**Program do weryfikowania, obliczania oraz gromadzenia danych odnośnie spalania pojazdów.**

**Specyfikacja Wymagań projektu**

**Wersja 1.4 opracowana do 3.02.2017**

Historia zmian projektu

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Wersja | Opis | Autor |
| 25.01.17 | 1.1 | Podstawowy zarys projektu: stworzenie szkieletu aplikacji desktopowej dla możliwości wspólnej pracy zespołu nad programem, opartego o Entity Framework | Edgar |
| 29.01.17 | 1.2 | Wprowadzenie funkcjonalności:  1. Interfejs graficzny  2. Możliwość dodawania do bazy samochodów oraz ich właściwości:  - nazwa  - data przeglądu  - rodzaj silnika  3. Możliwość dodawania tankowań (do bazy) dla poszczególnych samochodów  - data  - ilość zatankowanego paliwa  - przebyta odległość | Edgar, Krzysztof, Karol |
| 1.02.17 | 1.3 | Wprowadzenie poprawek:  - Dodanie powiadomienia odnośnie daty kolejnego przeglądu | Krzysztof, Karol |
| 3.02.17 | 1.4 | Finalna wersja | Edgar, Krzysztof, Karol |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Spis treści**

1. Wprowadzenie i cel powstania aplikacji

2.Wymagania biznesowe

3.Wymagania sprzętowe

4.Opis funkcjonalności

5.Wymagania niefunkcjonalne

1. Wprowadzenie i cel powstania aplikacji

Aplikacji do obliczania oraz gromadzenia danych odnośnie zużycia paliwa w pojeździe użytkownika.

2.Wymagania biznesowe

Aplikacja ma za zadanie przetwarzać dostarczane informacje od użytkownika, który poprzez interfejs użytkownika może dostarczyć do bazy informacje na temat zużycia paliwa w swoim samochodzie po przez wprowadzenie wymaganych danych, na podstawie których generowane są statystyki. Dzięki temu użytkownik jest w stanie dowiedzieć się jak kształtuje się zużycie paliwa dla danego pojazdu w określonym przedziale czasowym. Aplikacja jest tworzona z myślą o użytkownikach prywatnych, którzy mogą posiadać więcej niż jeden pojazd.

3.Wymagania sprzętowe

-Komputer z systemem Windows XP lub wyższy

4.Opis funkcjonalności

1. Dodawanie nowego wpisu auta do bazy

|  |
| --- |
| **Wymagania biznesowe** |
| Możliwość dodania auta |
| Podczas dodawania auta użytkownik wprowadza dane:  - nazwa własna (pole obowiązkowe)  - markę pojazdu (pole opcjonalne)  - model (pole opcjonalne)  - rok produkcji (pole opcjonalne)  - pojemność silnika (pole opcjonalne)  - moc silnika (pole opcjonalne)  - wybrać kategorię rodzaju paliwa (benzyna/diesel/gaz) (pole opcjonalne)  - wybrać masę pojazdu (samochód osobowy, dostawczy, bus) (pole opcjonalne) |
| Dodany pojazd po zapisaniu można edytować lub usunąć. |
| Możliwość podglądu zapisanych pojazdów. |

2. Dodawanie nowej stacji paliw do bazy

|  |
| --- |
| **Wymagania biznesowe** |
| Możliwość dodania stacji paliw |
| Podczas dodawania stacji paliw użytkownik wprowadza dane:  - nazwa własna (pole wymagane)  - marka stacji (pole wymagane) |
| Dodaną stację paliw po zapisaniu można edytować lub usunąć. |
| Możliwość podglądu zapisanych stacji paliw. |

3. Dodawanie danych odnośnie zużytego paliwa

|  |
| --- |
| **Wymagania biznesowe** |
| Możliwość dodania wielu pomiarów zużycia paliwa do wskazanego pojazdu |
| Podczas dodawania pomiaru użytkownik wprowadza obowiązkowe dane:  - ilość przejechanych kilometrów  - ilość zatankowanego paliwa  - stacja paliw na której był zatankowany pojazd  - tankowany pojazd  - data tankowania |
| Po dodaniu poszczególnych danych użytkownik ma możliwość wpisania kolejnych danych z innego pomiaru które zapisują się w bazie danych |
|  |

4. Generowanie zestawień i wykresów

|  |
| --- |
| **Wymagania biznesowe** |
| Możliwość wygenerowania zestawień wybranych pojazdów z bazy |
| Wygenerowane zestawienie jest prezentowane w formie diagramów, tabel dzięki którym klient może sprawdzić jak kształtuje się zużycie paliwa dla danego pojazdu w danej przestrzeni czasowej oraz stacji paliw. |

5.Wymagania niefunkcjonalne

Możliwość obsługi programu za pomocą myszki i klawiatury

System powinien działać szybko

System powinien posiadać prosty i przejrzysty układ graficzny

Baza danych powinna być wystarczającą pojemność do przechowania przepisów

System powinien być kompatybilny z systemami wymienionymi w punkcie 2

Obsługa programu powinna być intuicyjna

Program powinien być tak napisany, aby czas niesprawności po awarii był minimalny

System powinien sprawdzać poprawność wprowadzanych danych i informować o błędnych danych i konieczności ich poprawy