07.11.2021r.

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

Architektura i opis systemu

Autorzy: Krzysztof Smogór, Piotr Widomski
Promotor: Dr inż. Marek Kozłowski
Wersja 1.2

Streszczenie

Dokument ma za zadanie zapoznać czytelnika z szczegółowym opisem systemu. Wpierw opisane są tworzone oraz zewnętrznie dostarczone moduły. Następnie opisane zostały metody komunikacji między modułami. Dalej, za pomocą diagramów, opisane zostały zależności i współdziałanie modułów. Przedstawione zostały modele interfejsu użytkownika dla aplikacji klienckiej oraz panelu administratora. Kolejno opisane zostały używane narzędzia oraz technologie. Na końcu dokumentu znajduje się lista załączników.

Historia zmian

1.0	05.11.2021r.	Pierwsza wersja.
1.1	07.11.2021r.	Opis modułów, komunikacja, interfejs użytkownika oraz technologie
1.2	05.11.2021r..	Diagramy, zewnętrzne narzędzia oraz wstęp

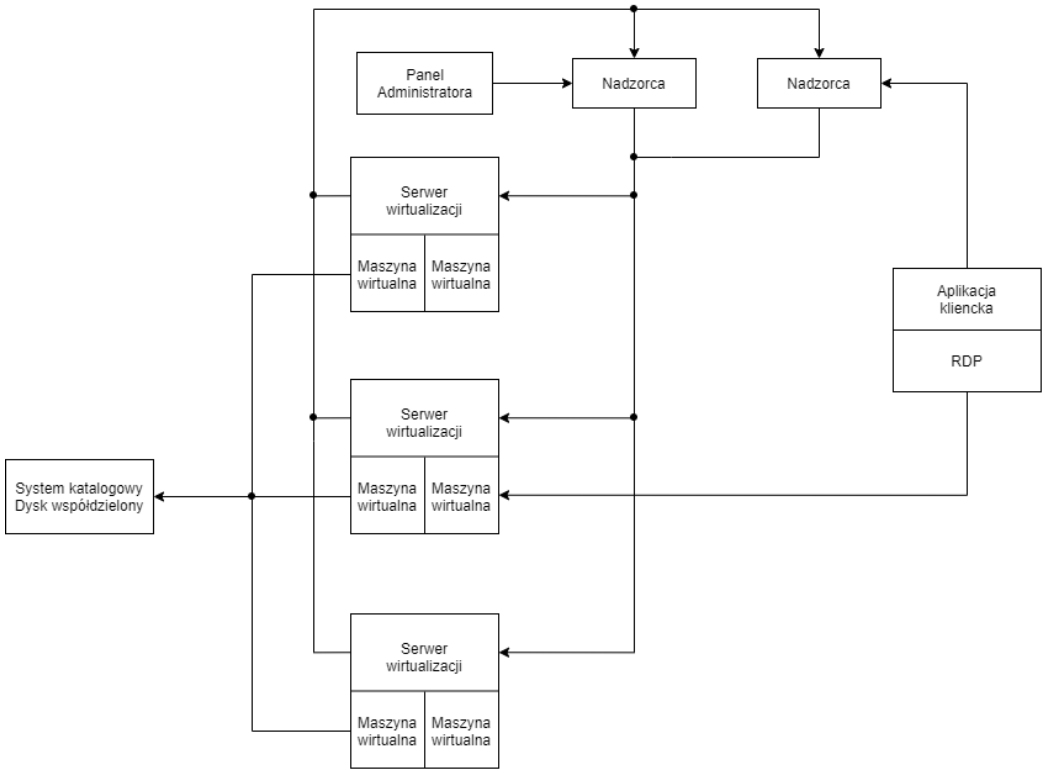
Spis treści

1 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.

2 Opis tworzonych modułów

2.1 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.

2.2 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.

2.3 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.

2.4 Panel administratora

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

3 Opis zewnętrznie dostarczonych modułów

3.1 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 sa 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 Queue[1]). Każda z nich powinna mieć nazwę jaka przedstawi się serwer wirtualizacji.

Powyższe 3 grupy kolejek umożliwią prawidłowe działanie systemu. Każdy z modułów utworzy odpowiednie kolejki w trakcie uruchamiania. Jedynym wymogiem prawidłowego uruchomienia komunikacji jest dostępny przez wszystkie serwery wirtualizacji oraz nadzorców proces brokera.

3.2 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.

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).

3.3 System katalogowy

Usługa systemu katalogowego jest potrzebna do uzyskania niezależności wyboru maszyny wirtualnej od danych logowania do systemu uruchomionego na maszynie wir-

[1]Opis zachowania kolejek na wyłączność

tualnej. 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.

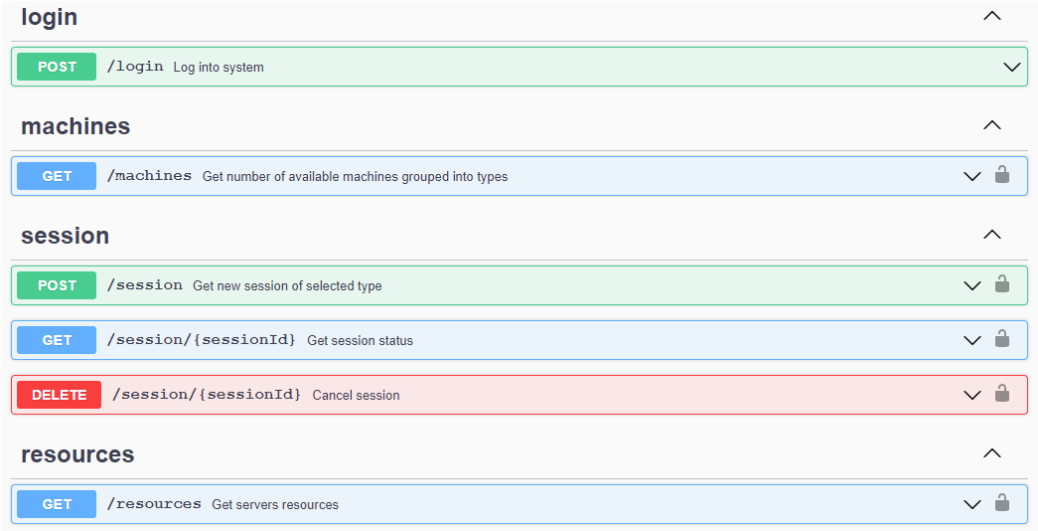
Aby system katalogowy był kompatybilny z systemami Windows oraz Linux uruchamianymi na maszynie wirtualnej skorzystamy z protokołu OpenLDAP do uzyskiwania danych o użytkownikach.

4 Komunikacja

4.1 Komunikacja użytkownika z systemem - REST API

Komunikacja aplikacji klienckiej oraz panelu administratora z systemem - nadzorcą - rozwiązana jest za pomocą REST API[2]. Wiadomości wysyłane są za pomocą protokołu HTTPS[3], który zapewnia ich szyfrowanie. W tym celu wymagane jest, aby na adres, pod którym udostępniony będzie system, wystawiony był odpowiedni certyfikat[4], gwarantujący jego tożsamość. Podczas tworzenia systemu i testów możliwe jest użycie sztucznego, własnoręcznie podpisanego certyfikatu[5].

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



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.

[2]Opis REST API
[3]Specyfikacja protokołu HTTP Over TLS
[4]Opis certyfikatu TLS/SSL
[5]Opis 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.

4.2 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 procesowane 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 "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 czas. 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.

5 Diagramy

5.1 Diagramy stanów

5.2 Diagramy aktywności

5.3 Diagramy klas

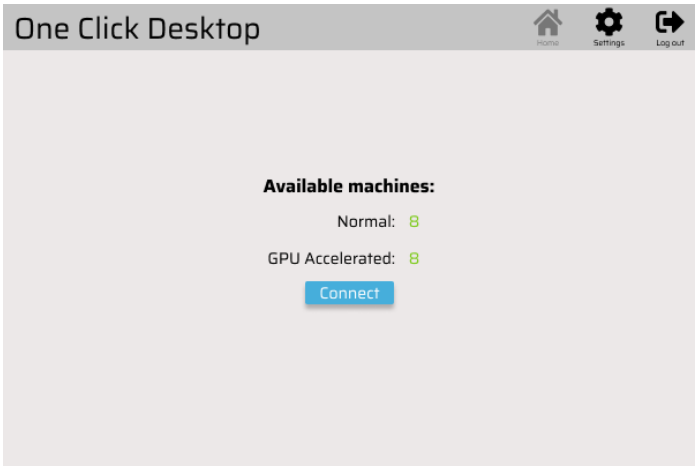
5.4 Diagramy sekwencji

6 Interfejs użytkownika

6.1 Aplikacja kliencka

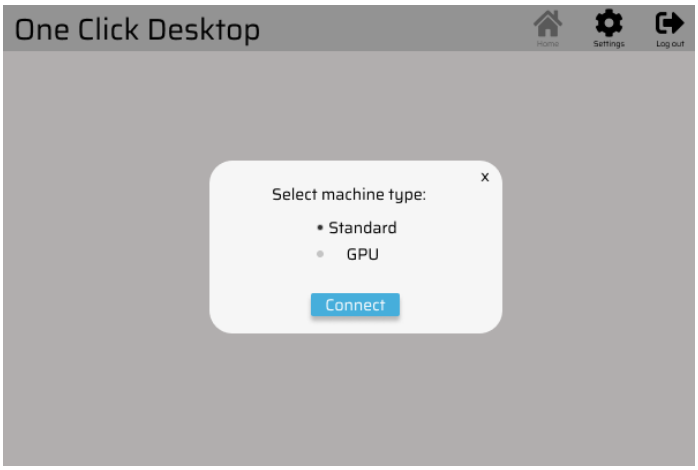
Aplikacja kliencka posiada interfejs użytkownika pozwalający na zalogowanie się oraz nawiązanie połączenia ze zdalną sesją. Użytkownikowi wyświetlany jest czynność, która aktualnie się odbywa, oraz w każdym momencie może zakończyć sesję.

Rysunek 3: Ekran logowania



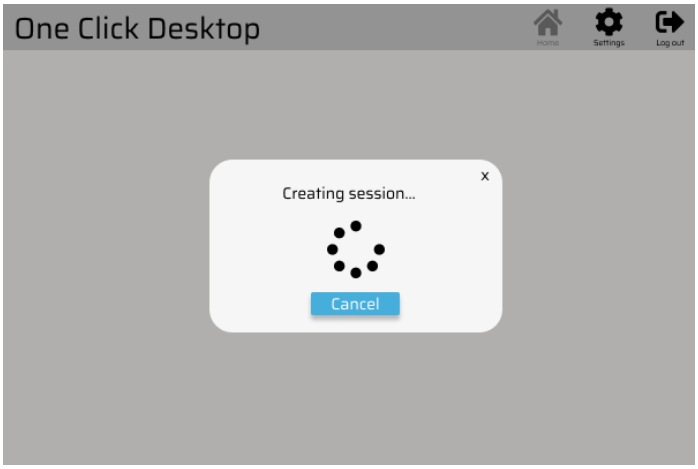
Rysunek 4: Główny widok zawierający dostępność maszyn

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.

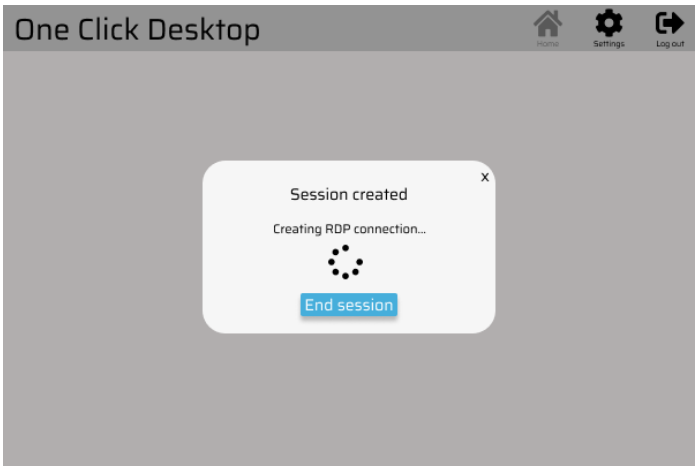


Rysunek 5: Wybór typu sesji

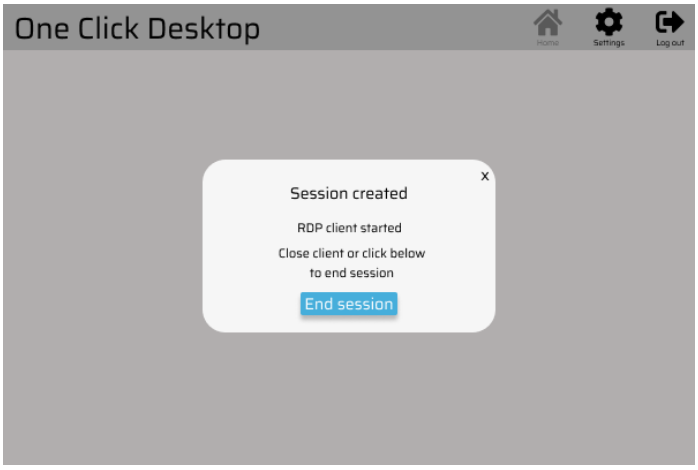
Do wyboru dostępne są jedynie typy sesji, które system określi jako dostępne. Wybranie typu sesji kliknięcie przycisku prowadzi do kolejnego ekranu. Następne ekrany przechodzą automatycznie do kolejnych bez interwencji użytkownika, aż do informacji o nawiązaniu połączenia lub błędu.



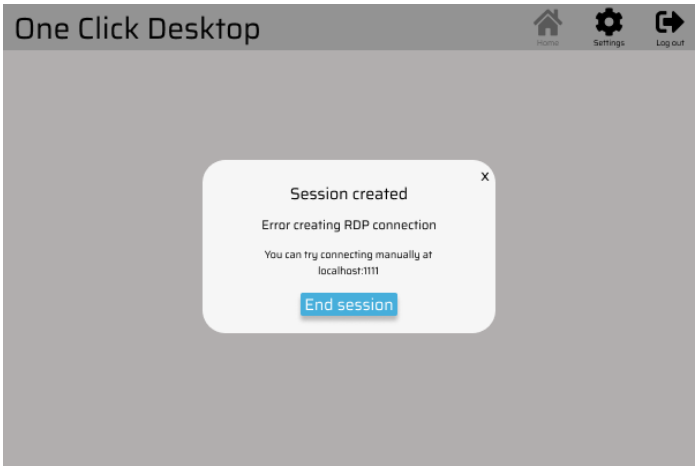
Rysunek 6: Tworzenie sesji



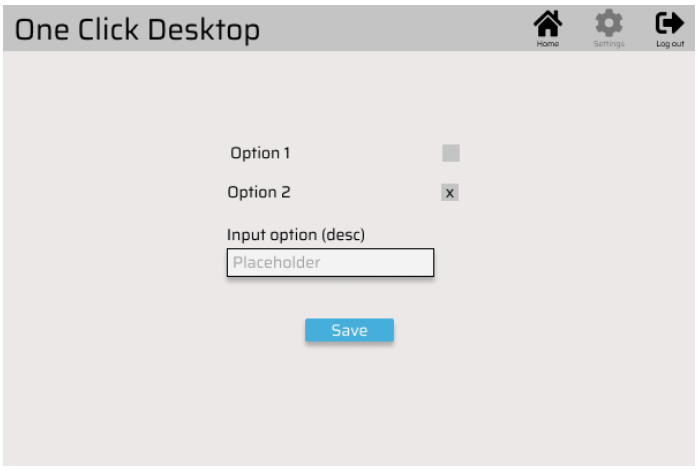
Rysunek 7: Nawiązywanie połączenia RDP



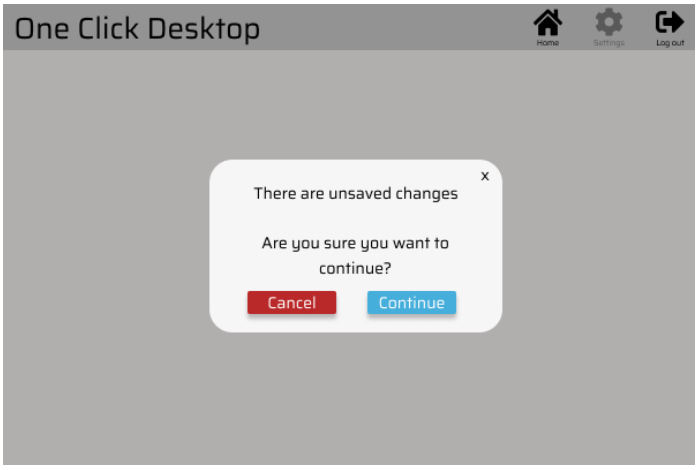
Rysunek 8: Połączenie nawiązane



Rysunek 9: Błąd przy nawiązywaniu połączenia



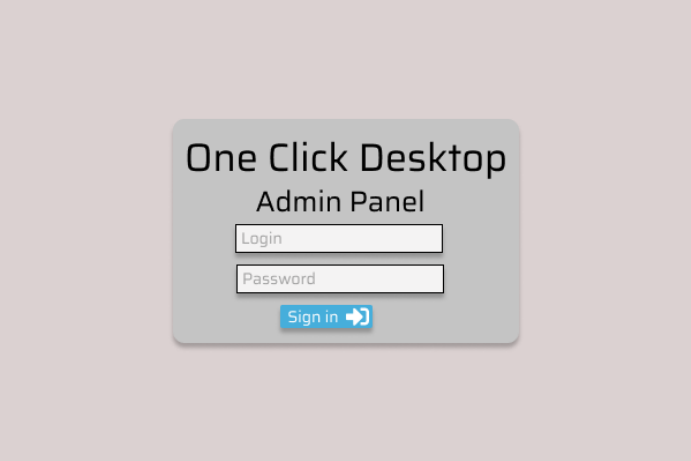
Rysunek 10: Ustawienia



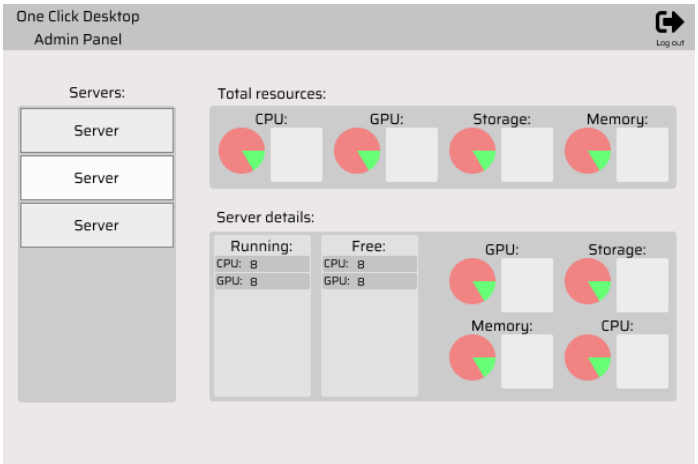
Rysunek 11: Powiadomienie przy wyjściu z ekranu ustawień z niezapisanymi zmianami

6.2 Panel administratora

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



Rysunek 12: Ekran logowania



Rysunek 13: 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.

7 Zewnętrzne narzędzia

7.1 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: dopisac wsyztskie potrzebne konfiguracje

Playbook można rozszerzać o potrzebne dane zależne od użycia.

7.2 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: dopisac wsyztskie potrzebne wymogi

7.3 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.

8 Wybrana technologia

- Aplikacja kliencka
  - Typescript[6] /Javascript[7]
  - Node.js[8] - środowisko uruchomieniowe używane do integracji z systemem użytkownika
  - Angular[9] - renderowanie widoków
  - Electron[10] - platforma programistyczna
  - Jest[11] - testy jednostkowe
  - Cypress[12] - testy integracyjne
- Panel administratora
  - Typescript/Javascript
  - Angular - platforma aplikacji WWW
  - Jest - testy jednostkowe
  - Cypress - testy integracyjne
- Nadzorca i serwer wirtualizacji
  - C#[13]
  - RabbitMQ[14] - broker asynchronicznych wiadomości
  - Ansible[15] - zarządzanie maszynami wirtualnymi
  - Vagrant[16] - tworzenie obrazów maszyn wirtualnych
  - libvirt[17] - uruchamianie maszyn wirtualnych
  - OpenLDAP[18] - dostępu do systemu katalogowego
  - NFS[19] - dostęp do katalogów domowych z maszyny wirtualnej
  - Arch Linux[20] - system operacyjny uruchamiany przez maszyny wirtualne
  - GNU/Linux - wspierany system operacyjny
- Różne
  - Swagger Codegen[21] - automatyczna generacja API na podstawie specyfikacji

[6]Strona projektu Typescript
[7]Obecny standard języka Javascript
[8]Strona projektu Node.js
[9]Strona projektu Angular
[10]Strona projektu Electron
[11]Strona projektu Jest
[12]Strona projektu Cypress
[13]Dokumentacja języka C#
[14]Strona projektu RabbitMQ
[15]Strona projektu Ansible
[16]Strona projektu Vagrant
[17]Strona projektu libvirt
[18]Strona projektu libvirt
[19]Opis na stronie firmy Microfost
[20]Strona systemu operacyjnego Arch Linux
[21]Opis narzędzia na stronie firmy Swagger

– RDP[22] - łączenie ze zdalnymi sesjami

[22]Dokumnetacja protokołu RDP od Microsoft

9 Załączniki

1. `REST_API_Overseer.yaml` - dokładna specyfikacja API w standardzie OpenAPI 3.0.3[23]

[23]Specyfikacja standardu OpenAPI w wersji 3.0.3