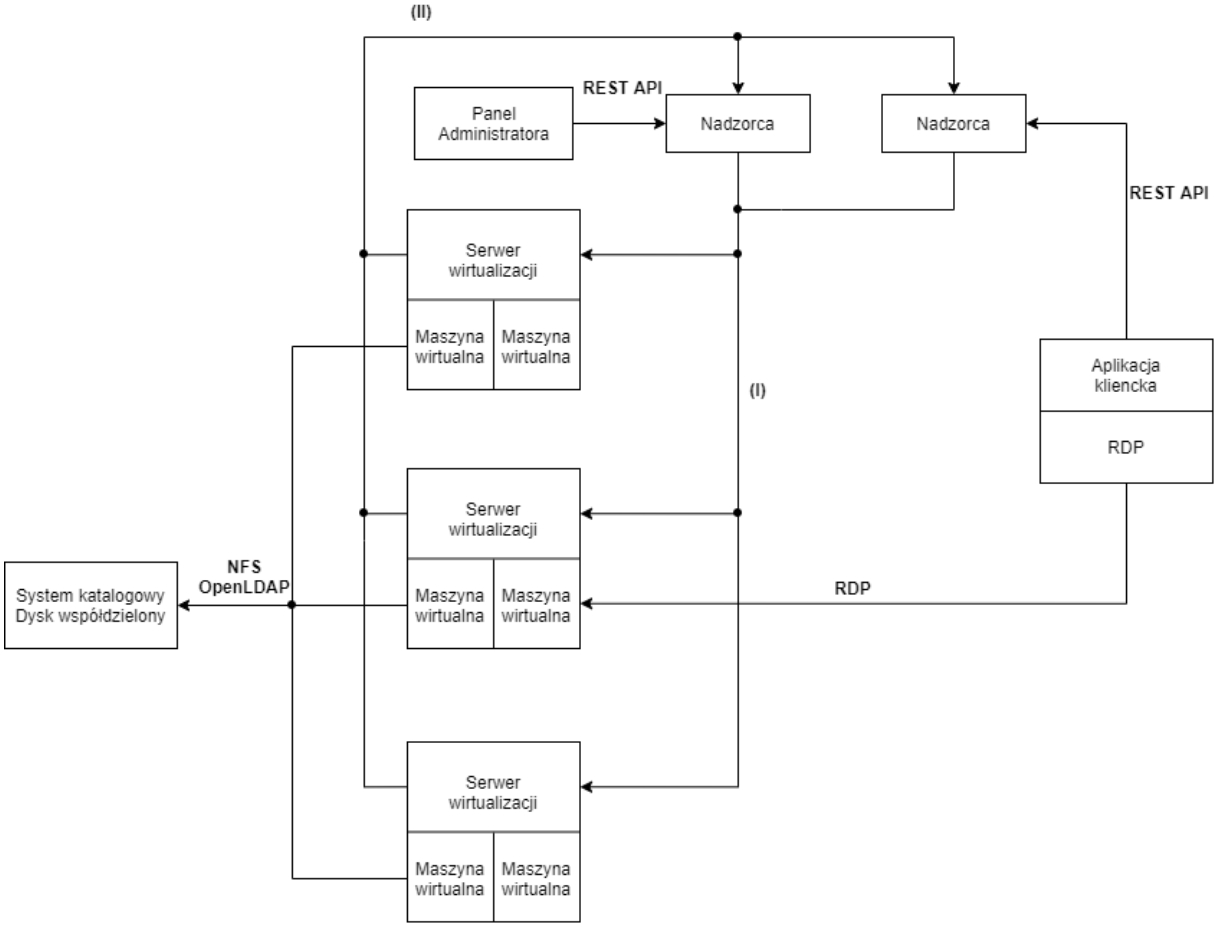
0.1. Architektura systemu

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 0.1: 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.

0.1.1. 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.

0.1.2. 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.

0.1.3. 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.

0.1.4. Panel administratora

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

0.1.5. 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]).

[1]Opis zachowania kolejek na wyłączność

(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.

0.2. Zewnętrzne narzędzia

0.2.1. Ansible

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

1. danych dostępowych do dysku sieciowego oraz wykorzystanym protokole,

2. danych dostępowych do usługi katalogowej.

Playbook można rozszerzać o potrzebne dane zależne od użycia.

0.2.2. 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

0.2.3. 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.

0.3. Technologie

- Aplikacja kliencka
  - Typescript[2] /Javascript[3]
  - Node.js[4] - środowisko uruchomieniowe używane do integracji z systemem użytkownika
  - Angular[5] - renderowanie widoków
  - Electron[6] - platforma programistyczna
  - Jest[7] - testy jednostkowe
  - Cypress[8] - testy integracyjne
- Panel administratora
  - Typescript/Javascript
  - Angular - platforma aplikacji WWW
  - Jest - testy jednostkowe
  - Cypress - testy integracyjne
- Nadzorca i serwer wirtualizacji

[2]Strona projektu Typescript
[3]Obecny standard języka Javascript
[4]Strona projektu Node.js
[5]Strona projektu Angular
[6]Strona projektu Electron
[7]Strona projektu Jest
[8]Strona projektu Cypress

  - C#[9]
  - RabbitMQ[10] - broker asynchronicznych wiadomości
  - Ansible[11] - konfigurowanie maszyn wirtualnych
  - Vagrant[12] - tworzenie obrazów maszyn wirtualnych oraz ich uruchamianie
  - libvirt[13] - zarządzanie maszynami wirtualnymi
  - OpenLDAP[14] - dostępu do systemu katalogowego
  - NFS[15] - dostęp do katalogów domowych z maszyny wirtualnej
  - Arch Linux[16] - system operacyjny uruchamiany przez maszyny wirtualne
  - GNU/Linux - wspierany system operacyjny
- Różne
  - Swagger Codegen[17] - automatyczna generacja API na podstawie specyfikacji
  - RDP[18] - łączenie ze zdalnymi sesjami

0.4. Stany biznesowe

0.5. Procesy biznesowe

0.6. Komunikacja

0.6.1. Komunikacja wewnętrzna

0.6.2. Komunikacja zewnętrzna

0.6.3. Sekwencje komunikacji

[9] Dokumentacja języka C#
[10] Strona projektu RabbitMQ
[11] Strona projektu Ansible
[12] Strona projektu Vagrant
[13] Strona projektu libvirt
[14] Strona projektu OpenLDAP
[15] Opis na stronie firmy Microsoft
[16] Strona systemu operacyjnego Arch Linux
[17] Opis narzędzia na stronie firmy Swagger
[18] Dokumentacja protokołu RDP od Microsoft