## Pytania i kwestie do rozstrzygnięcia

1. Proszę opisać lub jeżeli jest to możliwe przedstawić architekturę proponowanego rozwiązania.

   W tworzonych i rozwijanych przez nas systemach wykorzystujemy architekturę mikroserwisów, która nie tylko zapewnia niesamowitą skalowalność systemów, ale również umożliwia wykorzystanie różnych technologii, które będą realizowały najlepiej zadania poszczególnych mikroserwisów.
   W przypadku pojedynczej instancji mikroserwisy uruchamiane są w kontenerach, natomiast nic nie stoi na przeszkodzie, aby funkcjonowały one w ramach nie tylko kontenerów ale także odrębnych maszyn w różnych lokalizacjach celem zapewnienia bardzo wysokiego SLA.
   Przykładowa architektura mikroserwisów dla aplikacji Serwer Biletowy mogłaby obejmować następujące mikroserwisy:
   - frontService – API dla operatorów zewnętrznych
   - tariffService – komponent taryfowy
   - cityService – komponent zarządzający właścicielami taryf, pełniącymi rolę nadrzędną nad taryfą, jeżeli w ramach systemu mają funkcjonować taryfy różnych miast/przewoźników, itp.
   - transactionService – komponent bazy transakcji biletowych. Jeżeli wydanie biletu odbywa się w ramach systemu, także odpowiedzialny za wydane biletu (biletów – jeżeli w ramach pojedynczej transakcji dozwolone jest wydanie więcej jak jednego biletu). Ponadto obsługa zwrotów.
   - QR generator – centralizacja procesu generowania kodów QR/NFC
   - validationService – komponent zarządzania procesem kontroli
   - stockService – komponent zarządzania magazynem biletowym, jeżeli w systemie wymagana jest uprzednia rezerwacja puli biletowych
   -panelService (wraz z innymi mikroserwisami wspomagającymi) – panel administracyjny

2. Proszę opisać technologię proponowanego rozwiązania (np. z jakich komponentów będzie zbudowane rozwiązanie: m.in. system/y operacyjny, baza/y danych, Język/i programowania, Dockery, wirtualizacja, oprogramowanie pośrednie).

   Dla proponowanego rozwiązania najlepszym systemem operacyjnym będzie Linux wraz aplikacją Docker. Rekomendowaną bazą danych dla rozwiązania mógłby być PostgreSQL.
   W nawiązaniu do pkt. 1 powyżej rekomendujemy następujące języki programowania:
   - Node.js – technologia wydajna i odporna na dużą liczbę połączeń z niewielką ilością danych
   - JAVA SpringBoot – technologia rekomendowana dla mikroserwisów transakcyjnych, zarządzających dużymi zbiorami danych
   - Angular.js/React.js – interfejsy www – panel administracyjny

- PHP/Python – technologia rekomendowana dla mikroserwisów których utrzymanie będzie wiązało się z częstymi modyfikacjami, dostosowaniem do zmieniających się okoliczności zewnętrznych, itp.

3. Czy możecie zaproponować rozwiązanie chmurowe oparte o własną chmurę?

   Możemy zaproponować rozwiązanie chmurowe hostowane w oparciu o jednego z wiodących dostawców rozwiązań chmurowych, niekoniecznie o charakterze globalnym. Natomiast nie utrzymujemy własnej serwerowni, w oparciu o którą moglibyśmy zaproponować hosting w oparciu o własną chmurę.

4. Czy występują płatne licencje lub ograniczenia ilościowe itp.?

   W naszych rozwiązaniach wykorzystujemy technologie, których zasady licencyjne nie wiążą się z dodatkowymi opłatami ani ograniczeniami ilościowymi, chyba, że za porozumieniem i przy pełnej akceptacji Zamawiającego (np. mapy na frontendzie).

5. Czy możliwe będzie skalowanie rozwiązania, jako parametr wydajności proszę wziąć pod uwagę liczbę transakcji (obszar danych) oraz ich jednoczesność (czas dostępu do usługi)?

   W nawiązaniu do pkt. 1 rozwiązanie będzie skalowalne. Wydajność zależy od uzgodnionej konfiguracji i wydelegowanych zasobów sprzętowych. Na tą chwilę możemy tylko potwierdzić, że w jednym z postępowań nasze systemy spełniły wymów jednoczesnego procesowania 100 zapytań.

6. Jakie możecie zaproponować zabezpieczenia (np., szyfrowanie, klucze, certyfikaty), oparte o jakie komponenty (software oraz hardware)?

   W zależności od warstwy systemu:
   - dostęp dla operatorów zewnętrznych do API powinien odbywać się po uprzednim zestawieniu połączenia VPN. Ponadto każdy request powinien być identyfikowany kluczem operatora oraz uwierzytelniany sygnaturą cyfrową, wygenerowaną w oparciu o klucz „secret”.
   - w obszarze bezpieczeństwa procesu generowania QR – możliwe są dwa zasadnicze modele rozwiązania:
      - kod off-line – zawierający zaszyte parametry biletu, podpisane sygnaturą cyfrową z cyklicznie zmiennymi kompletami kluczy (algorytm asymetryczny)
      - kod on-line – zawierający identyfikator biletu, którego weryfikacja będzie odbywała się poprzez odpytanie serwera przez sprawdzarkę. Rozwiązanie to dodatkowo umożliwia zapewnienie cyklicznej zmiany identyfikatora (np. co 5 minut), co praktycznie wyeliminuje pozasystemową dystrybucję wygenerowanych kodów, ale wiąże się za zapewnieniem dostępu do sieci zarówno ze strony użytkownika, jak i sprawdzarki kontrolerskiej

7. Czy będzie wykorzystywana szyna danych, jeżeli tak to jakie rozwiązanie?

Nie widzimy zastosowania szyny danych dla rozwiązania, przynajmniej w obszarze komunikacji z operatorami zewnętrznymi. Założeniem jest udostępnienie im interfejsu w modelu REST z wykorzystaniem protokołu komunikacyjnego JSON.

Uzasadnieniem dla szyny danych mogłaby być komunikacja z istniejącymi systemami, ale wymaga to analizy pod kątem mnogości interfejsów, gdyż szyna danych jest frameworkiem, który należy oprogramować, więc analiza powinna wykazać zasadność przygotowania nadmiarowego komponentu.

8. Jak zostaną zaprojektowane zapytania i odpowiedzi, np. SOAP, REST itp.?

Nasze dotychczasowe doświadczenie zdecydowanie skłania nas do rekomendacji modelu REST wraz z formatem zapytań i odpowiedzi JSON, celem zapewnienia możliwie najmniejszego narzutu ilości przesyłanych danych pomiędzy elementami systemu, co ma szczególne znaczenie w przypadku komunikacji z urządzeniami mobilnymi.

9. Proszę opisać proponowane zabezpieczenia przed fraudami ze strony użytkowników aplikacji (np. QRcode, animowany gif, kod, itp.).

   Najlepszym zabezpieczeniem jest wspomniany w odpowiedzi na pkt. 6 model kontroli on-line z dużą częstotliwością aktualizacji identyfikatora (np. 5 minut). Ponadto bilety krótkookresowe powinny być zabezpieczone identyfikatorem pojazdu (o ile taka możliwość istnieje), nawet jeżeli jest to bilet przesiadkowy, co znacznie ograniczy możliwości fraudów.
   Dodatkowo każdy bilet powinien być przypisany do urządzenia, na którym został zakupiony, a zmiana urządzenia możliwa w sposób kontrolowany i limitowany.
   Oczywiście warto także zabezpieczyć warstwę prezentacji, tj. zapewnić kontrolerom możliwość weryfikacji, czy bilet został zaprezentowany w aplikacji mobilnej. Tutaj dobrze sprawdzają się tzw. elementy interaktywne, czyli nie tylko animacje, ale również efekty wizualne inicjowane przez użytkownika (np. animacja w okolicy kodu QR, w obszarze, który nie przeszkodzi w procesie kontroli, wyzwolone kliknięciem na kod lub tuż po jego zeskanowaniu).

10. Proszę opisać minimalny zestaw danych niezbędny do świadczenia usługi.

    Nie jestem pewien, czy dobrze rozumiem pytanie, natomiast z doświadczenia wiem, że najlepsze systemy to takie, które są w stanie funkcjonować niezależnie od innych, a integracje powinny być traktowane jako dodatkowe możliwości, które usprawnią wiele procesów, ale nie są niezbędne szczególnie

w początkowym okresie funkcjonowania systemu. Dlatego na dobrą sprawę Serwer Biletowy może funkcjonować jako niezależna aplikacja, także w obszarze niezależnie definiowanej taryfy biletowej, w związku z powyższym jako minimalny zestaw danych wskazałbym wymogi dotyczące bieżącej taryfy biletowej i ich planowanych modyfikacji celem właściwego zaprojektowania mechanizmów jej definiowania i walidacji oraz określenie wymogów dotyczących formatów raportowania.

11. Zamawiający będzie wymagał integracji z systemem ŚKUP w zakresie Modułu Taryf i Cenników - MTC (pobieranie raz na dobę, z możliwością jego edycji), systemu księgowo-finansowego, Modułu Analityczno-Raportowego (MAR) – dane transakcyjne, z przeprowadzonych kontroli.

    Integracja może odbywać się w dwóch modelach:
    - dla systemów zewnętrznych aplikacja udostępnia dedykowane interfejsy umożliwiające rejestrację w systemie taryfy biletowej oraz pobranie raportów w parametryzowanym zakresie i częstotliwości
    - aplikacja posiada mechanizmy cyklicznego odpytywania o taryfę biletową oraz cyklicznego generowania raportów wraz z ich dystrybucją do zewnętrznego systemu z wykorzystaniem uzgodnionego interfejsu

12. Propagacja informacji o biletach (serwer->klient czy inaczej).

    Jeżeli klientem jest operator zewnętrzny, to zalecaną propagacją jest kierunek klient->serwer, tzn. operator (klient) będzie odpytywał serwer i na tej podstawie realizowane będą funkcjonalności systemu. W przypadku innego modelu operatorzy musieliby udostępniać własne API, co może być problematyczne zarówno w obszarze spójności, zapewnienia bezpieczeństwa i szeroko pojętego utrzymania systemu, a uzasadnieniem takiego odwróconego modelu mogłyby być tylko transakcje o charakterze asynchronicznym, na chwilę obecną nie widzimy takiej potrzeby.

13. Propozycja zaimplementowania taryfy CICO, czy jest możliwa? (zintegrowanej z MTiC ŚKUP).

Tak, przewidujemy dedykowane mechanizmy, a architektura mikroserwisów także zapewnia bezproblemowy rozwój aplikacji o dedykowane komponenty, właśnie poprzez dodanie dedykowanych mikroserwisów odpowiedzialnych za nowe obszary funkcjonowania systemu.

14. Integracja z kontrolerkami.

Tak, poprzez wystawienie API dla kontrolerów niezbędne zarówno w modelu kontroli off-line (urządzenia cyklicznie muszą wtedy aktualizować klucze publiczne), jak i on-line, gdzie będą odpytywać o ważność biletów.

15. Raporty Real time.

Możliwe zarówno w panelu, jak i poprzez wystawiony webservice raportowy.

16. Raporty dzienne.

j. w.

17. Blokowanie sprzedaży.

Tak, per operator, ale także np. w przypadku trwającej kontroli w pojeździe lub jeżeli operator nie pobierze raz dziennie taryfy biletowej, przekroczy limit sprzedażowy lub wygaśnie jego kontrakt.

18. Diagnostyka.

Diagnostyka systemu obejmuje dwie zasadnicze płaszczyzny:

- obszar prawidłowego funkcjonowania systemu

- obszar prawidłowo realizowanych procesów dystrybucji biletów

W płaszczyźnie funkcjonowania systemu, monitorowane dostępności poszczególnych mikroserwisów, połączenia do usług zewnętrznych, a brak wykonania danego zadania cyklicznego, dodatkowo skutkować będzie alertem.

W płaszczyźnie realizowanych procesów dystrybucji biletów, diagnostyka bardziej będzie wynikała z przetwarzania danych sprzedażowych i wykrywania anomalii, przykładowo:

- zbyt duża liczba kontroli danego biletu – podejrzenie transakcji oszukańczej

- brak sprzedaży danego rodzaju biletu w zadanym czasie – problem z dostępnością biletu, błędem w zaktualizowanej taryfie, etc.

19. Tokenizacja nr tel.

W jednym z systemów szyfrujemy wrażliwe dane osobowe (a czasem do takowych zaliczany jest numer telefonu), zatem nawet w kodzie QR nie jest on dostępny w sposób jawny. Myślę, że podobne rozwiązanie spełniłoby Państwa oczekiwania w ww. obszarze.

20. Jakie dane od użytkownika są niezbędne w aplikacjach sprzedających bilety elektroniczne?

Do ustalenia, im mniej tym lepiej, natomiast celem zapewnienia bezpieczeństwa w procesie dystrybucji, użytkownik powinien być identyfikowany trwałym (w pewnym zakresie) i weryfikowanym identyfikatorem i tutaj dobrze się spisuje numer telefonu.

Ponadto podczas zakupu biletu imiennego, minimalnym zakresem danych jest jego imię i nazwisko

21. Zarządzanie kluczami prywatnymi i publicznymi.

W ramach validationService, przy założeniu centralizacji procesu generowania QR, bo alternatywą jest przeniesienie tego procesu na operatorów, wtedy zarządzanie kluczami wymaga nieco innego podejścia i musi ono dotyczyć osobno każdego operatora.

22. Nr seryjne, git, identyfikacja biletu.
    Centralizacja procesu wydawania biletów umożliwi nadawanie jednolitych numerów, w ramach serii dedykowanych na poszczególne kanały dystrybucji, czy operatorów.

23. Ważność biletu do kontroli.
    Centralizacja procesu wydawania biletów umożliwi walidację ważności biletu, w trybie on-line.

24. QR kod z zaszyfrowaną informacją o transakcji.
    Centralizacja procesu wydawania biletów wiąże się generowaniem ciągu znaków, będącego kodem QR. Dzięki temu możliwe będzie przyjęcie jednolitego modelu: np. off-line lub on-line, zapewnienie szyfrowania, czy weryfikacji poprzez sygnaturę cyfrową.

25. Zarządzanie podmiotami, przypisywanie biletów, prowizji, od kiedy, do kiedy.
    System umożliwia rejestrację zewnętrznych operatorów, określenie parametrów biznesowych współpracy:
    - prowizja od biletów
    - czas obowiązywania umowy
    - limit ilościowy
    - limit kwotowy
    - flaga aktywności

    oraz nadanie parametrów dostępowych do API. W przypadku przekroczenia któregokolwiek z limitu lub niewywiązanie się z innych zobowiązań (np. brak aktualizacji taryfy w wymaganym okresie czasu), API zablokuje wydawanie biletów, więc skutek będzie natychmiastowy.

26. Sposób, warunki podłączania aplikacji sprzedających bilety (np. jak dojadę, skycash)?
    Integralnym składnikiem systemu będzie API udostępnione operatorom zewnętrznym:
    - cykliczną aktualizację taryfy giletowej
    - rejestrację transakcji biletowych (wydawanie biletu)
    - zarządzanie zwrotami
    - fakturowanie (inicjowane przez użytkownika lub z poziomu BOK)
    - raportowanie
    Zewnętrzni operatorzy uwierzytelnianie będą dwuetapowo, zarówno w warstwie sieciowej, jak i w warstwie aplikacji poprzez nadanie kluczy dostępowych wymaganych w procesie komunikacji.

27. Detekcja nadużyć.
    Jeżeli system będzie wspierał proces kontroli, możliwa będzie weryfikacja ww. procesu, głównie pod kątem nadużyć. W przypadku zbyt częstej kontroli danego biletu system będzie mógł wysyłać alerty oraz blokować transakcje biletowe zakwalifikowane jako podejrzane.

28. Backupowanie danych.
    Możliwe w ramach rozwiązania chmurowego lub dedykowanego, w ramach umowy utrzymaniowej, należy być jednak świadomym dodatkowych kosztów związanych z przechowywaniem backupów.

29. Praca w klastrze.
    Zagadnienie częściowo opisane powyżej, architektura umożliwia rozproszenie aplikacji backendowej zarówno w obrębie poszczególnych mikroserwisów, jak i replikowanych całościowych aplikacji backendowych.

30. Możliwa integracja z urządzeniami.
    Jeżeli model będzie przewidywał zarządzanie procesem kontroli biletów, będzie wystawiał API dla urządzeń kontrolerskich. Możliwe jest także rozszerzenie systemu w kierunku kompleksowego systemu centralnego, zintegrowanego z innymi kanałami dystrybucji biletów, takich jak np. automaty biletowe.

31. Wydajność rozwiązania.
    Zagadnienie częściowo opisane powyżej, próbując oszacować realna odpowiedź na zapytanie nie powinna przekroczyć 200 ms, przy jednoczesnej obsłudze 100 zapytań.