

System e-Toll

Dokumentacja systemu do zbierania danych na temat ruchu pojazdów po polskich drogach i naliczania należności za przejazdy

13 kwietnia 2025

<https://github.com/Artur-Romaniuk/ais>
<https://www.overleaf.com/project/67e843e5c78b52b01a88211a>

Wstęp

System e-Toll do zbierania danych na temat ruchu pojazdów po polskich drogach i naliczania należności za przejazdy.

Zakres dokumentu

Rezultatem projektu powinien być dokument PDF, zawierający następujące elementy:

1. Wymagania funkcjonalne.
 - pogrupowana lista wymagań
 - zidentyfikować jeden kluczowy proces biznesowy i szczegółowo ten proces zdefiniować. Definicja procesu to po prostu:
 - cel procesu
 - stan początkowy
 - stan końcowy
 - kroki procesu, z uwzględnieniem sytuacji wyjątkowych.
2. Wymagania niefunkcjonalne.

- geograficzna skala działania
 - liczba obsługiwanych klientów
 - liczba obsługiwanych zdarzeń biznesowych w określonym czasie (na godzinę/dziennie/miesięcznie etc.)
 - wymagania wydajnościowe
 - wymagania niezawodnościowe
 - wymagania bezpieczeństwa
 - ...
3. Projekt systemu w postaci modelu C4 (<https://c4model.com/>)
- wystarczą 3 pierwsze poziomy: context, containers, components, nie wymagam poziomu code
 - diagram dynamiczny (dynamic diagram) realizujący opisany w wymaganiach proces biznesowy
 - diagram wdrożenia (deployment diagram)
4. Dyskusja zastosowanych wzorców i/lub taktyk architektonicznych - w celu wyboru odpowiednich rozwiązań należy się odwołać do wymagań funkcjonalnych i нефunkcjonalnych.
5. Decyzje architektoniczne w postaci modelu MAD 2.0

1 Wymagania funkcjonalne

1.1 Kluczowy proces biznesowy: Naliczanie i pobieranie opłat za przejazd

1.1.1 Cel procesu

1. Wykrywanie pojazdów na płatnych odcinkach dróg
2. Naliczanie należności zgodnie z taryfami
3. Pobieranie opłat od użytkowników

1.1.2 Stan początkowy

Pojazd niezarejestrowany w systemie jako znajdujący się na płatnym odcinku drogi, z kontem użytkownika posiadającym określony stan środków (prepaid) lub limitem kredytowym (postpaid).

1.1.3 Stan końcowy

Prawidłowo naliczona i pobrana opłata za przejazd, zaksięgowana w systemie finansowym, z wygenerowanym potwierdzeniem dla użytkownika oraz aktualizacją stanu konta.

1.1.4 Kroki procesu

1. Identyfikacja pojazdu wjeżdżającego na płatny odcinek (poprzez urządzenie OBU, aplikację mobilną lub kamery ANPR) [Dla redundancji, im więcej systemów tym większa szansa na identyfikację]
2. Zebranie danych o pojeździe (kategoria, klasa emisji spalin, masa)
3. Rozpoczęcie śledzenia przejazdu i rejestracja czasu wjazdu
4. Monitorowanie trasy przejazdu przez punkty kontrolne
5. Identyfikacja zjazdu z płatnego odcinka
6. Obliczenie należności na podstawie przebytej trasy i charakterystyki pojazdu
7. Weryfikacja dostępności środków na koncie użytkownika
8. Pobranie opłaty z konta użytkownika
9. Wygenerowanie potwierdzenia transakcji
10. Aktualizacja salda konta użytkownika

Obsługa sytuacji wyjątkowych:

- Jeśli identyfikacja pojazdu nie jest możliwa: uruchomienie procedury rejestracji incydentu z dokumentacją wizualną (zdjęcia niezidentyfikowanego pojazdu)
- W przypadku niewystarczających środków na koncie:
 1. Zablokowanie konta do momentu zapłacenia
 2. W przypadku braku zapłaty w określonym czasie, uruchomienie procedury windykacyjnej
- Przy awarii urządzenia OBU: automatyczne przełączenie na identyfikację przez system kamer ANPR
- W razie przerwy w komunikacji: lokalne buforowanie danych i synchronizacja po przywróceniu łączności
- Jeśli zidentyfikowano omijanie bramek:
 1. Poinformowanie użytkownika
 2. Uruchomienie procedury kontrolnej
 3. Naliczenie kary

1.2 Lista wymagań funkcjonalnych

1. **System rejestracji użytkowników** - System musi umożliwiać rejestrację nowych użytkowników z weryfikacją tożsamości, zbieraniem danych o pojazdach (w tym masa, klasa emisji, kategoria) oraz wyborem metody płatności (prepaid lub postpaid).
2. **Moduł geolokalizacji pojazdów** - System musi określać pozycję pojazdów z dokładnością do 10 metrów, wykorzystując dane GPS z urządzeń OBU lub aplikacji mobilnej, aktualizowane nie rzadziej niż co 30 sekund podczas przejazdu.
3. **System rozpoznawania tablic rejestracyjnych (ANPR)** - System musi identyfikować pojazdy poprzez kamery ANPR z dokładnością co najmniej 98% w różnych warunkach pogodowych i oświetleniowych, jako dopełnienie jeśli wszystkie systemy działają poprawnie lub alternatywa dla urządzeń OBU w razie awarii.

4. **Elastyczny system taryfowy** - System musi obsługiwać zróżnicowane taryfy opłat w zależności od: typu pojazdu, masy całkowitej, klasy emisji spalin, pory dnia, dnia tygodnia oraz stopnia zatłoczenia drogi.
5. **Moduł rozliczeń i płatności** - System musi obsługiwać różne metody płatności (karty płatnicze, przelewy, płatności mobilne, blik) z możliwością automatycznego doładowania konta prepaid oraz wystawiania faktur elektronicznych zgodnych z przepisami prawa.
6. **System powiadomień dla użytkowników** - System musi wysyłać automatyczne powiadomienia do użytkowników (aplikacja, mail + sms) o: niskim stanie konta, zbliżającym się terminie płatności, dokonanych transakcjach oraz zmianach w taryfach i regulaminie.
7. **Moduł raportowania i analityki** - System musi generować raporty dotyczące natężenia ruchu, generowanych przychodów, incydentów oraz efektywności egzekwowania opłat, z możliwością eksportu danych w formatach CSV i PDF.
8. **System wykrywania naruszeń** - System musi identyfikować próby obejścia opłat (np. manipulacja urządzeniem OBU, podawanie fałszywych danych o pojeździe) i automatycznie uruchamiać procedury weryfikacyjne.
9. **Portal samoobsługowy dla użytkowników** - System musi udostępniać portal internetowy oraz aplikację mobilną umożliwiającą użytkownikom zarządzanie kontem, przeglądanie historii przejazdów i opłat, generowanie raportów oraz aktualizację danych pojazdu.
10. **System korekty opłat** - System musi umożliwiać ręczną korektę naliczonej opłaty przez operatora w przypadku zgłoszenia błędu przez użytkownika. Zakładając, że system źle naliczył opłatę np. Użytkownik został obciążony za trasę, której nie przejechał lub przypisano niewłaściwą kategorię pojazdu. Użytkownik zgłasza błąd do obsługi systemu. W takim przypadku operator musi mieć możliwość ręcznej korekty naliczonej opłaty np. Anulowanie opłaty, zmniejszenie jej lub ponowne przeliczenie, jeżeli zgłoszenie użytkownika okaże się zasadne. Zasadność zgłoszenia musi być weryfikowalna poprzez fizyczne zdjęcia pojazdu.

2 Wymagania niefunkcjonalne

1. **Wydajność systemu** - System musi obsługiwać jednoczesne przetwarzanie danych z co najmniej 200,000 pojazdów w godzinach szczytu¹, z czasem odpowiedzi dla transakcji poniżej 1s oraz przetwarzaniem danych geolokalizacyjnych w czasie rzeczywistym.
2. **Dostępność systemu** - System musi zapewniać dostępność na poziomie 99,9% (maksymalny czas niedostępności: ~9 godzin rocznie), z planowanymi oknami serwisowymi w godzinach nocnych (1:00-4:00) i z odpowiednim wyprzedzeniem komunikowanym użytkownikom.
3. **Bezpieczeństwo danych** - System musi zapewniać:
 - (a) Szyfrowanie danych w spoczynku i podczas transmisji (minimum AES-256)
 - (b) Zgodność z normą ISO/IEC 27001
 - (c) Wielopoziomową autoryzację użytkowników (hasło + kod sms)
 - (d) Pełną zgodność z RODO, włączając automatyczne mechanizmy anonimizacji danych historycznych starszych niż 5 lat.
4. **Skalowalność** - System musi posiadać architekturę umożliwiającą skalowanie w celu obsługi wzrostu liczby użytkowników o 30% rocznie bez pogorszenia wydajności², z automatycznym zwiększeniem zasobów w odpowiedzi na zwiększone obciążenie w ciągu dnia.
5. **Niezawodność i odporność na awarie** - System musi zawierać rozwiązania wysokiej dostępności z nadmiarowością komponentów krytycznych, automatycznym przełączaniem awaryjnym poniżej 10 sekund oraz mechanizmem ciągłej replikacji danych między geograficznie oddalonymi centrami danych, zapewniając RPO (Recovery Point Objective) poniżej 5 minut i RTO (Recovery Time Objective) poniżej 30 minut.

¹<https://forsal.pl/transport/drogi/artykuly/8295957,najbardziej-obciazone-drogi-w-polsce-s8-s2-a4-s86-mapa.html>

²<https://kpmg.com/pl/pl/home/media/press-releases/2024/02/liczba-rejestracji-nowych-samochodow-osobowych-wzrosla-o-13-2-procent-w-2023-roku.html>

6. **Interoperacyjność** - System musi obsługiwać standardy interoperacyjności z europejskimi systemami elektronicznego poboru opłat (zgodnie z dyrektywą EETS³), zapewniając pełną wymianę danych poprzez standardowe API (REST/SOAP) z minimum 99,5% dostępnością interfejsów integracyjnych.
7. **Użyteczność i dostępność interfejsów** - Interfejsy użytkownika (portal i aplikacja mobilna) muszą spełniać standardy WCAG 2.1 na poziomie AA, obsługiwać minimum 5 języków (polski, angielski, niemiecki, ukraiński, rosyjski), zapewniać responsywność na urządzeniach mobilnych oraz wykazywać wskaźnik satysfakcji użytkowników (CSAT) na poziomie minimum 85%.
8. **Audytowanie i śledzenie aktywności** - System musi rejestrować wszystkie operacje w niezmiennialne logi z zachowaniem zgodności z wymogami prawnymi dotyczącymi dowodów elektronicznych, umożliwiać niemodyfikowalny ślad audytu dla wszystkich transakcji finansowych oraz zapewniać przechowywanie logów przez minimum 5 lat z możliwością szybkiego wyszukiwania.
9. **Efektywność zarządzania danymi** - System musi umożliwiać archiwizację i zarządzanie cyklem życia danych zgodnie z polityką retencji, zapewniać kompresję danych historycznych na poziomie minimum 80% oraz optymalizację zapytań do bazy danych z czasem odpowiedzi poniżej 10 sekund dla 90% zapytań raportowych.
10. **Utrzymywalność i modyfikowalność** - System musi być zaprojektowany z wykorzystaniem architektury modułowej i mikroserwisowej, umożliwiającej niezależną aktualizację poszczególnych komponentów bez przerywania działania całości systemu, z automatycznymi testami regresji pokrywającymi minimum 90% kodu oraz pełną dokumentacją techniczną aktualizowaną przy każdej **dużej** (takiej która sprawia że system nie jest kompatybilny z poprzednią wersją) zmianie.
11. **System integracji z zewnętrznymi bazami danych** - System musi komunikować się z zewnętrznymi bazami danych (np. CEPiK, rejestry

³<https://eur-lex.europa.eu/legal-content/PL/TXT/?uri=CELEX%3A32019L0520>

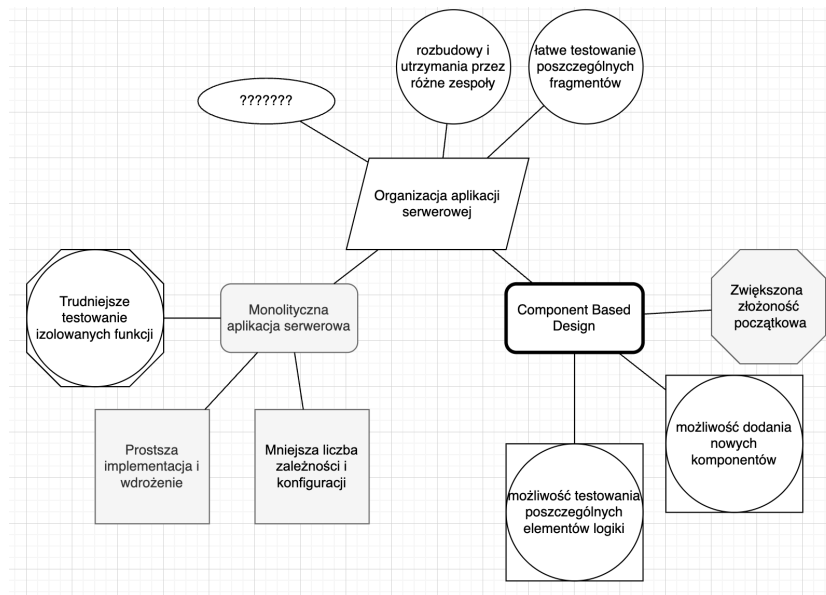
pojazdów innych krajów) w celu weryfikacji danych pojazdów oraz wymiany informacji o użytkownikach z zagranicznymi systemami poboru opłat.

12. **Zgodność prawna i regulacyjna** - System musi spełniać wszystkie obowiązujące przepisy prawa krajowego oraz unijnego dotyczące elektronicznego poboru opłat. Ustawę o drogach publicznych, dyrektywę EETS, przepisy podatkowe oraz przepisy dotyczące ochrony konkurencji i konsumentów.
13. **Czas wdrożenia poprawek krytycznych** - w przypadku wykrycia krytycznego błędu (np. Uniemożliwiającego naliczenie opłaty lub przetwarzanie przejazdów) poprawka musi zostać wdrożona w ciągu maksymalnie 24 godzin od momentu potwierdzenia błędu.
14. **Transparentność naliczanych opłat** - system musi umożliwiać użytkownikom końcowym wgląd w szczegółowe informacje dotyczące każdej naliczonej opłaty. Czas przejazdu, odcinki dróg, taryfy oraz podstawy naliczenia.
15. **Elastyczność taryf** - system musi obsługiwać dynamiczne taryfy drogowe (np. Zmienne w zależności od natężenia ruchu czy poziomu emisji pojazdu) z możliwością wdrażania nowych taryf bez konieczności przerywania działania systemu.
16. **Personalizacja powiadomień dla użytkowników** - system musi umożliwić użytkownikom wybór otrzymywania powiadomień np. (sms, e-mail, powiadomienie push w aplikacji) z opcją definiowania progów powiadomień (np. Przekroczenie salda, opłata powyżej X zł etc.)

3 Decyzje architektoniczne

3.1 Podział na warstwy

- Warstwa prezentacji: trzy różne aplikacje klienckie — aplikacja webowa, mobilna i aplikacja na urządzenia embedded.
- Warstwa logiki biznesowej: centralna aplikacja serwerowa (serverApp), w której znajdują się komponenty takie jak UserService, PaymentService, PositionService itd.



Rysunek 1: Diagram podziału na warstwy

- Warstwa dostępu do danych: komponenty UserRepository, PaymentRepository.
- Warstwa danych: baza danych Oracle z mechanizmem failover.

Decyzja: Rozdzielenie odpowiedzialności na warstwy zwiększa modularność i ułatwia zarządzanie kodem oraz jego testowanie.

Alternatywa: Architektura mikroservisowa

Zalety alternatywy:

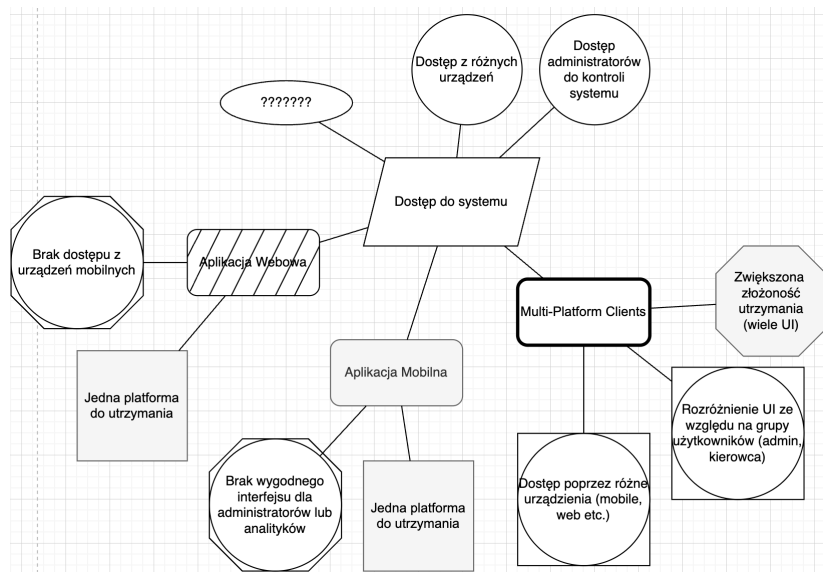
- Lepsza skalowalność poszczególnych komponentów
- Możliwość niezależnego wdrażania i rozwijania usług
- Lepsza odporność na awarie (awaria jednego serwisu \neq awaria całego systemu)

Wady alternatywy:

- Większa złożoność wdrożeniowa (DevOps, CI/CD, monitoring)

- Konieczność rozwiązywania problemów związanych z komunikacją między serwisami
- Trudniejsze debugowanie i testowanie end-to-end

3.2 Modularność i komponenty (Component-based Design)



Rysunek 2: Diagram komponentów

- Serwerowa aplikacja została podzielona na komponenty pełniące konkretne role (SignInController, TollController, MainComponent, itd.).
- Każdy komponent ma jasno zdefiniowaną odpowiedzialność, zgodnie z zasadą Single Responsibility Principle.

Decyzja: Wprowadzenie komponentów umożliwia łatwe rozszerzanie i testowanie poszczególnych fragmentów systemu.

Alternatywa: Monolityczna aplikacja serwerowa

Zalety alternatywy:

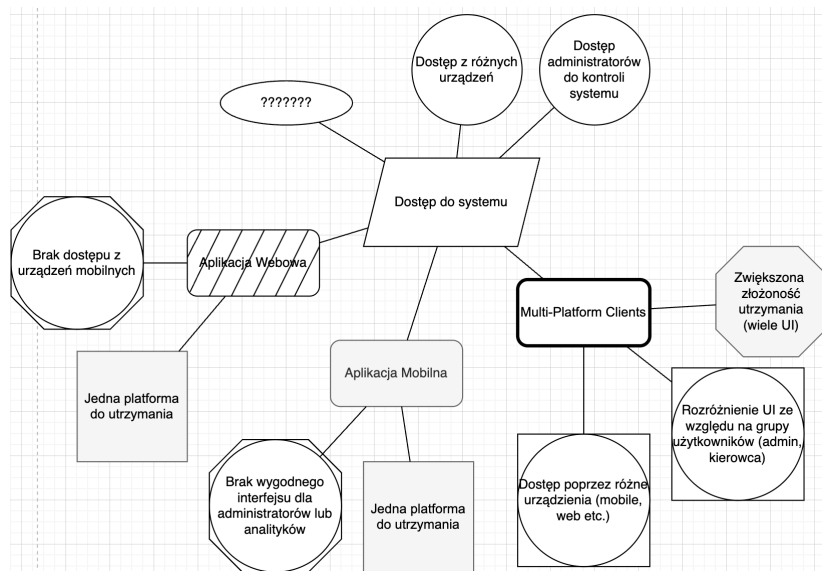
- Prostsza implementacja i wdrożenie

- Mniejsza liczba zależności i konfiguracji
- Mniej złożone środowisko developerskie

Wady alternatywy:

- Trudniejsza skalowalność i refaktoryzacja
- Każda zmiana wymaga redeploy całej aplikacji
- Trudniejsze testowanie izolowanych funkcji

3.3 Wielokanałowy dostęp (Multi-Platform Clients)



Rysunek 3: Diagram wielokanałowego dostępu

- Użytkownicy mogą korzystać z systemu za pomocą aplikacji mobilnej, webowej lub urządzeń embedded.

Decyzja: Umożliwienie różnym grupom użytkowników (np. administratorzy vs kierowcy) dostępu do funkcji systemu w najbardziej dogodny sposób.

Alternatywa: Tylko aplikacja mobilna (np. PWA lub natywna)

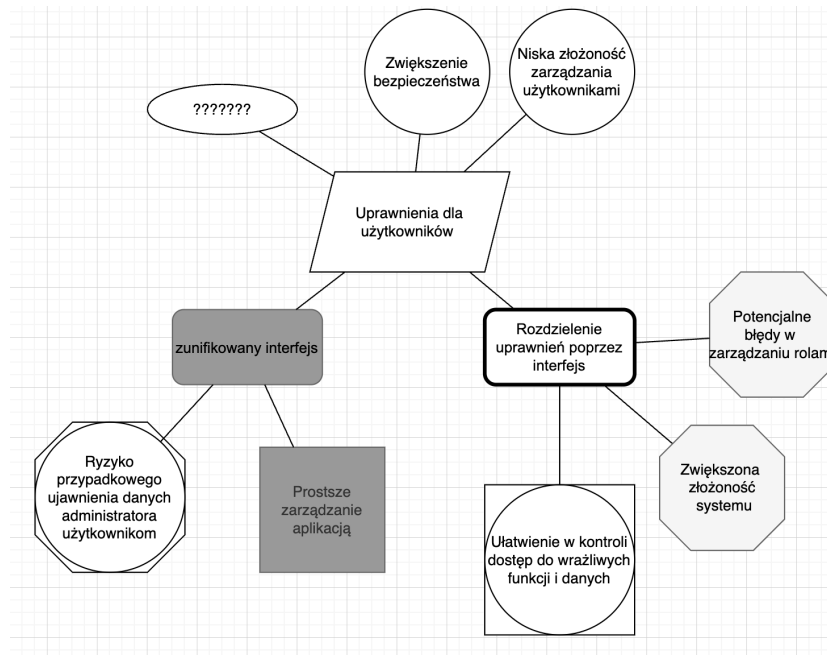
Zalety alternatywy:

- Uproszczony interfejs użytkownika
- Jedna platforma do utrzymania
- Lepsze dopasowanie do kontekstu użytkownika (kierowcy)

Wady alternatywy:

- Brak wygodnego interfejsu dla administratorów lub analityków
- Mniejsza elastyczność użytkowania
- Trudności z dostępem do systemu z urządzeń stacjonarnych

3.4 Rozdzielenie ról i uprawnień



Rysunek 4: Diagram ról i uprawnień

- Administrator ma dostęp tylko przez aplikację webową.

- Kierowca może korzystać z trzech różnych interfejsów — w zależności od potrzeb.

Decyzja: Jasny podział ról zwiększa bezpieczeństwo i ergonomię systemu.

Alternatywa: Jeden zunifikowany interfejs z uprawnieniami na poziomie konta

Zalety alternatywy:

- Mniejsze zróżnicowanie UI
- Mniej kodu i testów związanych z różnymi platformami

Wady alternatywy:

- Mniejsza przejrzystość
- Możliwość przypadkowego ujawnienia funkcji nieprzeznaczonych dla danego typu użytkownika

3.5 Integracja z zewnętrznymi systemami

- System płatności (systemPłatnosci) oraz system archiwizacji danych (archiwum) są zewnętrznymi systemami zintegrowanymi z systemem e-Toll.

Decyzja: Wydzielenie tych odpowiedzialności do zewnętrznych systemów pozwala na lepsze skalowanie oraz wykorzystanie istniejących rozwiązań.

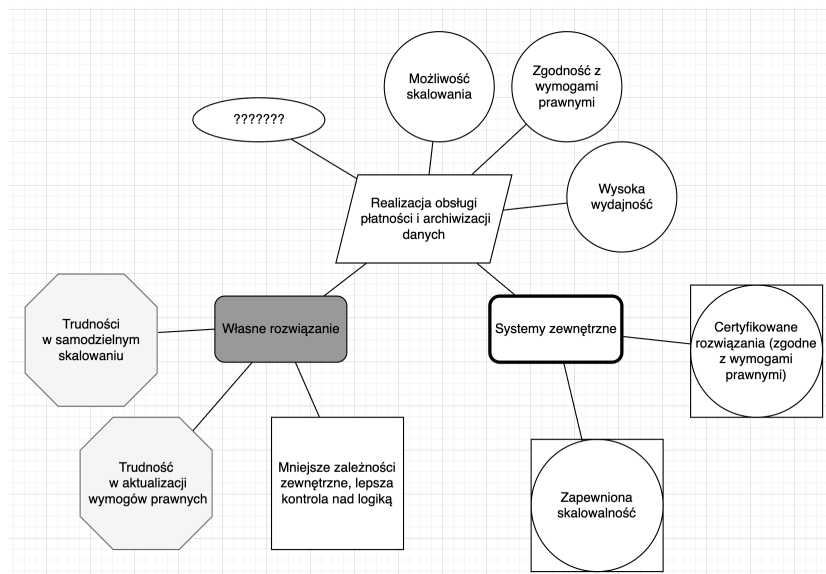
Alternatywa: Wszystko w ramach jednej aplikacji (np. własny moduł płatności i archiwizacji)

Zalety alternatywy:

- Większa kontrola nad logiką
- Mniejsze zależności zewnętrzne

Wady alternatywy:

- Większe ryzyko błędów w obszarach regulowanych prawnie (np. płatności)
- Większe koszty utrzymania i certyfikacji
- Brak skalowalności i elastyczności



Rysunek 5: Diagram integracji z systemami zewnętrznymi

3.6 Wysoka dostępność i odporność na awarie

- Baza danych jest replikowana (primary–secondary).
- Oddzielne deployment node’y dla różnych typów urządzeń oraz dla aplikacji serwerowej i bazy danych.
- Failover serwera bazy danych — Oracle - Secondary.

Decyzja: Architektura uwzględnia mechanizmy zapewniające ciągłość działania systemu nawet w przypadku awarii.

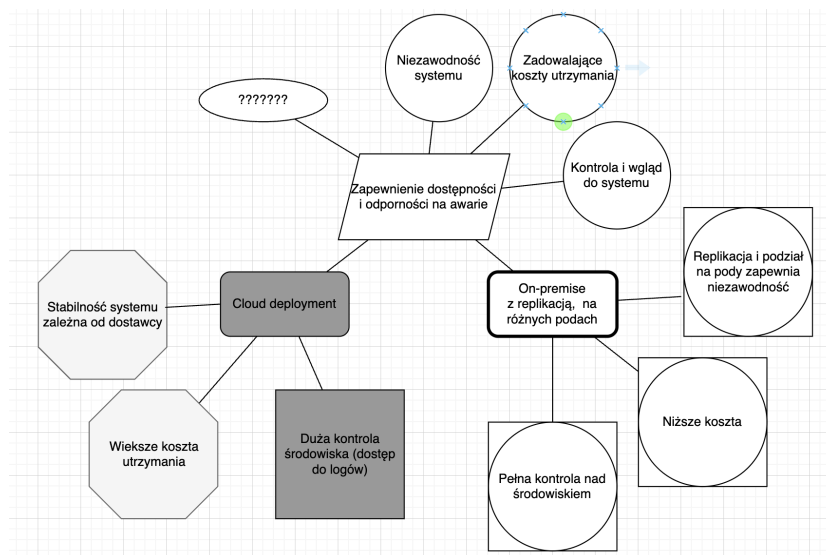
Alternatywa: Deployment w chmurze

Zalety alternatywy:

- Duża skalowalność
- Bezpieczeństwo i redundancja

Wady alternatywy:

- Wyższe koszty
- Przetwarzanie danych przez inne podmioty



Rysunek 6: Diagram wysokiej dostępności

3.7 Wybór technologii wdrożeniowych

- System hostowany na serwerach z Ubuntu 24.04 LTS.
- Serwer aplikacji oparty o Apache Tomcat 8.x.
- Baza danych Oracle 12c.

Decyzja: Użycie sprawdzonych technologii o długoterminowym wsparciu i wysokiej wydajności.

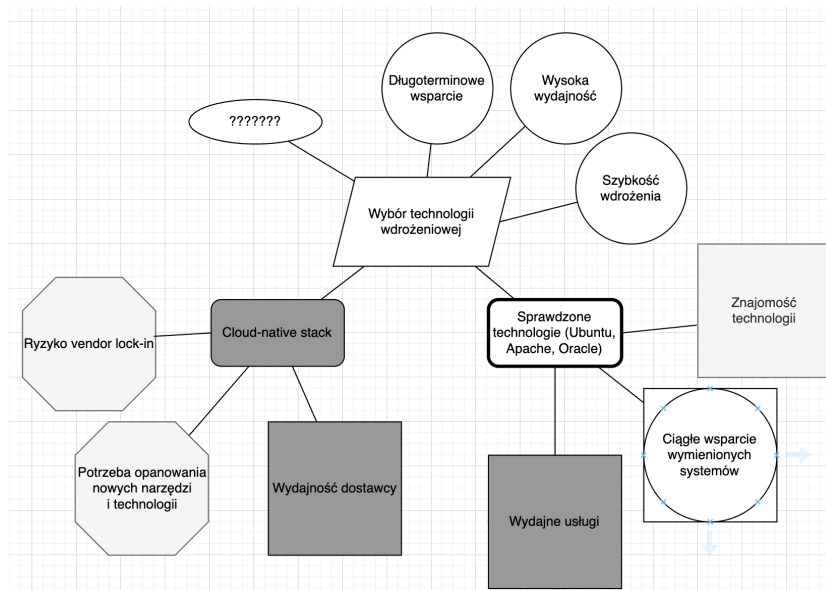
Alternatywa: Cloud-native stack – np. AWS/GCP

Zalety alternatywy:

- Automatyczne skalowanie, monitoring, CI/CD
- Niższe koszty utrzymania fizycznej infrastruktury
- Łatwiejsze zarządzanie kontenerami i usługami

Wady alternatywy:

- Uzależnienie od chmury (vendor lock-in)
- Potrzeba opanowania nowych technologii
- Potencjalne wyższe koszty początkowe



Rysunek 7: Diagram technologii wdrożeniowych