***Strona tytułowa***

### **Kluczowe Wymagania Funkcjonalne:**

1. **Zarządzanie Flotą:**
   * System musi umożliwiać dodawanie, usuwanie i modyfikację informacji o poszczególnych samochodach w flocie.
   * Funkcja powinna obsługiwać dane takie jak model, numer rejestracyjny, rok produkcji, itp.
2. **Zarządzanie Flotami:**
   * System musi oferować funkcję dodawania, usuwania i modyfikacji danych dotyczących poszczególnych flot.
   * Floty powinny być identyfikowane, a informacje na ich temat przechowywane, takie jak nazwa floty, opis, przypisane samochody, itp.
3. **Monitoring Auta:**
   * Aplikacja powinna umożliwiać podgląd pojedynczego samochodu, prezentując informacje o stanie technicznym, zużyciu paliwa i przypisanym kierowcy.
   * Dostęp do danych monitoringu powinien być zabezpieczony odpowiednimi uprawnieniami.
4. **Filtrowanie:**
   * System musi dostarczać funkcję filtrowania w wyszukiwarce aut z bliskim terminem zużycia podzespołów samochodowych.
   * Filtry powinny obejmować różne parametry, takie jak stan techniczny, terminy serwisów, itp.
5. **Śledzenie Naprawy:**
   * Aplikacja powinna pozwalać na przypisanie samochodu do trybu naprawy, umożliwiając śledzenie terminów i lokalizacji warsztatu.
   * Status naprawy powinien być dostępny dla uprawnionych użytkowników.
6. **Uwierzytelnianie Użytkownika:**
   * System musi umożliwiać użytkownikom tworzenie kont, logowanie się i resetowanie haseł.
   * Wymagane są odpowiednie mechanizmy bezpieczeństwa, takie jak potwierdzenia emailowe przy rejestracji.
7. **Wyszukiwanie:**
   * Aplikacja powinna oferować funkcję zaawansowanego wyszukiwania opartego na słowach kluczowych.
   * Wyszukiwanie powinno obejmować zarówno pojazdy, jak i floty, umożliwiając szybkie odnajdywanie potrzebnych informacji.

### **Koncepcja Architektury Systemu (z punktu widzenia klienta):**

1. **Mikroserwisowa Architektura:**
   * System będzie oparty na architekturze mikroserwisowej, co umożliwia niezależne rozwijanie i skalowanie poszczególnych funkcji.
2. **Interfejs Użytkownika:**
   * Prosty, intuicyjny interfejs użytkownika z wyraźnym podziałem na moduły zarządzania pojazdami, flotami, monitorem aut, itp.
3. **Responsywność:**
   * Aplikacja będzie responsywna, dostępna zarówno na urządzeniach stacjonarnych, jak i mobilnych.

### **Wymagania Bezpieczeństwa i Ochrony:**

1. **Uwierzytelnianie i Autoryzacja:**
   * Bezpieczny mechanizm uwierzytelniania i autoryzacji użytkowników.
   * Ograniczony dostęp do danych zależnie od roli użytkownika.
2. **Ochrona Danych:**
   * Szyfrowanie danych w trakcie przesyłania i przechowywania.
   * Regularne backupy danych oraz zabezpieczenia przed utratą danych.

### **Wymagania Bezpieczeństwa i Ochrony (Implementacje):**

1. **Uwierzytelnianie i Autoryzacja:**
   * **Implementacja:** Wykorzystanie protokołu OAuth 2.0 w połączeniu z JWT (JSON Web Token) do uwierzytelniania użytkowników. Przydzielenie ról (administrator, kierowca, operator floty) oraz nadawanie odpowiednich uprawnień na podstawie tych ról. Zastosowanie silnych haseł, a także monitorowanie i blokowanie nieudanych prób logowania.
2. **Ochrona Danych:**
   * **Implementacja:** Wdrożenie protokołu HTTPS dla komunikacji między klientem a serwerem w celu zabezpieczenia danych w trakcie przesyłania. Szyfrowanie danych na poziomie bazy danych, z wykorzystaniem algorytmów silnego szyfrowania. Kontrola mechanizmem CORS. Regularne przeprowadzanie audytów bezpieczeństwa w celu identyfikacji i usuwania potencjalnych luk.
3. **Kontrola Dostępu:**
   * **Implementacja:** Wykorzystanie list kontroli dostępu (ACL) do precyzyjnego zarządzania dostępem do różnych funkcji systemu. Umożliwienie administratorom definiowania dostępu dla poszczególnych użytkowników i grup, kontrolując dostęp do danych oraz funkcji na poziomie aplikacji.
4. **Monitorowanie Działalności Użytkowników:**
   * **Implementacja:** Rejestracja i analiza logów w celu monitorowania aktywności użytkowników. Ostrzeżenia lub blokowanie kont w przypadku wykrycia podejrzanej aktywności. Powiadomienia dla administratorów w przypadku nieprawidłowych prób dostępu.
5. **Bezpieczeństwo Sesji:**
   * **Implementacja:** Ustawienie krótkich czasów ważności sesji, stosowanie mechanizmów automatycznego wylogowywania użytkowników po określonym czasie bezczynności. Użycie bezpiecznych mechanizmów przechowywania informacji o sesji, takich jak HttpOnly i Secure cookies.
6. **Zaszyfrowane Przechowywanie Haseł:**
   * **Implementacja:** Zastosowanie biblioteki [werkzeug](https://stackoverflow.com/questions/43028930/werkzeug-generate-password-hash-is-there-any-point) do tworzenia zahashowanej formy haseł do przechowywania w bazie. Regularne przeglądy i aktualizacje polityk dotyczących bezpieczeństwa haseł.

### **Wymagania Wydajnościowe:**

1. **Serwery:**
   * Szacowanie ilości serwerów, uwzględniając przewidywany wzrost liczby użytkowników i danych w systemie.
2. **Łącza Sieciowe:**
   * Zapewnienie odpowiedniej przepustowości łączy sieciowych do płynnego przesyłania danych pomiędzy mikroserwisami.
3. **Optymalizacja:**
   * Regularna optymalizacja kodu i baz danych w celu utrzymania wysokiej wydajności. Stosowanie dobrych praktyk programowania.

### **Wymagania Wydajnościowe (Implementacje):**

1. **Skalowalność Mikroserwisów:**
   * **Implementacja:** Zastosowanie narzędzi i platform do zarządzania kontenerami, takich jak Kubernetes, w celu automatycznego skalowania mikroserwisów w odpowiedzi na zmienne obciążenie. Regularne monitorowanie wydajności poszczególnych mikroserwisów w celu zoptymalizowania ich działania.
2. **Cache’owanie Danych:**
   * **Implementacja:** Wykorzystanie mechanizmów cache'owania dla często używanych danych, aby zminimalizować czas odpowiedzi. Użycie narzędzi takich jak Redis lub Memcached do efektywnego przechowywania cache.
3. **Asynchroniczność:**
   * **Implementacja:** Wdrożenie mechanizmów przetwarzania asynchronicznego, szczególnie w obszarach, gdzie odpowiedź nie jest natychmiastowo potrzebna, np. przetwarzanie danych wejściowych czy wysyłanie powiadomień. Użycie kolejek wiadomości RabbitMQ.
4. **Optymalizacja Bazy Danych:**
   * **Implementacja:** Regularne analizy wydajnościowe bazy danych, optymalizacja zapytań SQL, indeksowanie odpowiednich pól. Użycie narzędzi do monitorowania i analizy wydajności baz danych.
5. **Wielowątkowość:**
   * **Implementacja:** Wdrożenie obsługi wielu wątków w serwerze aplikacyjnym, aby obsługiwać równoczesne żądania. Użycie puli wątków (thread pool) do efektywnego zarządzania zasobami.
6. **Optymalizacja Przesyłania Danych:**
   * **Implementacja:** Minimalizacja ilości przesyłanych danych poprzez selektywne pobieranie tylko niezbędnych informacji.
7. **Testy Obciążeniowe:**
   * **Implementacja:** Regularne przeprowadzanie testów obciążeniowych w środowisku produkcyjnym, aby zidentyfikować potencjalne obszary problematyczne związane z wydajnością i dostosować infrastrukturę lub kod aplikacji w odpowiedzi na wyniki testów.