# Raport końcowy projektu z programowania obiektowego

**Autor**: Wojciech Pawlik, 265976

**Termin zajęć:** środa, 15:15

## Opis projektu i wstęp teoretyczny

Tematem projektu była aplikacja internetowa, służąca jako pomoc dla analityków E-sportowych gry Counter-Strike: Global Offensive . Aplikacja skupia się w okół różnych kalkulatorów wyliczających różne statystyki m.in. średnie KDA, KDRatio, Cenę zakupów przedmiotów w grze itp. Aplikacja powstała na bazie ASP.NET MVC i została napisana w wysokopoziomowym języku C#. Baza ta implementuje patern Model-View-Controller, który rozdziela logikę biznesową, prezentację danych i obsługę interakcji użytkownika na trzy różne elementy. Model odpowiedzialny jest za reprezentację danych i logiki biznesowej. View czyli widok odpowiedzialny jest za prezentację użytkownikowi danych z modelu w atrakcyjnej formie i nie zawiera logiki biznesowej. Kontroler jest pośrednikiem między Modelem a Widokiem. Przetwarza żądania użytkownika, pobiera dane z Modelu, wykonuje operacje biznesowe oraz przekazuje dane do widoku.

## Przykładowe zdjęcia wyglądu aplikacji

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 1. Strona główna z nawigacją

Obraz zawierający zrzut ekranu, tekst, diagram, design

Opis wygenerowany automatycznie

Rysunek 2. Kalkulator zakupów

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 3. Obliczone zakupy

## Struktura aplikacji

Zgodnie z paternem Model-View-Controler aplikacja została podzielona na odpowiednie modele, widoki oraz kontrolery. W folderze Models znajdują się klasy odpowiedzialne za reprezentację danych używanych w całej aplikacji. W klasach Models została zawarta enkapsulacja poprzez ustawienie setterrów na prywatne. W folderze Models znajduje się podfolder o nazwie Services, w którym przechowywana jest **klasa StatsCalculator**. Napisane są w niej metody dokonujące obliczeń na stworzonych obiektach.

**Klasa CsGoItem**- reprezentuje dane przedmiotów tworzonych w kalkulatorze zakupów. Jest to klasa abstrakcyjna.

**Klasa Weapon oraz Grenade**- są to klasy dziedziczące po CsGoItem. Reprezentują dane konkretnych rodzajów przedmiotów ze sklepu. Posiadają metody nadpisujące metody z klasy CsGoItem. Służą do obliczania ostatecznej ceny przedmiotu oraz przekonwertowanie jej do stringa w celu wyświetlenia z dodatkowymi informacjami z specyficznymi dla poszczególnych klas.

**Klasa Stats**- reprezentuje statystki takie jak zabójstwa, asysty oraz śmierci.

**Klasa Player**- reprezentuje dane graczy utworzonych w porównywarce graczy. Klasa ta dziedziczy po klasie Stats.

**Klasa Rilfer oraz Sniper**- to klasy również dziedziczące po klasie Stats. Reprezentują dane dotyczące konkretnych typów gracza, dla których możemy wyliczyć statystyki w podstronie KalkulatorKDA.

**Klasa RiflerViewModel, PlayerViewModel, SniperViewModel, CsGoItemViewModel**- są to klasy służące do przechowywania danych z widoków.

W katalogu Views znajdują się podfoldery z nazwami odpowiadającymi podstroną, które reprezentują. Każdy podfolder zawiera widoki generowane za pośrednictwem kontrolerów. Widok podzielone są na formularze do uzupełnienia danych oraz na widoki rezultatów generowane po stworzeniu instancji klas oraz dokonaniu na nich odpowiednich obliczeń.

W Katalogu Controllers znajdują się wszystkie kontrolery odpowiadające za przetwarzanie żądań takich jak POST czy GET, pobieranie danych z modelów oraz dokonanie na nich obliczeń. Są one również podzielone odpowiednio do nazw podstron.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Rysunek 4. Struktura plików

## Opis niektórych części kodu

W klasie StatsCalculator znajdują się metody obliczające statystyki takie jak KDRatio, KDA, procent strzałów w głowę, procent pierwszych zabójstw, ilość zabójstw snajperką, ilość zabójstw bez celowania. Obliczenia wykonywane w metodach są bardzo proste. Jako argument przyjmują obiekt odpowiedni do wykonywanych obliczeń(Stats, Sniper, Rifler) i wykonują proste dzielenie. Obsługują przypadki, kiedy podane statystki są niepoprawne. Zaokrąglają wyniki do 2 miejsc po przecinku.

Klasa Weapon oraz Grenade zawierają nadpisane metody z klasy CsGoItem. Pierwsza z nich CalculateTotalPrice() służy do wyliczenia całkowitego kosztu zakupu przedmiotu. Mnoży cenę i ilość przedmiotu. Druga ConvertPriceToString() służy to przekonwertowania ostatecznej ceny do stringa w celu wyświetlenia go z dodatkowymi informacjami. Dla Weapon wyświetla nazwe przedmiotu, ilość amunicji itp.

Kontroler PlayerController w pierwszej kolejności obsługuje wyświetlanie widoku Player. Później wywołuje metodę HttPost, która przetwarza dane przesłane z widoku Player po wysłaniu formularza. Na wejściu przyjmuje tablicę obiektów PlayerViewModel. Jeśli dane wprowadzone w formularzu są poprawne, tworzone są obiekty graczy na podstawie wprowadzonych danych. Następnie porównywane są różne statystki graczy. Wyniki zapisywane są do ViewBagów czyli dynamicznych obiektów służących do przekazywania danych z widoków do kontrolera. Następnie wyniki są przekazywane do widoku FiltredPlayers. W kontrolerze zawarta jest prywatna metoda, która oblicza stosunek KDa do ceny dla danego zawodnika. Metoda FiltredPlayer przyjmuje tablicę obiektów graczy jako argument i zwraca wynik jako widok.

Kontroler ShoppingCalculatorController odpowiedzialny jest za przesył danych potrzebnych do obliczenia zakupów. Na początku wykonuję akcje GET, która wyświetla formularz. Następnie wykonywana jest akcja POST, która obsługuje przesłany formularz. Przyjmuje listę modeli CsGoItemViewModel, które reprezentują przedmioty CS:GO wprowadzone przez użytkownika. W zależności od typu przedmiotu (Weapon lub Grenade) tworzy obiekty klasy Weapon lub Grenade. Następnie oblicza całkowitą cenę przedmiotów i przypisuje ją do ViewBag. Tworzy listę przedmiotów (items), do której dodaje utworzone obiekty broni lub granatów i oblicza całkowitą cenę wszystkich przedmiotów przypisując ją do ViewBag. Na końcu przekierowuje do widoku CalculatedPurchase, przekazując listę przedmiotów jako model.

Widok ShoppingCalculator odpowiedzialny jest za wyświetlenie formularza kalkulatora zakupów. Na początku widzimy deklarację modelu, który wskazuje na to, że widok oczekuje na listę obiektów typu CsGoItemViewModel jako model. @using (Html.BeginForm…) rozpoczyna formularz, którego dane zostaną przesłane do akcji ShoppingCalculator. Kontrolera ShopingCalculatorController za pomocą metody POST. Następnie pętla for generuje pola dla 6 przedmiotów. W kodzie został zawarty fragment napisany w JavaScripcie. Jest on napisany w celu obsługi zmiany typu przedmiotu i odpowiednio wyświetla pola specyficzne dla broni lub granatów.