapytania. `GET` zwraca obiekt sesji z aktualnym stanem. Jeżeli sesja jest gotowa, to zawiera on

[4]Opis REST API
[5]Specyfikacja protokołu HTTP Over TLS
[6]Opis certyfikatu TLS/SSL

też adres, z którym należy nawiązać połączenie RDP. `DELETE` umożliwia anulowanie sesji.

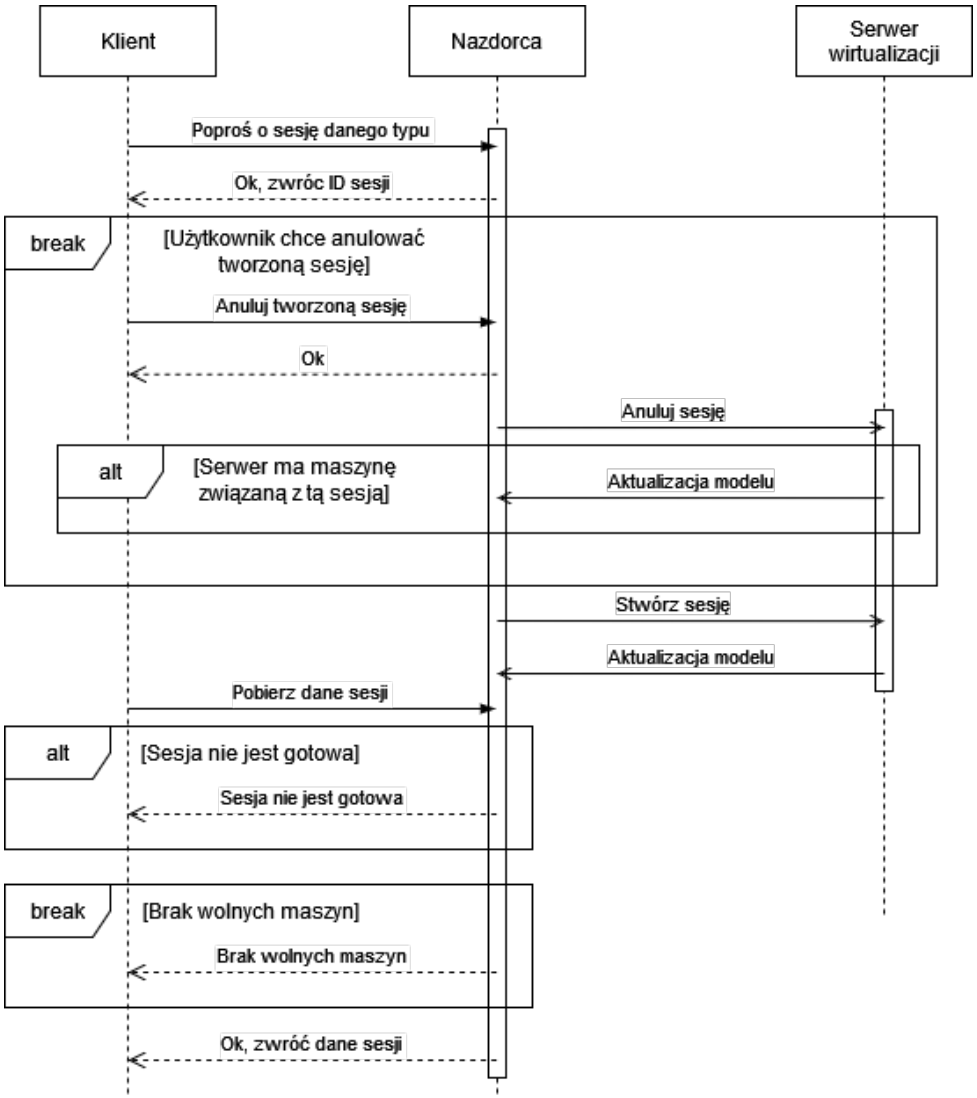
- Resources - udostępnia informację o zasobach działających serwerów wirtualizacji. Dostępny jedynie dla administratora.

Ważną informacją jaką musi posiadać system to fakt, czy użytkownik rzeczywiście jest podłączony do maszyny wirtualnej. System uzyskuje tą informację komunikując się z brokerem wiadomości odpowiedzialnym za komunikację z użytkownikami. Każda aplikacja kliencka po podłączeniu się do maszyny wirtualnej poprzez protokół RDP tworzy kolejkę o takiej nazwie jak uzyskany identyfikator sesji. Serwer wirtualizacji sprawdza co jakiś czas, czy na końcu kolejki istnieje jakikolwiek konsument. Gdy użytkownik się rozłączy to kolejka jest usuwana przez aplikację kliencką, co pozwala serwerowi wykryć odłączenie się użytkownika.

2.6. Sekwencje komunikacji

2.6.1. Utworzenie sesji

Celem tej sekwencji komunikacji jest odnalezienie istniejącej już sesji lub stworzenie nowej. Zakładamy, że w systemie zawsze jest jakaś wolna maszyna. Inaczej zgłaszamy użytkownikowi błąd.



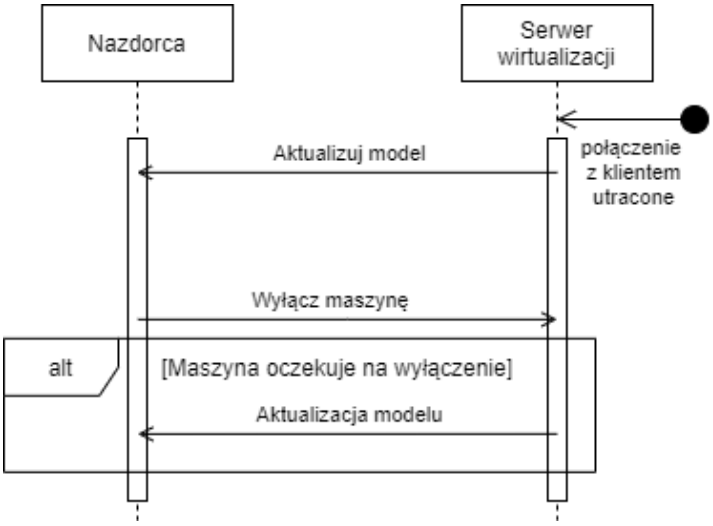
Rysunek 2.9: Sekwencja komunikacji utworzenia sesji

Po prośbie użytkownika nadzorca znajduje wolna maszynę i prosi konkretny serwer wirtualizacji aby spróbował utworzyć sesje dla pewnego użytkownika. Gdy to się uda, ten wysyła zbiorczą kolejką do nadzorców informacje o zmianie modelu. W przeciwnym przypadku nadzorca powtarza wyszukanie wolnej maszyny. O tym, czy nadzorca musi powtórzyć wyszukiwanie decyduje stan otrzymanej maszyny (m.in. czy sesja do niej przypisana należy do tego użytkownika).

Może się zdarzyć także anulowanie wyszukiwania przez użytkownika. Wtedy jeżeli maszyna jest już przydzielona użytkownikowi, to serwer wirtualizacji jest powiadamiany o anulowaniu sesji, aby ją anulował. Jeżeli nie została jeszcze utworzona, to nadzorca dopilnuje aby więcej nie szukać sesji lub wyłączy ją w miarę potrzeby.

2.6.2. Zakończenie sesji

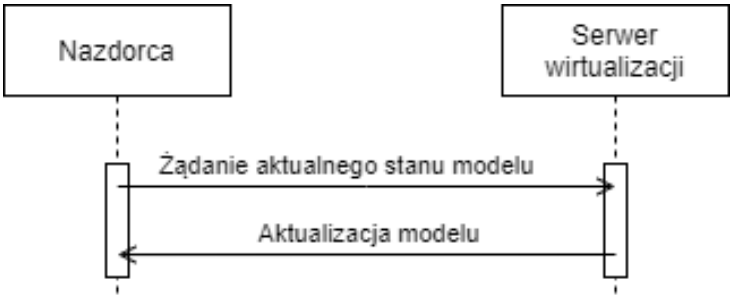
Sekwencja ta zainicjowana jest poprzez utracenie połączenia z użytkownikiem maszyny. Serwer powiadamia o tym fakcie nadzorców, po czym oczekuje na polecenie wyłączenia maszyny przesłane przez nadzorcę. Serwer może odmówić z powodów różnic modelu, lub jeżeli maszyna znów jest używana przez użytkownika, ignorując wiadomość.



Rysunek 2.10: Sekwencja komunikacji zakończenia sesji

2.6.3. Aktualizacja stanu

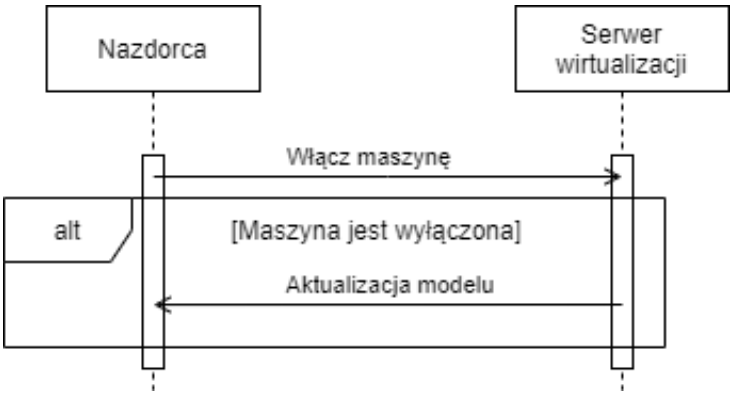
Nadzorca może w każdej chwili poprosić wszystkie serwery wirtualizacji o przesłanie ich aktualnego stanu poprzez wspólna kolejkę do serwerów wirtualizacji. Serwery muszą bezwarunkowo odpowiedzieć aktualnym stanem do wspólnej kolejki zwrotnej.



Rysunek 2.11: Sekwencja komunikacji aktualizacji stanu systemu

2.6.4. Włączenie maszyny

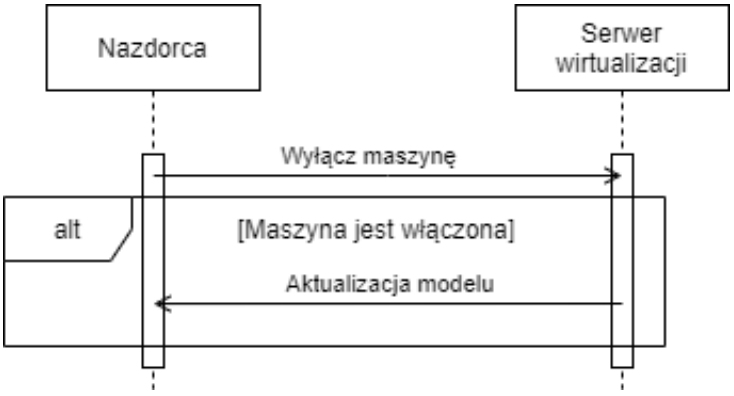
Nadzorca może poprosić konkretny serwer wirtualizacji aby utworzył maszynę o konkretnej nazwie. Jeżeli maszynie nie istnieje, to zostanie uruchomiona oraz serwer odeśle powiadomienie zbiorcza kolejką o zmianie modelu. W przeciwnym wypadku nie zrobi nic.



Rysunek 2.12: Sekwencja komunikacji włączenia maszyny

2.6.5. Wyłączenie maszyny

Nadzorca może poprosić konkretny serwer wirtualizacji aby wyłączył konkretną maszynę wirtualną. Jeżeli maszynę można wyłączyć to zostanie on wyłączona. Następnie serwer wirtualizacji odeśle powiadomienie zbiorcza kolejką o zmianie modelu. W przeciwnym wypadku nie zrobi nic.



Rysunek 2.13: Sekwencja komunikacji wyłączenia maszyny

2.7. Wykorzystane technologie

2.8. Wymagania systemu

System do poprawnej pracy wymaga konfiguracji środowiska na wielu płaszczyznach.

2.8.1. Komunikacja sieciowa między modułami

System składa się przynajmniej z:

1. Przynajmniej jednego nadzorcy.

2. Serwerów wirtualizacji (może być ich 0).
3. Maszyny wirtualne uruchamiane na serwerach wirtualizacji.
4. Serwera HTTP udostępniającego panel administracyjny.
5. Brokera wiadomości do komunikacji wewnętrznej.
6. Brokera wiadomości do komunikacje zewnętrznej (może być tym samym brokerem co wewnętrzny).
7. Dowolnej liczby aplikacji klienckich.

Dostępność wewnętrznego brokera

Broker wewnętrzny powinien być dostępny dla każdego nadzorcy oraz każdego serwera wirtualizacji. Jest to wymagane do prawidłowej komunikacji pomiędzy nadzorcami a serwerami wirtualizacji.

Dostępność zewnętrznego brokera

Broker zewnętrzny powinien być dostępny dla każdego serwera wirtualizacji oraz dla każdego klienta łączącego się z systemem. Jest to wymagane do stwierdzenia czy użytkownik nadal jest podłączony do maszyny wirtualnej.

Dostępność nadzorców

Nadzorcy powinni być dostępni dla aplikacji klienckich, którzy poprzez nich komunikują się z resztą systemu. Administrator po pobraniu panelu administracyjnego z serwera http powinien móc wysyłać zapytania do nadzorców.

Dostępność serwerów wirtualizacji

Serwery wirtualizacji nie muszą być dostępne dla żadnego z modułów.

Dostępność serwera http z panelem administracyjnym

Serwer HTTP powinien być dostępny dla każdego z administratorów.

Dostępność maszyn wirtualnych

Maszyny wirtualne powinny być dostępne dla każdego z klientów. Jest to potrzebne do pracy na nich poprzez protokół RDP.

2.8.2. Wymagania aplikacji klienckiej

Dla systemu Windows aplikacja kliencka jako plik wykonywalny pobrana z oficjalnych wydań powinna uruchomić się bez żadnych wcześniejszych konfiguracji. Do działania skorzysta ona z klienta RDP dostarczonego przez Microsoft wraz z systemem Windows. Aplikacja na pewno działa dla systemu Windows 10 i nowszych. Dla starszych wersji systemu Windows aplikacja może nie działać prawidłowo.

W przypadku systemu Linux należy posiadać środowisko graficzne oraz mieć zainstalowanego klienta FreeRDP. Pozostałe zależności są dostarczone wewnątrz pliku `.appimage`. Aplikacja na pewno działa na dystrybucji bazowanych na Debianie oraz Archu. Dla innych dystrybucji aplikacja może nie działać prawidłowo.

2.8.3. Wymagania aplikacji nadzorcy i serwera panelu administracyjnego

Do uruchomienia aplikacji nadzorcy i serwera HTTP z panelem administracyjnym potrzeba zainstalowanego `Dockera` w systemie. Każdy z tych dwóch modułów można zbudować do kontenera, w którym wszystkie zależności zostaną spełnione.

2.8.4. Wymagania serwera wirtualizacji

Serwer wirtualizacji oprócz działającej usługi `Dockera` potrzebuje dodatkowych usług. Do prawidłowego działania wymagana jest działająca usługa zarządcy wirtualnych maszyn `libvirt`. Aby prawidłowo uruchomić maszynę wirtualna potrzebna jest uruchomiona usługa zapory sieciowej oraz pakiet `dnsmasq`. Wymagana jest także inicjalizacja struktur usługi `vagrant` dla użytkownika uruchamiającego serwer wirtualizacji. Konkretnie chodzi o utworzenie folder `.vagrant.d` w folderze domowym użytkownika.

Aby uruchamiane maszyny wirtualne były dostępne dla innych urządzeń potrzeba utworzyć interfejs sieciowy w trybie bridge. Nazwa interfejsu powinna być przekazana do pliku konfiguracyjnego serwera wirtualizacji. Maszyny wirtualne uzyskają wtedy dostęp do sieci w sposób jakby były fizycznymi komputerami w sieci.

Czasami przy uruchomieniu maszyny wirtualnej przy użyciu `vagranta` oraz uruchomieniu wybranych kontenerów w `dockerze` komunikacja sieciowa z maszyny wirtualnej poprzez podłączony interfejs w trybie bridge może zostać ograniczona. Wtedy należy pilnować by w trakcie działanie systemu zapora sieciowa posiadała zasadę na samej górze łańcucha `FORWARD`, która będzie bezwarunkowo zezwalać trasować pakiety z urządzeń w trybie bridge.

2.8.5. Automatyzacja konfiguracji

Przykładowa konfiguracja oraz narzędzia do automatycznej konfiguracji przed uruchomieniem dostępne sa w module `configuration`.

Składa się on ze skryptów konfiguracyjnych Ansible, nazywanych dalej playbook, oraz zmiennych opisujących konfigurowane komputery.

By uruchomić skrypt dla pewnego systemu operacyjnego, który chcemy skonfigurować, musi on spełniać wymogi opisane w dokumentacji Ansible.

Grupy i zmienne

Konfiguracja podzielona jest na 2 grupy: `overseer` i `virtsrv`. Odpowiadają one za reprezentację systemów przygotowanych odpowiednio dla nadzorców oraz serwerów wirtualizacji. Dodatkowo wytyczona jest sztuczna grupa `all` opisująca wszystkie konfigurowane systemy.

Jedyną wspólną zmienną dla wszystkich maszyn jest nazwa użytkownika, który będzie uruchamiał i odpowiadał za zasoby systemu OneClickDesktop.

Dla każdej maszyny z osobna należy dane dostępowe do komunikacji z konfigurowanym systemem. Dodatkowo należy podać hasło dla dostępu do uprawnień superużytkownika.

Zmienne dla serwera wirtualizacji

Playbook dla serwera wirtualizacji wykona wszystkie kroki opisane w 2.8.4. Aby tego dokonać należy zdefiniować dane dla tworzonego interfejsu typu bridge. Należy zdefiniować nazwę interfejsu sieciowego, który zostanie połączony do nowo tworzonego urządzenia bridge. Skrypt nada mu nazwę, która także trzeba zdefiniować. Playbook korzysta z NetworkManagera do zmiany konfiguracji, więc trzeba podać nazwę *connection* (jednostka logiczna w NetworkManagerze) skojarzonego z początkowo wybranym interfejsem sieciowym.

Zmienne dla nadzorcy

Aby uruchomić nadzorce wystarczą wspólne wymagania dla wszystkich grup.

2.8.6. Wymagania szablonu maszyny wirtualnej

Maszyna wirtualna uruchamiana w ramach systemu OneClickDesktop musi być w postaci Vagrant Boxa. Przykładowy box został utworzony w czasie rozwoju systemu i dostępny jest w chmurze Vagrant pod nazwą smogork/archlinux-rdp. Jest to Arch Linux, który spełnia wszystkie wymagania aby zostać uruchomionym w ramach systemu.

Do przygotowania szablonu proponujemy aby skorzystać właśnie z tego obrazu. Jeżeli jednak jest potrzeba utworzenie niestandardowego szablonu to musi on:

- Spełniać minimalne wymagania opisane w dokumentacji Vagranta.
- Mieć zainstalowany pewien menadżer okienek. Przykładowy szablon zawiera menadżera okienek XFCE.
- Przy uruchomieniu udostępniać usługę zdalnego pulpitu RDP. Przykładowy szablon korzysta z implementacji xrdp.
- Każdy nowy interfejs sieciowy powinien być skonfigurowany aby uzyskać adres IP z usługi DHCP.
- Przy starcie systemu urchamiać usługę qemu-guest-agent.

Nie ma ograniczenia na uruchamiany system operacyjny wewnątrz systemu OneClickDesktop. Jedynie taki szablon musi spełniać powyższe wymagania.

Aby zbudować własny szablon należy skonsultować się z dokumentacją wtyczki vagrant-libvirt.

2.9. Uruchomienie systemu

W tym podrozdziale zakładamy, że każdy system operacyjny został skonfigurowany poprawnie zgodnie z opisem w 2.8. Wtedy można przejść do uruchamiania systemu.

2.9.1. Pozyskanie aplikacji klienckiej

Zalecanym sposobem pozyskania aplikacji klienckiej jest udanie się do sekcji z wydaniami kodu modułu aplikacji klienckiej oraz pobranie najnowszej wersji dla wybranego systemu operacyjnego. Przy pierwszym uruchomieniu trzeba pobrać archiwum plików zawierające aplikację oraz plik konfiguracyjny. Bez pliku konfiguracyjnego aplikacja nie uruchomi się.

2.9.2. Zbudowanie kontenerów

Moduły: panelu administracyjnego, nadzorcy i serwera wirtualizacji można uruchomić pod postacią kontenera Dockerowego. Jest to zalecany sposób uruchamiania tych trzech modułów.

Ujednolicony sposób budowania kontenerów

Każdy z tych 3 modułów ma przygotowany skrypt `build.sh`, który powinien prawidłowo zbudować kontener. Skrypt ten wykona polecenie do budowania i oznaczy kontener odpowiednią

nazwą. Należy wykonać go zawsze z poziomu głównego folderu repozytorium modułu. Każdy kontener przy starcie wykona skrypt `assets/entry_point.sh`.

Budowanie kontenera serwera wirtualizacji

Kontener serwera wirtualizacji wymaga specjalnie przygotowanego wcześniej kontenera zawierającego Wszystkie potrzebne zależności do uruchomienia aplikacji. Aby go zbudować należy skorzystać z pliku `runtime_container/Dockerfile` i oznaczyć go nazwą `one-click-desktop/virtualization-server-runtime`. Kontener zawierający aplikacje w czasie budowania będzie oczekiwał, ze taki obraz istnieje.

Po zbudowaniu kontenera z zależnościami można przystąpić do zbudowania kontenera głównego. Należy do tego celu skorzystać z dostarczonego pliku `Dockerfile` w głównym folderze repozytorium.

Budowanie kontenera nadzorcy i panelu administratora

W przypadku tych dwóch modułów nie trzeba wykonać żądnych dodatkowych przygotowań. Wystarczy skorzystać z dostarczonego pliku `Dockerfile` w głównym folderze repozytorium.

2.9.3. Budowanie modułów

Każdy z modułów posiada dokładną instrukcję budowania w pliku `README.md`. Uruchamianie zbudowanych modułów poleca się tylko w przypadku rozwoju aplikacji. Przy wdrażaniu systemu najszybciej i najbezpieczniej jest skorzystać z metod opisanych w 2.9.1 i 2.9.2.

Moduły napisane w technologii .NET

W przypadku budowania modułu nadzorcy i serwera wirtualizacji potrzebne sa narzędzia deweloperskie .NET 5.0. W trakcie budowania aplikacji wszystkie użyte biblioteki zostaną pobrane przy użyciu menadżera pakietów Nuget. Przy uruchomieniu serwera wirtualizacji potrzebny będzie dodatkowo do wymagań zdefiniowanych w 2.8.4 zainstalowany vagrant wraz z wtyczka vagrant-libvirt w wersji przynajmniej 0.7.0.

Moduły napisane w technologii Node

W przypadku budowania modułów aplikacji klienckiej oraz panelu administracyjnego potrzebny będzie pakiet Node. Przed zbudowaniem aplikacji należy pobrać wszystkie użyte biblioteki przy użyciu menadżera pakietów npm. Są one zdefiniowane w projekcie aplikacji i zostaną pobrane automatycznie.

2.9.4. Konfiguracja aplikacji klienckiej

Aplikacja kliencka wymaga pliku konfiguracyjnego o nazwie `config.json` w tym samym folderze co plik wykonywalny. W pliku konfiguracyjnym użytkownik musi podać:

- Adres jednego z nadzorców(`basePath`) - adres musi być w formacie URI.
- Adres zewnętrznego brokera wiadomości(`rabbitPath`) - adres musi być w formacie URI.
- Czy aplikacja powinna używać danych dostępowych takich samych jak przy logowaniu do systemu(`useRdpCreadentials`) - `true` albo `false`.
- Czy aplikacja powinna uruchamiać zintegrowanego klienta RDP(`startRdp`) - `true` albo `false`. Przydaje się to przy wykorzystaniu innego niż zalecany klient RDP. Aplikacja po uzyskaniu sesji wyświetli dane dostępowe do przypisanej maszyny wirtualnej.

2.9.5. Konfiguracja panelu administracyjnego

Przy uruchomieniu panelu administracyjnego trzeba przekazać adres jednego z nadzorców. Aby tego dokonać należy ustawić zmienną środowiskową `API_URL` na adres dostępowy jednego z nadzorców.

2.9.6. Konfiguracja nadzorcy

Nadzorca posiada wiele parametrów związanych z działaniem całego systemu. Kontroluje on między innymi czas oczekiwania na powrót użytkownika do porzuconej sesji. Wszystkie parametry należy przekazać mu poprzez plik konfiguracyjny.

Nazwa pliku konfiguracyjnego musi spełniać wzorzec `appsettings.${Nazwa_srodowiska}.ini`. W przypadku uruchamiania aplikacji wewnątrz kontenera ustawione jest środowisko o nazwie `Production`. Aplikacja nadzorcy domyślnie przeszukuje folder `./config` w poszukiwaniu plików konfiguracyjnych. Można zmienić ścieżkę do folderu konfiguracyjnego poprzez parametr uruchomienia `-c path/to/other/config/folder/`.

W pliku konfiguracyjnym nadzorca poszukuje 2 specjalnych sekcji:

- `JwtSettings` - przechowuje parametr wykorzystywany do generowania unikatowych tokenów autoryzacyjnych
- `OneClickDesktop` - przechowuje parametry związane z działaniem systemu OneClickDesktop.

Poza tymi sekcjami można tam umieścić dowolne sekcje, które będzie przetwarzać ASP.NET. Jednak warte uwagi są standardowe sekcje:

- `Logging` - parametry loggera dostarczonego przez Microsoft.
- `Kestrel` - parametry serwera HTTP udostępniającego aplikację. W wypadku aplikacji nadzorcy bez ustawienia certyfikatu nie zadziała ona w trybie HTTPS.

Pozostają jeszcze 2 ważne parametry bez sekcji zaimplementowane przez ASP.NET:

- `AllowedHosts` - lista adresów z których zapytania będą obsługiwane.
- `urls` - lista adresów na których aplikacja będzie nasłuchiwać zapytań.

Parametry sekcji `JwtSettings`

Sekcja ta zawiera jedynie parametr `Secret`. Należy go ustawić na dowolny ciąg znaków.

Parametry sekcji `OneClickDesktop`

Sekcja zawiera parametry związane z komunikacja pomiędzy jednostkami systemu oraz kilka parametrów określających interwały czasowe zdarzeń. W dostarczonej przykładowej konfiguracji wykomentowane wartości oznaczają wartości domyślne.

- `OverseerId` - identyfikator nadzorcy używany w komunikacji poprzez wewnętrznego brokera wiadomości. Powinna być unikatowa w skali jednej instancji systemu OneClickDesktop. Domyślnie przyjmuje wartość `overseer-test`.
- `RabbitMQHostname` - adres dostępowy wewnętrznego brokera wiadomości. Domyślnie przyjmuje wartość `localhost`. Bez działającego procesu brokera pod tym adresem aplikacja wyłączy się z błędem zaraz po uruchomieniu.
- `RabbitMQPort` - port dostępowy wewnętrznego brokera wiadomości. Domyślnie przyjmuje wartość `5672`. Bez działającego procesu brokera pod tym portem aplikacja wyłączy się z błędem zaraz po uruchomieniu.
- `ModelUpdateInterval` - ilość sekund co ile nadzorca prosi serwery wirtualizacji o aktualizacje modelu. Domyślnie przyjmuje wartośc `60`. Zalecane jest aby wartość tego parametru opisywała czas od 30 do 60 sekund.
- `DomainShutdownTimeout` - ilość minut ile musi upłynąć od oznaczenia maszyny wirtualnej stanem `Oczekiwanie na wyłączenie maszyny` do jej wyłączenia. Domyślnie przyjmuje wartość `15`.

- `DomainShutdownCounterInterval` - ilość sekund co ile nadzorca sprawdza czy maszyna wirtualna nadal oczekuje na zamknięcie. Zaleca się aby ten czas dzielił czas z parametru `DomainShutdownTimeout` na równe części. Takich części nie powinno być mniej nić 10 ale także więcej niż 100. Każde takie sprawdzenie zużywa czas procesora.

2.9.7. Konfiguracja serwera wirtualizacji

Serwer wirtualizacji posiada wiele parametrów związanych z zasobami przekazanymi do dyspozycji systemu. Wszystkie parametry należy przekazać mu poprzez pliki konfiguracyjne.

Pliki te muszą znaleźć się w jednym folderze. Głównym plikiem konfiguracyjnym jest `virtsrv.ini`. Zawiera on parametry związane z komunikacją wewnątrz i na zewnątrz systemu. Dodatkowo znajdują się tam informacje o udostępnionych zasobach dla serwera wirtualizacji. Dodatkowe pliki konfiguracyjne reprezentują typy maszyn wirtualnych uruchamianych na tym serwerze wirtualizacji. Jedynym odstępstwem od reguły jest plik konfiguracyjny dla pakietu NLog. Musi on się znajdować w tym samym folderze co aplikacja oraz nazywać się `NLog.config`.

Aplikacja serwera wirtualizacji domyślnie przeszukuje folder `./config` w poszukiwaniu plików konfiguracyjnych. Można zmienić ścieżkę do folderu konfiguracyjnego poprzez parametr uruchomienia `-c path/to/other/config/folder/`.

Główny plik konfiguracyjny

W głównym pliku konfiguracyjnym możemy wyróżnić 2 sekcje:

- `OneClickDesktop` - przechowuje parametry związane z działaniem systemu OneClickDesktop.
- `ServerResources` - przechowuje parametry określające zgrubnie wszystkie udostępnione zasoby.

W sekcji `OneClickDesktop` występują parametry:

- `VirtualizationServerId` - identyfikator serwera wirtualizacji używany w komunikacji poprzez wewnętrznego brokera wiadomości. Powinna być unikatowa w skali jednej instancji systemu OneClickDesktop. Domyślnie przyjmuje wartość `virtsrv-test`.
- `OversserCommunicationShutdownTimeout` - po upływie tylu sekund bez komunikacji od jakiegokolwiek nadzorcy serwer wirtualizacji uznaje, że zabrakło nadzorców w systemie. Nastąpi wtedy wyłączenie aplikacji serwera wirtualizacji. Domyslnie parametr przyjmuje wartość `120`. Zaleca się aby wartość parametru była większa niż dwukrotność parametru `ModelUpdateInterval` z konfiguracji nadzorcy opisanej w 2.9.6.

- `VagrantFilePath` - ścieżka do specjalnie przygotowanego pliku wsadowego dla Vagranta. Wykorzystywany jest przez system do uruchamiania i wyłączania maszyn wirtualnych. Aplikacja nie zadziała prawidłowo bez ustawionej wartości parametru.

- `PostStartupPlaybook` - skrypt wykonywany przy pomocy Ansible'a zaraz po uruchomieniu maszyny wirtualnej. Koniecznie należy przetestować czy konfiguracja wykonuje się prawidłowo. Przy każdym błędzie wykonania skryptu maszyna wirtualna zostanie usunięta. Domyślnie przyjmuje wartość `res/poststartup_playbook.yml`.

- `VagrantboxUri` - identyfikator szablonowej maszyny wirtualnej. Musi ona spełniać szereg wymogów opisanych w 2.8.6. W przeciwnym wypadku system nie będzie w stanie uruchomić takiego szablonu. Aplikacja nie zadziała prawidłowo bez ustawionej wartości parametru.

- `BridgeInterfaceName` - nazwa interfejsu sieciowego w skonfigurowanego w trybie bridge, do którego zostanie podłączona każda uruchomiona maszyna wirtualna. Domyślnie przyjmuje wartość `br0`.

- `BridgedNetwork` - adres sieci, do której podłączona zostanie maszyna wirtualna poprzez `BridgeInterfaceName`, podany w formacie CIDR. Jeżeli maszyna nie będzie posiadać adresu z podanej sieci po uruchomieniu to zostanie wyłączona. Aplikacja nie zadziała prawidłowo bez ustawionej wartości parametru.

- `InternalRabbitMQHostname` - adres dostępowy wewnętrznego brokera wiadomości. Domyślnie przyjmuje wartość `localhost`. Bez działającego procesu brokera pod tym adresem aplikacja wyłączy się z błędem zaraz po uruchomieniu.

- `InternalRabbitMQPort` - port dostępowy wewnętrznego brokera wiadomości. Domyślnie przyjmuje wartość `5672`. Bez działającego procesu brokera pod tym portem aplikacja wyłączy się z błędem zaraz po uruchomieniu.

- `ExternalRabbitMQHostname` - adres dostępowy zewnętrznego brokera wiadomości. Domyślnie przyjmuje wartość `localhost`. Bez działającego procesu brokera pod tym adresem aplikacja wyłączy się z błędem zaraz po uruchomieniu.

- `ExternalRabbitMQPort` - port dostępowy zewnętrznego brokera wiadomości. Domyślnie przyjmuje wartość `5673`. Bez działającego procesu brokera pod tym portem aplikacja wyłączy się z błędem zaraz po uruchomieniu.

- `ClientHeartbeatChecksForMissing` - liczba nieudanych sprawdzeń czy aplikacja kliencka

nadal jest połączona do maszyny wirtualnej. Po tej liczbie prób maszyna wirtualna oznaczana jest jako oczekująca na zamknięcie. Domyslnie przyjmuje wartośc `2`.

- `ClientHeartbeatChecksDelay` - czas w milisekundach pomiędzy sprawdzeniami, czy aplikacja kliencka nadal jest połączona do maszyny wirtualnej. Domyślnie przyjmuje wartość `10000`

W sekcji `ServerResources` występują parametry:

- `Cpus` - liczba wątków które może wykorzystać system. Domyślnie przyjmuje wartość `2`.
- `Memory` - ilość pamięci operacyjnej w MiB jaką może wykorzystać system. Domyślnie przyjmuje wartość `2048`.
- `Storage` - ilość przestrzeni dyskowej w GiB jaką może wykorzystać system. Domyślnie przyjmuje wartość `100`.
- `GPUsCount` - ilość kart graficznych przekazanych do systemu. Domyslnie przyjmuje wartośc `0`.
- `MachineTypes` - lista typów maszyn wirtualnych. Każda nazwa typu jest oddzielona od sąsiednich przecinkiem. Nazwa typu musi składać się ze znaków opisanych nastepującym wyrażeniem regularnym `[a-zA-Z0-9\-]`. Dla każdej z nazw wyszukiwany jest plik konfiguracyjny o o nazwie zaczynającej się nazwą typu i kończącą się `_template.ini`.

Gdy $n =$ `GPUsCount` jest większy od 0, wtedy w pliku muszą znaleźć się sekcje od `ServerGPU.1` do `ServerGPU.`n włącznie. W przeciwnym wypadku serwer wirtualizacji zakończy się z błędem zaraz po uruchomieniu.

Każda z tych sekcji zawiera parametr `AddressCount`, który określa jak duża jest grupa IOMMU w której znajduje się przekazywane urządzenie PCI. W zależności $m =$ `AddressCount` w tej sekcji muszą znaleźć się parametry od `Address_1` do `Address_`m włącznie. W przeciwnym wypadku serwer wirtualizacji zakończy się z błędem zaraz po uruchomieniu. Każdy z tych parametrów opisuje adres pojedynczego urządzenia na magistrali PCI. Adres na magistrali PCI musi zostać opisany w formacie uzyskanym z polecenia `lspci -{domain:4}:{bus:2}:{slot:2}.{function:1}`.

Przykładowa sekcja zasobów z jednym GPU przekazanym do systemu:

```
[ServerResources]
Cpus=6
Memory=4096
```

```
Storage=200
GPUsCount=1
MachineTypes=cpu,gpu

[ServerGPU.1]
AddressCount=2
Address_1=0000:03:00.0
Address_2=0000:03:00.1
```

Pliki konfiguracyjne opisujące typy maszyn wirtualnych

W pliku opisującym typy zawarte jest ile potrzeba zasobów aby uruchomić maszynę wytworzonej z tego typu.

Każdy typ ma przypisaną nazwę. Oznaczmy ją jako `${template_name}`. Plik opisujący typ musi mieć nazwę `${template_name}_template.ini`. Plik konfiguracyjny dla wybranego typu musi znajdować się w folderze z pozostałymi plikami konfiguracyjnymi. Jeżeli dla jakiejkolwiek nazwy typu aplikacja nie znajdzie pliku konfiguracyjnego to zakończy się z błędem zaraz po uruchomieniu.

W znalezionym pliku musi być sekcja o nazwie `${template_name}_template`. Zwiera ona parametry:

- `HumanReadableName` - nazwa reprezentowana w aplikacji klienckiej dla użytkownika. Domyślnie przyjmuje wartość `${template_name}`.
- `Cpus` - liczba wątków potrzebna do uruchomienia maszyny tego typu. Domyślnie parametr przyjmuje wartość `2`.
- `Memory` - ilość pamięci operacyjnej w MiB potrzebnej do uruchomienia maszyny tego typu. Domyślnie parametr przyjmuje wartość `512`.
- `Storage` - ilość przestrzeni dyskowej w GiB potrzebnej do uruchomienia maszyny tego typu. Domyślnie parametr przyjmuje wartość `20`.
- `AttachGpu` - informacja czy maszyna tego typu przy uruchomieniu powinna mieć przekazaną kartę graficzną. Domyślnie parametr przyjmuje wartość `false`.

2.9.8. Procedura uruchomienia

Prawidłowy start systemu powinien zachować następująca kolejność:

1. Uruchomić zewnętrznego i wewnętrznego brokera wiadomości.
2. Poczekać na prawidłowy start brokerów wiadomości.
3. Uruchomić wymaganą liczbę nadzorców.
4. Poczekać na prawidłowy start przynajmniej jednego z nadzorców.
5. uruchomić serwer HTTP udostępniający panel administracyjny.
6. Uruchomić wszystkie serwery wirtualizacji.
7. Poczekać aż w systemie znajdą się w pełni uruchomione maszyny wszystkich zarejestrowanych typów.

Po opisanych krokach system jest gotowy do podłączenia się przez użytkowników. Ważne jest aby przed uruchomieniem jakiegokolwiek nadzorcy brokery wiadomości były już prawidłowo zainicjalizaowane. W przeciwnym wypadku aplikacja nadzorcy zakończy się z błędem. Serwer wirtualizacji także zakończy się z błędem, jeżeli zabraknie brokerów wiadomości. Dodatkowo, jeżeli nie będzie żadnego nadzorcy nasłuchującego poprzez wewnętrznego brokera wiadomości, serwer wirtualizacji zgłosi błąd i zakończy pracę.

2.10. Interakcja z systemem

3. Analiza rozwiązania

4. Podsumowanie

[1]

Bibliografia

[1] Nobody Jr. My article, 2006.

Wykaz symboli i skrótów

RDP	Remote Desktop Protocol
*	operator gwiazdka
~	tylda

Spis rysunków

Spis tabel

Spis załączników

1. Załącznik 1
2. Załącznik 2
3. Jak nie występują, usunąć rozdział.