

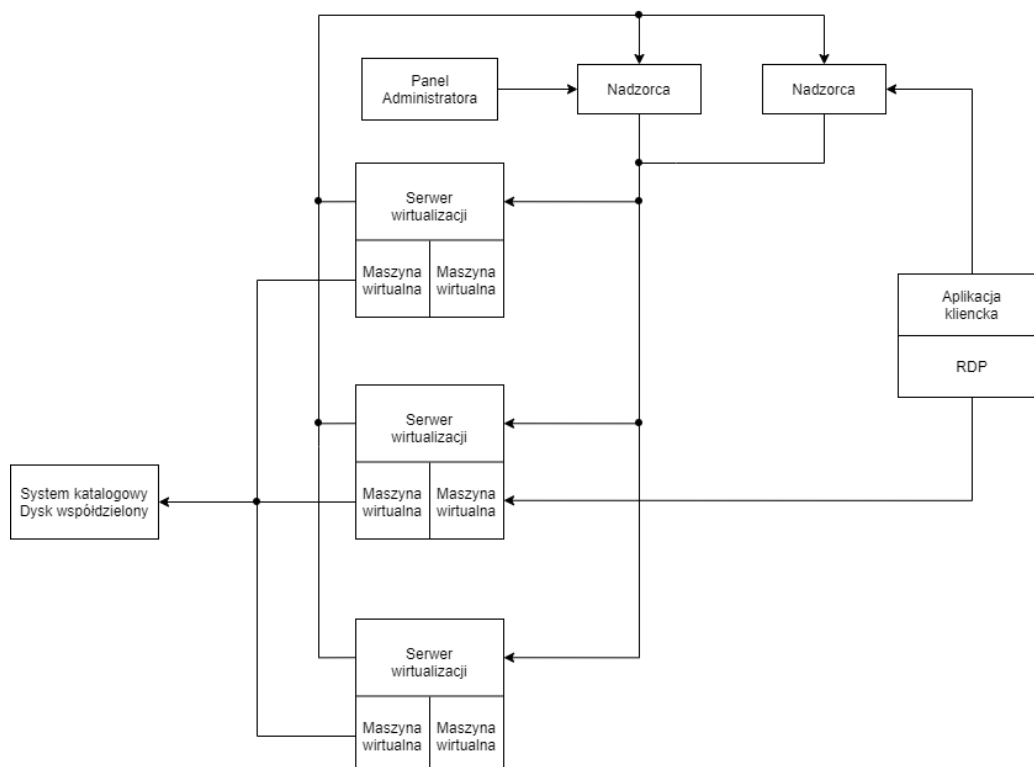
1 BLa bla bla - wymagane punkty jak z deliverable-one (MODULARNOSC!!!)

2 Architektura systemu

Przedstawiony system składa się z następujących modułów:

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

Schematyczny obraz systemu przedstawia poniższy rysunek.



Rysunek 1: Schematyczna architektura systemu

Z założenia system powinien móc skalować się w dwóch wymiarach, to znaczy:

1. Zwiększanie liczby serwerów wirtualnych - zwiększenie liczby sesji dla użytkowników.
2. Zwiększenie liczby nadzorców - zwiększenie liczby obsługiwanych klientów jednocześnie.

Szczegółowe opisy poszczególnych modułów będą omówione w następnych rozdziałach: Opis tworzonych modułów oraz Opis zewnętrznie dostarczonych modułów.

3 Opis tworzonych modułów

3.1 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 retencje i możemy zmniejszyć obciążenie poszczególnych nadzorców.

3.2 Serwer wirtualizacji

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

3.3 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.

3.4 Panel administratora

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

4 Opis zewnętrznie dostarczonych modułów

4.1 Broker wiadomości

Komunikacje wewnątrz systemu, czyli pomiędzy serwerami wirtualizacji oraz nadzorcami, będzie realizowali poprzez kolejki wiadomości. Wiadomości będą przekazywane pomiędzy modułami przez brokera, którego nie będziemy implementować. Zdecydowaliśmy się na skorzystanie z systemu RabbitMQ, który zapewni niezawodność komunikacji pomiędzy modułami. Pozostaje jednak problem hazardu oraz wyścigów, które zostaną wyeliminowane w logice komunikacji nadzorcy oraz serwera wirtualizacji.

Aby system mógł funkcjonować zdefiniowane zostaną kolejki:

- (I) Kolejka kończąca się na każdym z serwerów wirtualizacji i dla każdego z nich wiadomości są powielane. Służy ona do wysyłania próśb od nadzorców do serwerów wirtualizacji.
- (II) Kolejka kończąca się na każdym z nadzorców i dla każdego z nich wiadomości są powielane. Służy ona do wysyłania informacji do nadzorców o zmianie stanu w serwerze wirtualizacji. Dodatkowo może służyć do wymiany danych pomiędzy nadzorcami.
- (III) Kolejka kończąca się wyłącznie na pojedynczym serwerze wirtualizacji. Liczba kolejek zgadza się z liczbą serwerów wirtualizacji aktywnych w systemie. Służą one do sprawdzania, czy serwer wirtualizacji nadal pracuje po drugiej stronie. Skorzystamy z funkcjonalności kolejek na wyłączność (Exclusive Queue¹). Każda z nich powinna mieć nazwę jaka przedstawi się serwer wirtualizacji.
- (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 utworzy odpowiednie kolejki w trakcie uruchamiania. Jedynym wymogiem prawidłowego uruchomienia komunikacji jest dostępny dla wszystkich serwerów wirtualizacji oraz nadzorców proces brokera.

4.2 Dysk sieciowy

Usługa wspólnej przestrzeni dyskowej jest potrzebna do uzyskania niezależności wyboru maszyny wirtualnej od folderu użytkownika. Każda maszyna wirtualna powinna przy starcie otrzymać adres oraz dane dostępowe do takiego dysku sieciowego. Dane zostaną dostarczone poprzez wykonanie Ansible playbooka po uruchomieniu maszyny.

¹Opis zachowania kolejek na wyłączność

W przypadku maszyn wirtualnych uruchamiających system Linux potrzebne są dane do połączenia przez protokół NFS, a przy systemie Windows dane do protokołu SAMBA. Niezależnie od protokołu dane nie mogą się różnić (struktura katalogów oraz zawartość plików).

4.3 System katalogowy

Usługa systemu katalogowego jest potrzebna do uzyskania niezależności wyboru maszyny wirtualnej od danych logowania do systemu uruchomionego na maszynie wirtualnej. Każda maszyna wirtualna powinna przy starcie otrzymać adres oraz dane dostępowe do takiego systemu katalogowego. Dane zostaną dostarczone poprzez wykonanie Ansible playbooka po uruchomieniu maszyny.


Aby system katalogowy był kompatybilny z systemami Windows oraz Linux uruchamianymi na maszynie wirtualnej skorzystamy z protokołu OpenLDAP do uzyskiwania danych o użytkownikach.

5 Komunikacja

5.1 Komunikacja użytkownika z systemem - REST API

Komunikacja aplikacji klienckiej oraz panelu administratora z systemem - nadzorcą - rozwiązana jest za pomocą REST API². Wiadomości wysyłane są za pomocą protokołu HTTPS³, który zapewnia ich szyfrowanie. W tym celu wymagane jest, aby na adres, pod którym udostępniony będzie system, wystawiony był odpowiedni certyfikat⁴, gwarantujący jego tożsamość. Podczas tworzenia systemu i testów możliwe jest użycie sztucznego, własnoręcznie podpisanego certyfikatu⁵.

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



login			^
POST	/login	Log into system	v
machines			^
GET	/machines	Get number of available machines grouped into types	v
session			^
POST	/session	Get new session of selected type	v
GET	/session/{sessionId}	Get session status	v
DELETE	/session/{sessionId}	Cancel session	v
resources			^
GET	/resources	Get servers resources	v

Rysunek 2: 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. 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 też adres, z którym należy nawiązać połączenie RDP. Ten endpoint, oraz wszystkie następne wymagają autoryzacji poprzez umieszczenie tokenu otrzymanego podczas logowania w odpowiednim nagłówku wiadomości, oraz dostępne są tylko dla użytkownika.

²Opis REST API

³Specyfikacja protokołu HTTP Over TLS

⁴Opis certyfikatu TLS/SSL

⁵Opis własnoręcznie podpisanego certyfikatu TLS/SSL

- Session - pozwala na wysłanie prośby o uzyskanie sesji, pobranie stanu sesji oraz jej anulowanie.
- Resources - udostępnia informację o zasobach działających serwerów wirtualizacji. Dostępny jedynie dla administratora.

5.2 Komunikacja wewnątrz systemu - broker wiadomości

Komunikacja wewnątrz systemu opiera się na kolejkach opisanych w Opis zewnętrznie dostarczonych 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 nadzorczy przetwarzanie "odpowiedzi" zostanie uruchomione przez zmianę modelu.
- Do monitorowania utrzymania połączenia z brokerem użyty jest wbudowany mechanizm, który umożliwia wywołanie odpowiedniej procedury, gdy moduł nie wyśle wiadomości o podtrzymaniu połączenia przez określony czas⁶. Używając tego nadzorcy wykrywają, 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.

⁶Mechanizm wykrywania aktywności konsumentów w kolejce

5.3 Informacja o działaniu klienta w systemie

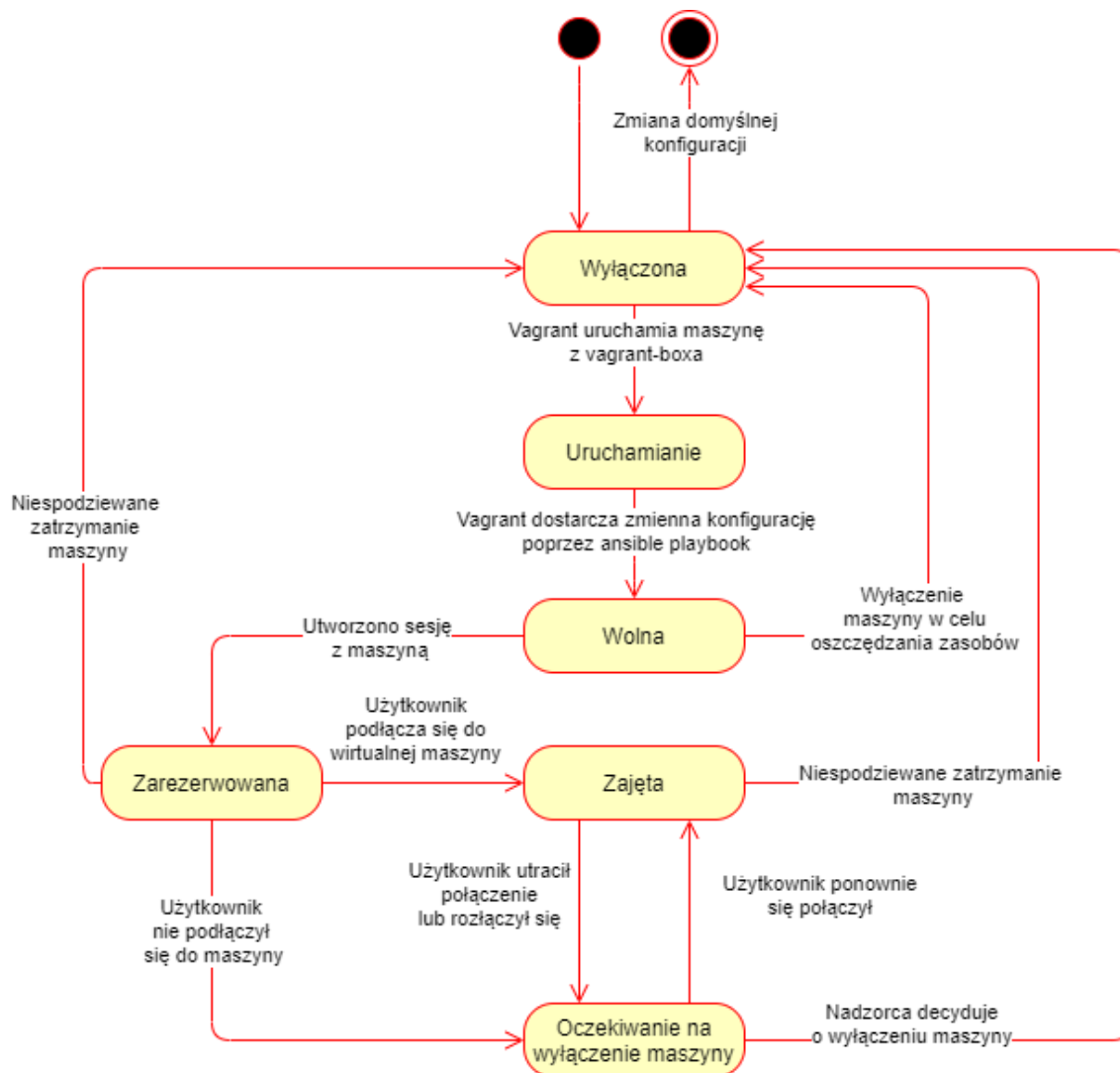
Ważną informacją, która musi posiadać system, to fakt, czy użytkownik rzeczywiście jest podłączony do maszyny wirtualnej. System uzyskuje ta informację komunikując się z jeszcze jednym brokerem wiadomości, który jest dedykowany do komunikacji z użytkownikami. Każda aplikacja kliencka po podłączeniu się 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.

6 Diagramy

6.1 Diagramy stanów

6.1.1 Maszyna wirtualna

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

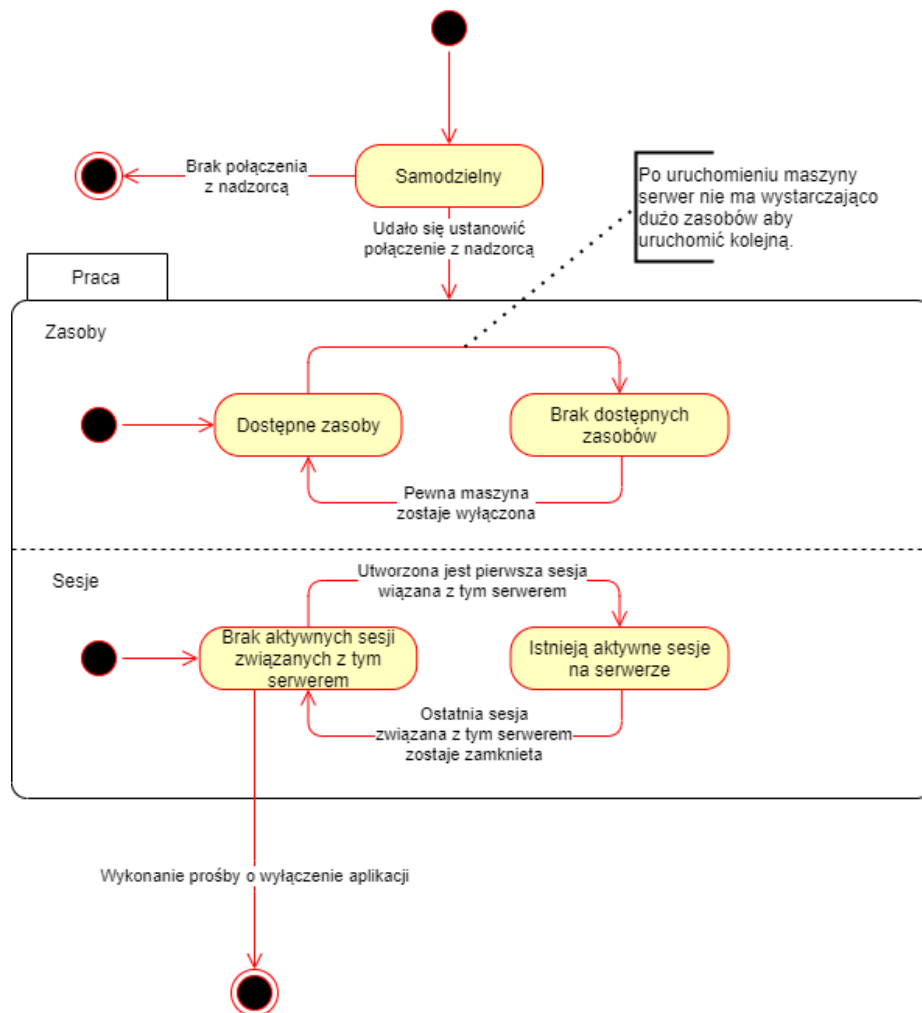


Rysunek 3: Diagram stanów dla maszyny wirtualnej

Maszyna aby być całkowicie uruchomiona musi być zaopatrzona we wszystkie konfiguracje. Stan "Wolna" oznacza możliwość przypisania sesji do tej maszyny w razie potrzeby. Po rezerwacji użytkownik nie musi od razu się zalogować (posiada pewien czas na podłączenie się do systemu). Maszyna przy oczekiwaniu na podłączenie się użytkownika (również po zerwaniu połączenia) oczekuje w stanie "Oczekiwanie na wyłączenie maszyny". Gdy użytkownik pracuje na maszynie (wiemy o tym przez monitorowanie kolejki zdefiniowanej w opisie kolejki użytkowników) wtedy jest ona w stanie "Zajęta".

6.1.2 Serwer wirtualizacji

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



Rysunek 4: Diagram stanów dla serwera wirtualizacji

Przy starcie serwer wirtualizacji oczekuje działającego nadzorcę w sieci. Jeżeli się taki nie znajdzie kończy się z błędem. Gdy jednak odnajdzie takowy rozpoczyna się praca serwera. Ze względu na zasoby może on mieć wolne zasoby aby utworzyć nowe maszyny, lub też nie. Jednak ważniejszym stanem z perspektywy działania serwera są połączeni do niego użytkownicy. W przypadku gdy podłączony jest do niego przynajmniej jeden użytkownik nie może się zakończyć jego praca. Jeżeli system ma wyłączyć się prawidłowo ostatni użytkownik musi rozłączyć się z używaną maszyną wirtualną.

6.1.3 Użytkownik



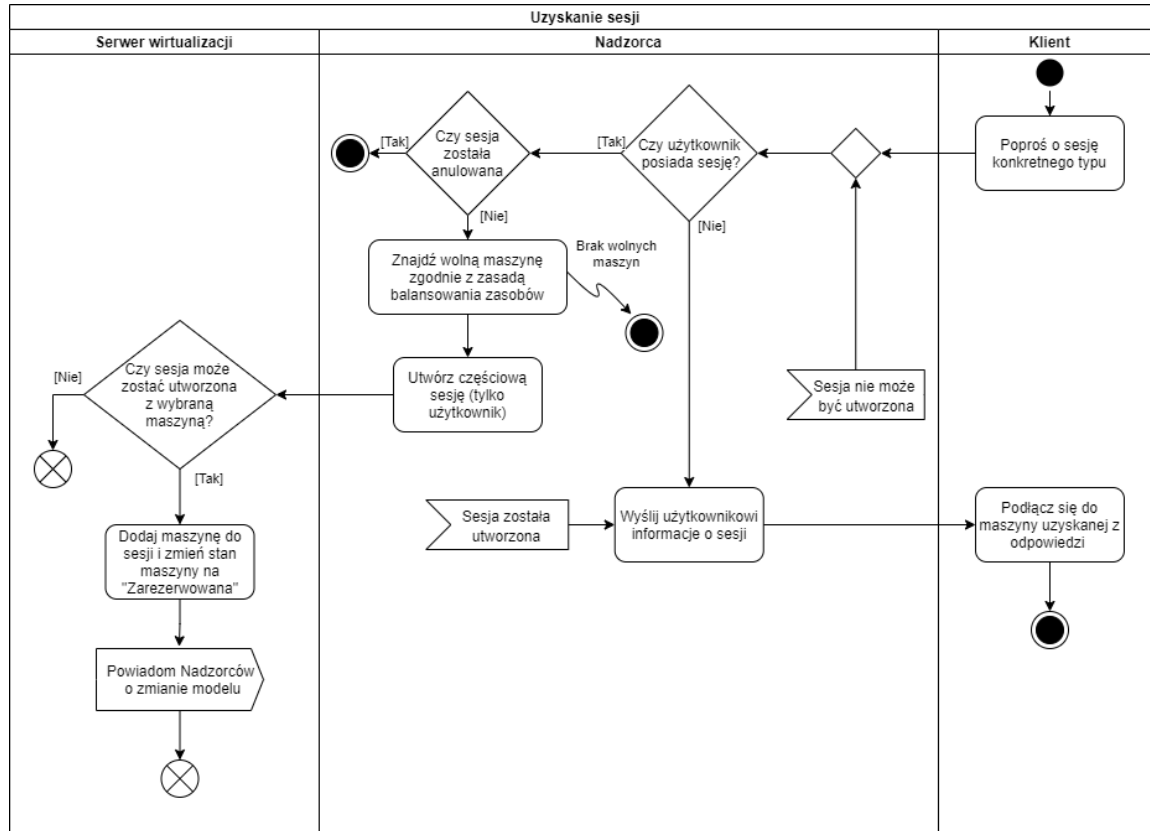
Rysunek 5: Diagram stanów dla użytkownika

Użytkownik z perspektywy systemu po zalogowaniu może być 2 dwóch stanach: pracuje w ramach swojej sesji lub też nie. W stanie "Połączony" aplikacja na której pracuje ma obowiązek powiadamiać serwer wirtualizacji, że ciągle jest obecny. Przy zmianie stanu do innego informowanie musi ustać.

6.2 Diagramy aktywności

6.2.1 Uzyskanie sesji

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



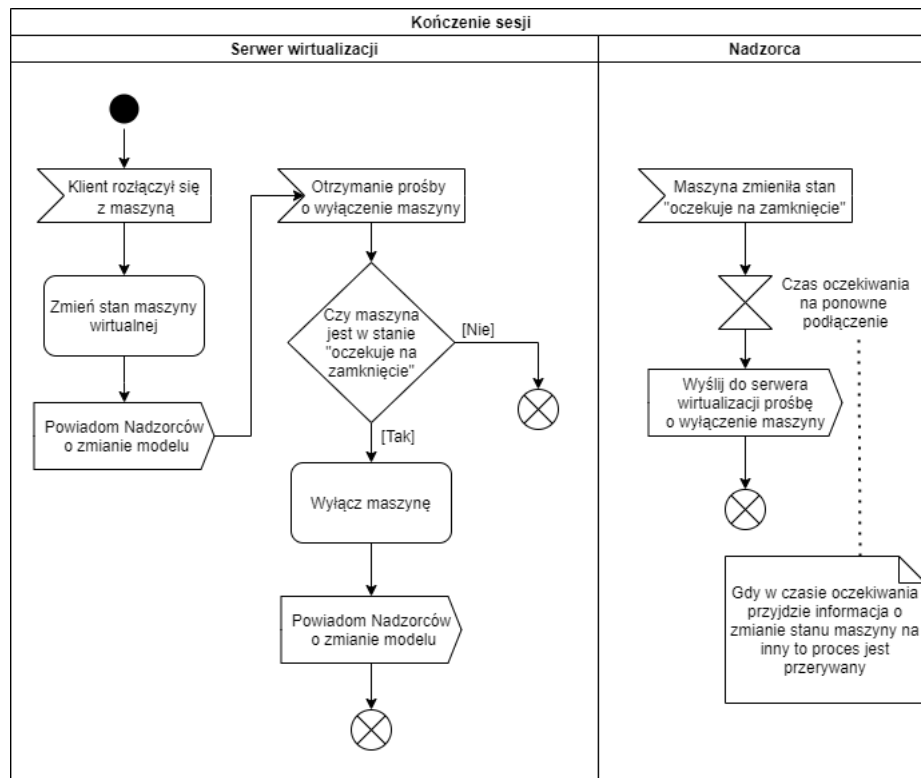
Rysunek 6: Diagram aktywności dla uzyskania sesji

W przypadku istniejącej już sesji nadzorca przekazuje od razu cały gotowy obiekt sesji. W przeciwnym razie nadzorca zaglądając do modelu systemu znajduje pewną wolną maszynę (dla konkretnego stanu modelu za każdym razem musi być wybrana ta sama maszyna - powtarzalność) i informuje serwer wirtualizacji o prośbie utworzenia sesji. Gdy nie znajdzie wolnej maszyny, to zgłasza błąd do użytkownika. Taka sytuacja nie powinna się wydarzyć, ponieważ system zawsze powinien trzymać jakiś zapas wolnych maszyn. Jeżeli jednak nie ma żadnej wolnej maszyny oznacza to pewien deficyt zasobów i jest to nietypowa sytuacja. Może się zdarzyć, że model jest nieaktualny oraz nie można utworzyć sesji z wcześniej wybraną maszyną. Wtedy prośba zostaje odrzucona, ale zmiana modelu powinna zaraz nadejść (co zapoczątkuje powtórzenie próby utworzenia sesji).

Dodatkowo uzyskanie sesji przez użytkownika będzie zrealizowane asynchronicznie. Użytkownik oddzielnym zapytaniem będzie prosić o uzyskanie sesji, a innym będzie prosić o dane swojej sesji. Obiekt jest w pełni utworzony, gdy odpowiedź nadzorca zawiera identyfikator sesji oraz adres maszyny do połączenia. Szczegóły znajdują się w opisie REST API.

6.2.2 Kończenie sesji

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

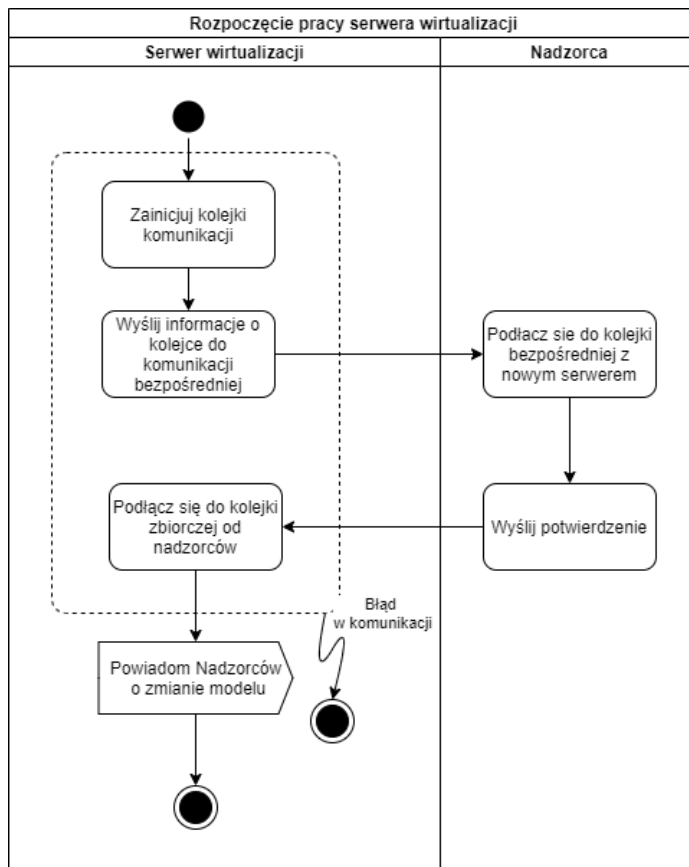


Rysunek 7: Diagram aktywności dla zakończenia sesji

Rozpoczyna się w momencie, gdy użytkownik odłączy się od systemu lub utraci połączenie oraz minie pewien ustalony czas bez podłączenia się ponownie przez użytkownika. Ważnym jest aby o utracie połączenia lub rozłączeniu informuje serwer wirtualizacji poprzez zmianę modelu. Decyzję o wyłączeniu maszyny podejmuje jednak nadzorca. Powoduje to, że zmiana konfiguracji nadzorców będzie oznaczać spójną reakcję całego systemu. Dodatkowo umożliwia to w perspektywie czasu utworzenie bardziej złożonego algorytmu zarządzania zasobami.

6.2.3 Rozpoczęcie pracy serwera wirtualizacji

Proces opisuje przyjęcie nowego serwera wirtualizacji do systemu.



Rysunek 8: Diagram aktywności dla rozpoczęcia pracy serwera wirtualizacji

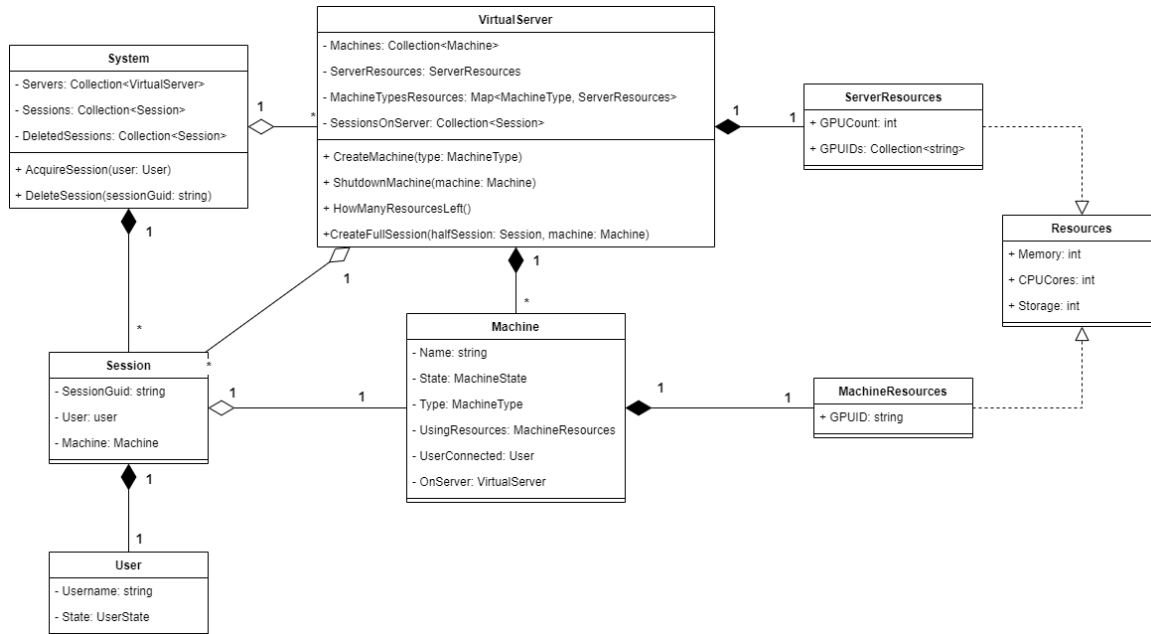
Serwer wirtualizacji bez działającego nadzorcy nie będzie w stanie obsługiwać użytkowników. Oznacza to, że jeśli przy starcie wystarczająco wiele razy nie wykryje brokera wiadomości lub nadzorcy po drugiej stronie kolejek⁷ to się wyłączy. Jeżeli jednak nadzorca będzie po drugiej stronie to serwer podłączy się do kolejek wspólnych kolejek oraz przekaże informacje nadzorcom o kolejce bezpośredniej. Gdy komunikacja będzie ustanowiona bezwarunkowo wyśle stan swojego modelu do nadzorców.

⁷Nadzorcy po drugiej stronie wspólnej kolejki mogą zostać wykryci poprzez mechanizm potwierdzenia wykonania zadania

6.3 Diagramy klas

6.3.1 Model systemu

Aby nadzorca wiedział jak zarządzać systemem potrzebuje przechowywać model systemu. Poniższa struktura danych przedstawia aktualny stan całego systemu.



Rysunek 9: Diagram klas dla modelu systemu

W skład tego modelu wchodzi informacje o:

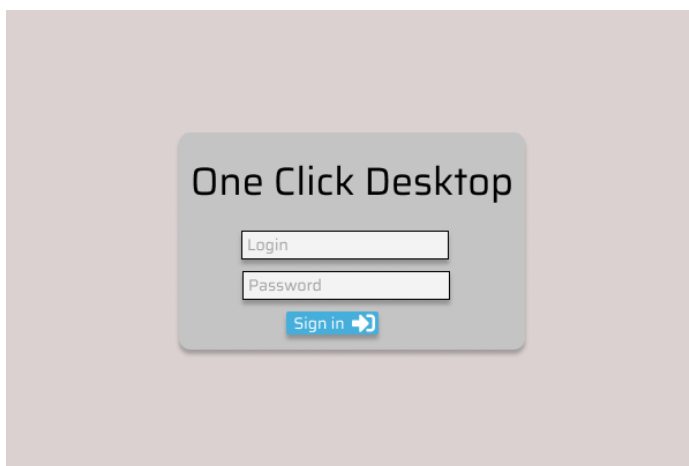
1. Dostępnych serwerach wirtualizacji.
2. Dostępnych zasobach na serwerach.
3. Aktualnie działających maszynach, ich typach oraz zajmowanych przez nie zasobach.
4. Aktualnie trwających sesjach z użytkownikami.

6.4 Diagramy sekwencji

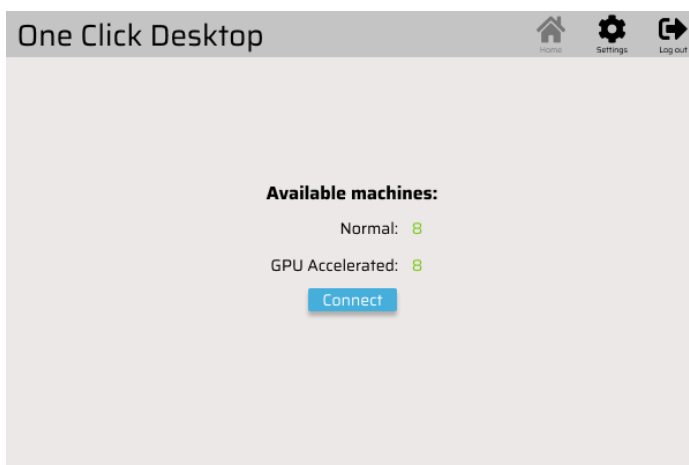
7 Interfejs użytkownika

7.1 Aplikacja kliencka

Aplikacja kliencka posiada interfejs użytkownika pozwalający na zalogowanie się oraz nawiązanie połączenia ze zdalną sesją. Użytkownikowi wyświetlany jest czynność, która aktualnie się odbywa, oraz w każdym momencie może zakończyć sesję.

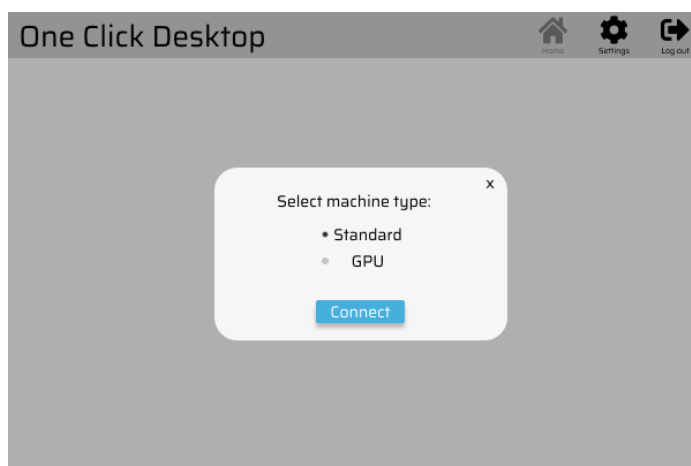


Rysunek 10: Ekran logowania



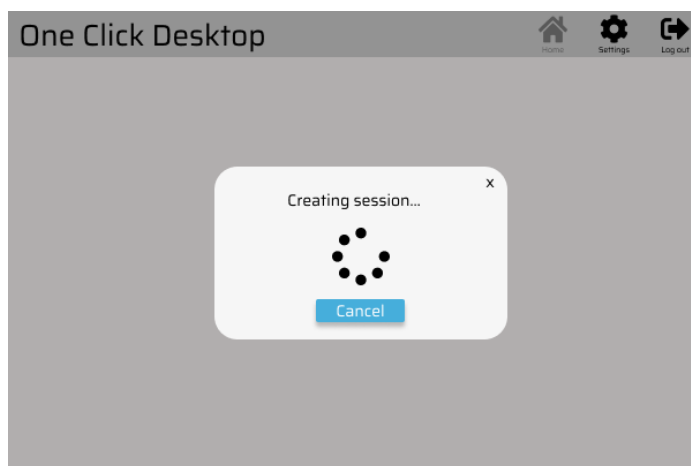
Rysunek 11: Główny widok zawierający dostępność maszyn

Na tym ekranie użytkownik może rozpocząć proces uzyskiwania sesji, przejść do ekranu ustawień lub wylogować się. Jeżeli w systemie nie ma dostępnych maszyn, lub nie uda się uzyskać informacji o ich dostępności, to przycisk połączenia jest niedostępny. Wciśnięcie tego przycisku prowadzi do kolejnego ekranu.

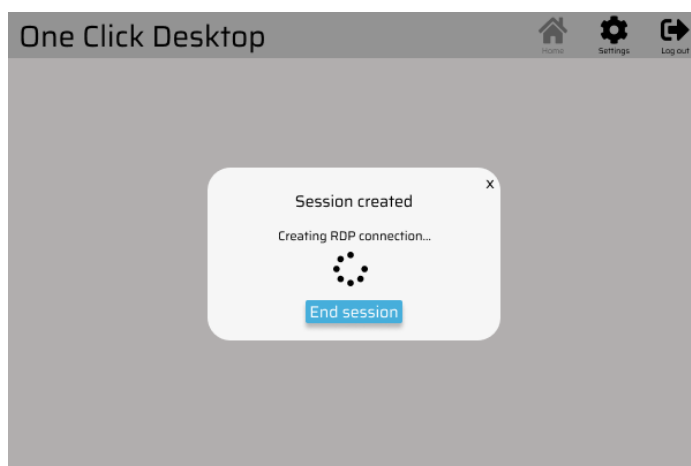


Rysunek 12: Wybór typu sesji

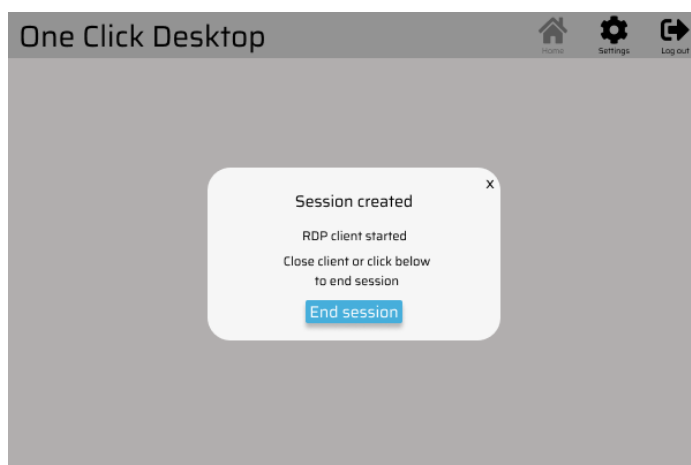
Do wyboru dostępne są jedynie typy sesji, które system określi jako dostępne. Wybranie typu sesji kliknięcie przycisku prowadzi do kolejnego ekranu. Następne ekrany przechodzą automatycznie do kolejnych bez interwencji użytkownika, aż do informacji o nawiązaniu połączenia lub błęd.



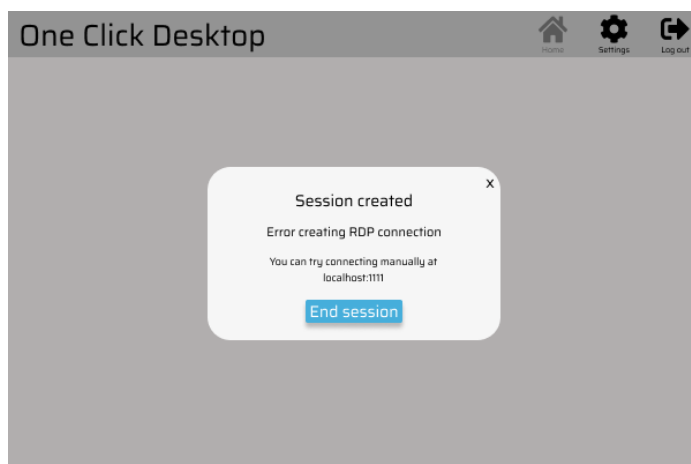
Rysunek 13: Tworzenie sesji



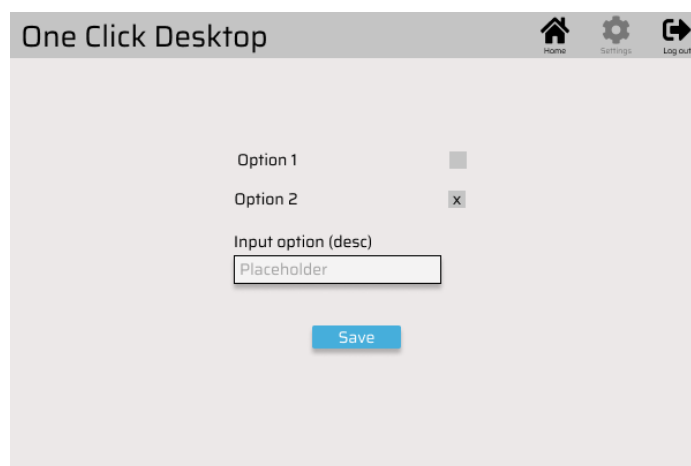
Rysunek 14: Nawiązywanie połączenia RDP



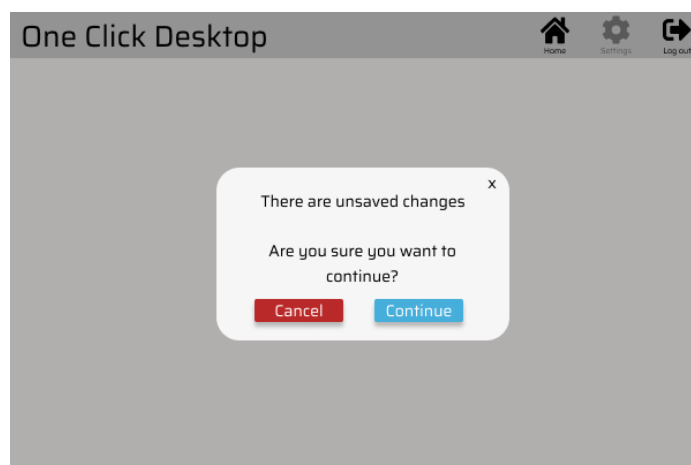
Rysunek 15: Połączenie nawiązane



Rysunek 16: Błąd przy nawiązywaniu połączenia



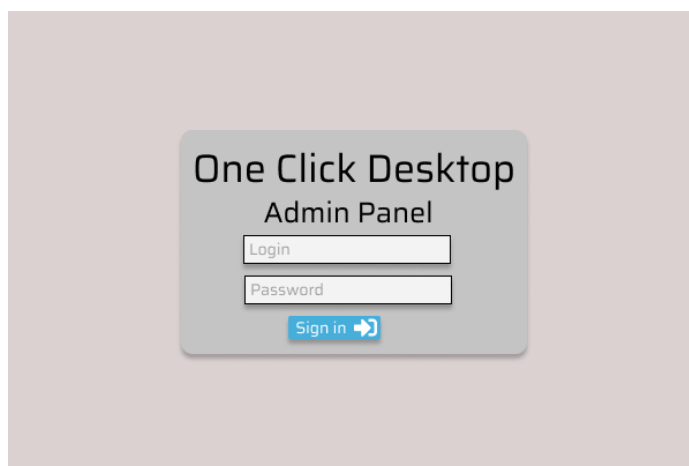
Rysunek 17: Ustawienia



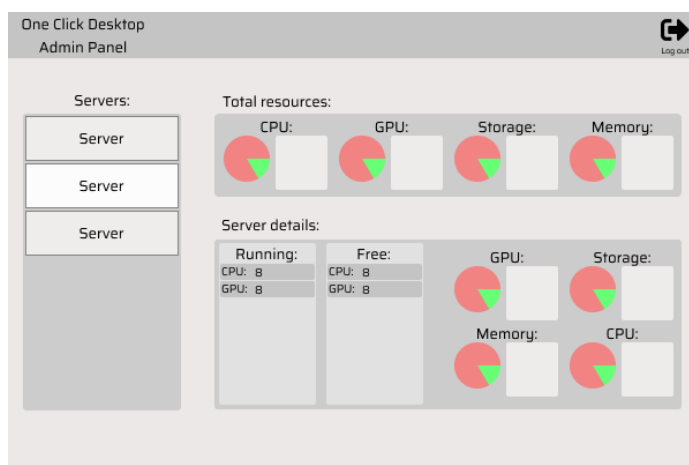
Rysunek 18: Powiadomienie przy wyjściu z ekranu ustawień z niezapisanymi zmianami

7.2 Panel administratora

Panel administratora posiada skromny interfejs umożliwiający zalogowanie się oraz podgląd zużycia zasobów.



Rysunek 19: Ekran logowania



Rysunek 20: Widok zużycia zasobów

Na tym widoku możemy zobaczyć zużycie zasobów globalne oraz dla każdego z serwerów wirtualizacji. Dodatkowo dla serwera wyświetlona jest również ilość działających i możliwych do uruchomienia maszyn każdego z typów.

8 Zewnętrzne narzędzia

8.1 Ansible

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

1. Dane dostępowe do dysku sieciowego oraz wykorzystany protokół
2. Dane dostępowe do usługi katalogowej
3. TODO: dopisać wszystkie potrzebne konfiguracje

Playbook można rozszerzać o potrzebne dane zależne od użycia.

8.2 Vagrant

Vagrant zostanie wykorzystany w celu łatwej parametryzacji oraz powtarzalnego tworzenia maszyn wirtualnych z przygotowanego wcześniej obrazu systemu. Głównie wykorzystany będzie mechanizm Vagrantboxów, 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. TODO: dopisać wszystkie potrzebne wymagania

8.3 Libvirt z QEMU

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

1. Tworzenie maszyn wirtualnych.
2. Uruchamianie maszyn wirtualnych.
3. Przyporządkowanie zasobów maszynom wirtualnym (w tym kraty graficzne).
4. Wyłączanie maszyn wirtualnych.
5. Sprawdzanie, czy maszyna działa na serwerze wirtualizacji.

9 Wybrana technologia

- Aplikacja kliencka
 - Typescript⁸ /Javascript⁹
 - Node.js¹⁰ - środowisko uruchomieniowe używane do integracji z systemem użytkownika
 - Angular¹¹ - renderowanie widoków
 - Electron¹² - platforma programistyczna
 - Jest¹³ - testy jednostkowe
 - Cypress¹⁴ - testy integracyjne
- Panel administratora
 - Typescript/Javascript
 - Angular - platforma aplikacji WWW
 - Jest - testy jednostkowe
 - Cypress - testy integracyjne
- Nadzorca i serwer wirtualizacji
 - C#¹⁵
 - RabbitMQ¹⁶ - broker asynchronicznych wiadomości
 - Ansible¹⁷ - konfigurowanie maszyn wirtualnych
 - Vagrant¹⁸ - tworzenie obrazów maszyn wirtualnych oraz ich uruchamianie
 - libvirt¹⁹ - zarządzanie maszynami wirtualnymi
 - OpenLDAP²⁰ - dostępu do systemu katalogowego
 - NFS²¹ - dostęp do katalogów domowych z maszyny wirtualnej
 - Arch Linux²² - system operacyjny uruchamiany przez maszyny wirtualne

⁸Strona projektu Typescript

⁹Obecny standard języka Javascript

¹⁰Strona projektu Node.js

¹¹Strona projektu Angular

¹²Strona projektu Electron

¹³Strona projektu Jest

¹⁴Strona projektu Cypress

¹⁵Dokumentacja języka C#

¹⁶Strona projektu RabbitMQ

¹⁷Strona projektu Ansible

¹⁸Strona projektu Vagrant

¹⁹Strona projektu libvirt

²⁰Strona projektu OpenLDAP

²¹Opis na stronie firmy Microsoft

²²Strona systemu operacyjnego Arch Linux

- GNU/Linux - wspierany system operacyjny
- Różne
 - Swagger Codegen²³ - automatyczna generacja API na podstawie specyfikacji
 - RDP²⁴ - łączenie ze zdalnymi sesjami

²³Opis narzędzia na stronie firmy Swagger

²⁴Dokumentacja protokołu RDP od Microsoft