0.1. Końcowy stan systemu

System, w swojej ostatecznej postaci, spełnia wszystkie postawione mu wymagania. Mimo trudności w implementacji niektórych funkcjonalności, w szczególności zarządzania maszynami wirtualnymi z poziomu kodu `C#`, ostateczny projekt działa zgodnie z oczekiwaniami. Projekt tworzony w ramach tej pracy uznajemy zatem za udany.

Elementem, który mógł zostać lepiej wykonany jest zarządzanie maszynami wirtualnymi przy pomocy Vagranta. Z powodu braku interfejsu programistycznego udostępnianego przez ten program, wykorzystywany jest on w naszym systemie za pośrednictwem wywołań poleceń powłoki. Nie jest to najlepsze rozwiązanie, o czym świadczy liczba błędów, które wynikły podczas tworzenia systemu. Jest to w naszym przekonaniu najbardziej podatny na błędy element systemu. Taki sposób rozwiązania jest spowodowany chęcią skorzystania z Vagranta podczas projektowania systemu. Na tamtym etapie nie sprawdziliśmy, czy dostępne są narzędzia umożliwiające korzystanie z niego z poziomu kodu. W momencie odkrycia braku takich narzędzi, było już za późno na zastąpienie Vagranta innym rozwiązaniem.

Projektując komunikację wewnętrzną za pośrednictwem RabbitMQ postawiliśmy na wykorzystanie po jednej kolejce na odbieranie wiadomości bezpośrednich oraz zbiorczych. W połączeniu z faktem, że przesyłane wiadomości posiadają różną formę, doprowadziło to do potrzeby deserializacji wiadomości na różne typy obiektów, w zależności od typu otrzymanej wiadomości. Naszym zdaniem problem ten nie został przez nas rozwiązany zadowalająco i mógłby zostać wykonany lepiej.

Z pominięciem wyżej wymienionych aspektów, uważamy projekt komunikacji i procesów biznesowych za dobrze wykonany, w sposób umożliwiający łatwą implementację. Dzięki głęboko przemyślanym rozwiązaniom w tych płaszczyznach, ich implementacja była mało problematyczna. Podczas testów nie wynikły również również żadne problemy, które wymagałyby modyfikacji ustalonych procesów i sekwencji.

0.2. Możliwe ścieżki rozwoju

0.2.1. Panel administratora

Funkcjonalność panelu administratora jest obecnie ograniczona. Pozwala on jedynie na podejrzenie stanu zasobów w systemie. Informacja ta pozwala ocenić, czy potrzebne jest uruchomienie nowych instancji serwera wirtualizacji. Jest to jednak jedyna dostępna informacja.

Możliwym rozwinięciem jest udostępnienie wglądu w cały model systemu przechowywany przez nadzorców. Umożliwi to ocenę stopnia zajętości maszyn każdego typu oraz ilości użytkowników systemu. Dodatkowo wgląd do stanu konkretnych maszyn i sesji może ułatwić rozwiązywanie problemów.

0.2.2. Użycie konkretnych kart graficznych

W konfiguracji typu maszyny wirtualnej możliwe jest jedynie wskazanie, czy ma ona posiadać kartę graficzną na wyłączność. Oznacza to, że otrzymana karta graficzna nie jest sprecyzowana i może być dowolną ze skonfigurowanych w serwerze wirtualizacji.

W celu uniknięcia niespodzianek, co do modelu otrzymanej karty graficznej, preferowane byłaby możliwość sprecyzowania konkretnego modelu karty graficznej przekazywanej do maszyny danego typu. Możliwym sposobem osiągnięcia tej funkcjonalności jest rozszerzenie konfiguracji serwera wirtualizacji o możliwość nadania identyfikatora przekazywanym kartom graficznym.

0.3. Otwarte problemy

0.3.1. Konfiguracja połączenia RabbitMQ

Klient RabbitMQ używany przez moduły wewnętrzne do połączenia z instancją brokera udostępnia wiele opcji konfiguracji, które nie są udostępniane przez moduły. Konfiguracja ta definiuje opcje połączenia, takie jak hasło autoryzacji, czy niestandardowe ścieżki połączenia. W przypadku korzystania z niestandardowej instancji brokera, konfiguracja ta może okazać się niezbędna. Z tego powodu powinno być możliwe przekazanie ustawień połączenia z brokerem wiadomości w plikach konfiguracyjnych modułów wewnętrznych.

0.3.2. Monitorowanie stanu połączenia klienta

Do monitorowania stanu połączenia z maszyną wirtualną wykorzystywany jest broker wiadomości RabbitMQ. Klient uznawany jest za połączonego gdy istnieje kolejka z nazwą odpowiadającą identyfikatorowi sesji. Rozwiązanie to jest bardzo niestandardowe, kolejki używane są w nie zamierzony sposób. Dodatkowo klient RabbitMQ nie udostępnia możliwości powiadamiania o braku kolejki w sposób pasywny. Implementacja ta wymaga działania instancji brokera, która jest dostępna spoza wnętrza systemu. Może to powodować luki w bezpieczeństwie. Pożądane byłoby zastąpienie tej implementacji mechanizmem przeznaczonym do tego typu rozwiązań.

0.3.3. Znane problemy wyścigów

Pomimo przywiązania dużej uwagi do zaprojektowania komunikacji i procesów w sposób niwelujący ilość możliwych problemów wynikających z konkurencji, wciąż istnieją miejsca, w których mogą one wystąpić. Znanym problemem jest możliwość uruchomienia dwa razy więcej maszyn niż zamierzono podczas startu systemu. Spowodowane jest to tym, że nadzorca otrzymując informację o dostępności serwera wirtualizacji poprosi go o utworzenie wymaganych maszyn wirtualnych. Jeżeli zanim serwer prześle informację o wykonaniu zadania, drugi serwer wirtualizacji również stanie się dostępny, nadzorca poprosi go o utworzenie tych samych maszyn. Taki ciąg wydarzeń doprowadzi do działania dwa razy większej liczby maszyn niż było zamierzone.

Możliwym rozwiązaniem problemu może być przeprojektowanie procesu uruchamiania wymaganych maszyn, w celu uwzględnienia możliwości pojawienia się kolejnych instancji serwera wirtualizacji.