ZASADY TWORZENIA DOBREGO KODU

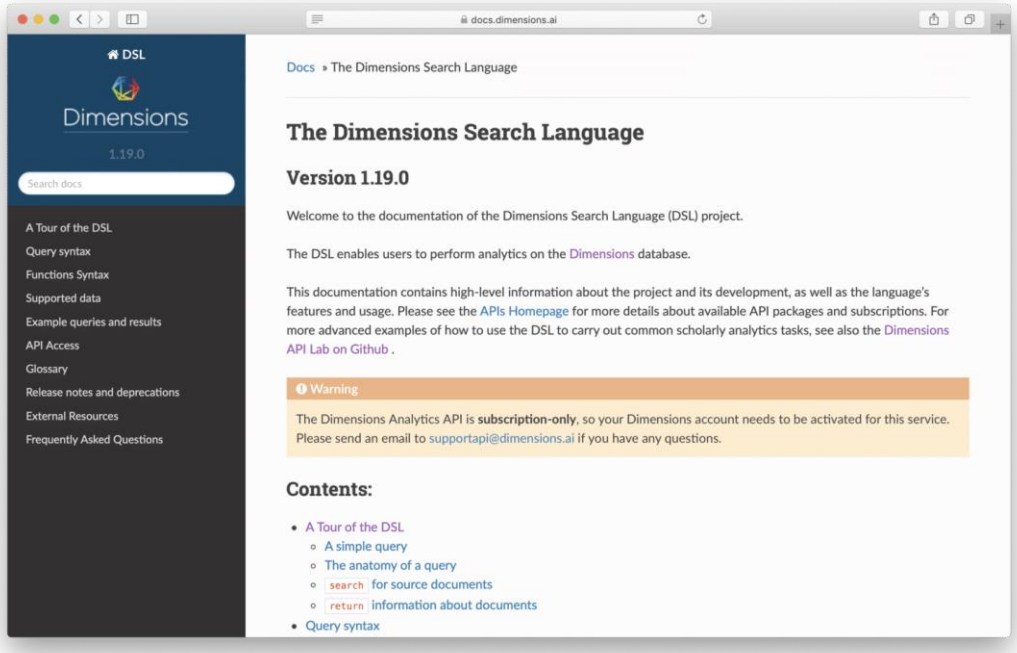
PODSTAWY PROGRAMOWANIA

DOBRY KOD, WIĘKSZA GWARANCJA NA SUKCES

- Musimy pamiętać o tym, aby nasze projekty utrzymywać w taki sposób, aby kod, który zapisujemy być zrozumiały, czytelny oraz łatwy do utrzymania dzięki czemu, ani my sami, ani nasz zespół nie będzie miał problemu z pracą nad projektem. To nie tylko opanowanie języka programowania, ale także o zdobycie umiejętności myślenia logicznie, rozwiązywania problemów i współpracy zespołowej.

O CZYM MUSIMY PAMIĘTAĆ PODCZAS TWORZENIA DOBREGO KODU?

- Przed tworzeniem dobrego kodu, zaprojektuj na sam początek dokumentacje na temat tworzonego kodu. Zastanów się, co będzie wykonywał Twój kod i jakiej technologii będzie wymagał kod. Warto jest zapisać swój pomysł lub opis naszego kodu w dokumentacji, aby móc trzymać się tego, co chcemy stworzyć. Może być to w postaci dokumentu PDF lub nawet aktualizowanej strony Wiki.



CZYTELNOŚĆ KODU TO PODSTAWA DOBREGO KODU

- Pamiętaj, że tworząc zmienne dobrze jest nazywać je odpowiednio, jak np. `***liczba_porzadkowa***`czy `***wartosc_zamowienia***` To zdecydowanie jest bardziej zrozumiałe niż `***x1***`, `***liczba2***`. Dodawanie komentarzy do kodu również ułatwia nam oraz zespołowi pracę nad kodem

```
1    public string nazwa_uzytkownika;
2    public int poziom_gracza;
3    public int poziom_zdrowia;
4    public int poziom_glodu;
5    public Item[] ekwipunek;
6    public double[] koordynaty = new double[2];
```

ZASADY CZYSTEGO KODU - KISS

- Zasada KISS (**Keep It Simple, Stupid**) jest zasadą projektowania, która mówi nam, że rozwiązania powinny być proste, zamiast skomplikowanych. Prostota jest kluczowa dla zrozumienia, utrzymania i rozwijania kodu. KISS zachęca do unikania nadmiernego skomplikowania projektów czy kodu, co może prowadzić do trudności w zrozumieniu, debugowaniu i utrzymaniu kodu w przyszłości.

```
1   using System;
2
    Odwołania: 0
3   public class Calculator
4   {
        Odwołania: 0
5       public int Add(int a, int b) {
6           return a + b;
7       }
8
        Odwołania: 0
9       public int Subtract(int a, int b) {
10          return a - b;
11      }
12
        Odwołania: 0
13      public int Multiply(int a, int b) {
14          return a * b;
15      }
16
        Odwołania: 0
17      public double Divide(int a, int b) {
18          if (b != 0)
19              return (double)a / b;
20          else
21          {
22              Console.WriteLine("Nie można dzielić przez 0!");
23              return double.NaN; // NaN (Not a Number) to specjalna wartość reprezentująca brak wartości numerycznej
24          }
25      }
26  }
```

ZASADY CZYSTEGO KODU - DRY

- DRY (Don't Repeat Yourself) to kolejna ważna zasada w programowaniu. Oznacza ona, że w kodzie źródłowym nie powinno się powtarzać tego samego fragmentu kodu. Zamiast tego, powtarzające się fragmenty kodu powinny być zrefaktoryzowane do jednej wspólnej funkcji, aby unikać redundancji i ułatwić późniejsze zmiany.

```
1   using System;
2
    Odwołania: 4
3   public class Product
4   {
        Odwołania: 4
5       public string Name { get; set; }
        Odwołania: 4
6       public decimal Price { get; set; }
        Odwołania: 3
7       public decimal VatRate { get; set; }
8
        Odwołania: 2
9       public decimal CalculatePriceWithVat()
10      {
11          decimal totalPrice = Price + (Price * VatRate);
12          return totalPrice;
13      }
14  }
15
    Odwołania: 0
16  class Program
17  {
        Odwołania: 0
18      static void Main()
19      {
20          Product product1 = new Product { Name = "Laptop", Price = 1000, VatRate = 0.23 };
21          Product product2 = new Product { Name = "Smartphone", Price = 500, VatRate = 0.23 };
22
23          // Obliczanie ceny z podatkiem za pomocą metody CalculatePriceWithVat()
24          decimal totalPrice1 = product1.CalculatePriceWithVat();
25          decimal totalPrice2 = product2.CalculatePriceWithVat();
26
27          Console.WriteLine($"Total price for {product1.Name}: {totalPrice1:C}");
28          Console.WriteLine($"Total price for {product2.Name}: {totalPrice2:C}");
29
30          Console.ReadLine();
31      }
32  }
```

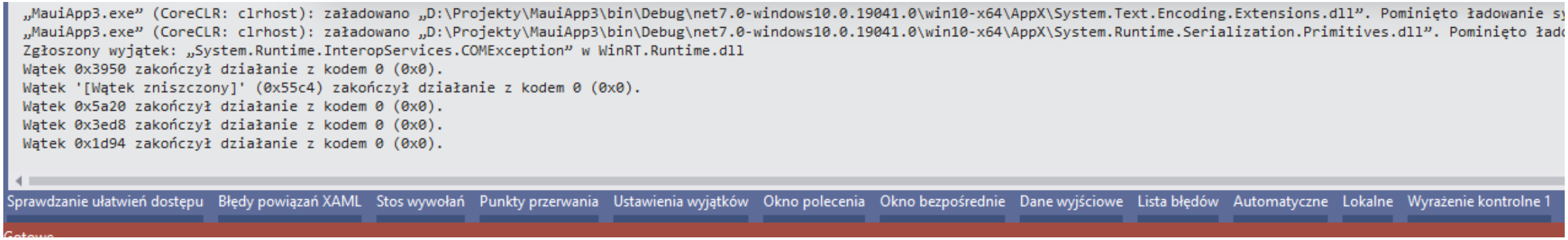
OBSŁUGA WYJĄTKÓW I BŁĘDÓW

- Obsługa wyjątków i błędów jest kluczowym aspektem programowania, który pozwala na zarządzanie sytuacjami, które mogą wystąpić podczas wykonywania kodu. Wyjątki to sytuacje, które mogą prowadzić do nieoczekiwanych zachowań programu, takie jak dzielenie przez zero czy próba dostępu do nieistniejącego pliku. Obsługa błędów pozwala na kontrolowane reagowanie na te sytuacje, zamiast po prostu zatrzymywania działania programu. Są dwa rodzaje wyjątków. **Kontrolowane i niekontrolowane**.

```
1  try
2  {
3      // Sprawdzamy czy użytkownik jest pełnoletni
4      if (age < 18)
5      {
6          throw new Exception("User is underage");
7      }
8  }
9  catch (Exception ex)
10 {
11     Console.WriteLine("Error: " + ex.Message);
12 }
```

POZBĄDŹ SIĘ „ROBALI” DLA DOBRA KODU

- Debugowanie kodu jest kluczowym procesem podczas tworzenia oprogramowania. Pozwala ono na identyfikację i naprawę błędów (bugs) w kodzie źródłowym, co przyczynia się do poprawy jakości aplikacji. Możemy wyłapywać je na wiele sposobów, jak np.: Wyświetlanie w konsoli wartości (**Console.WriteLine**), kontrolowanie zmiennych w debugerze, czytanie logów programu, kolaboracja zespołowa



STRUKTURY DANYCH I ALGORYTMY

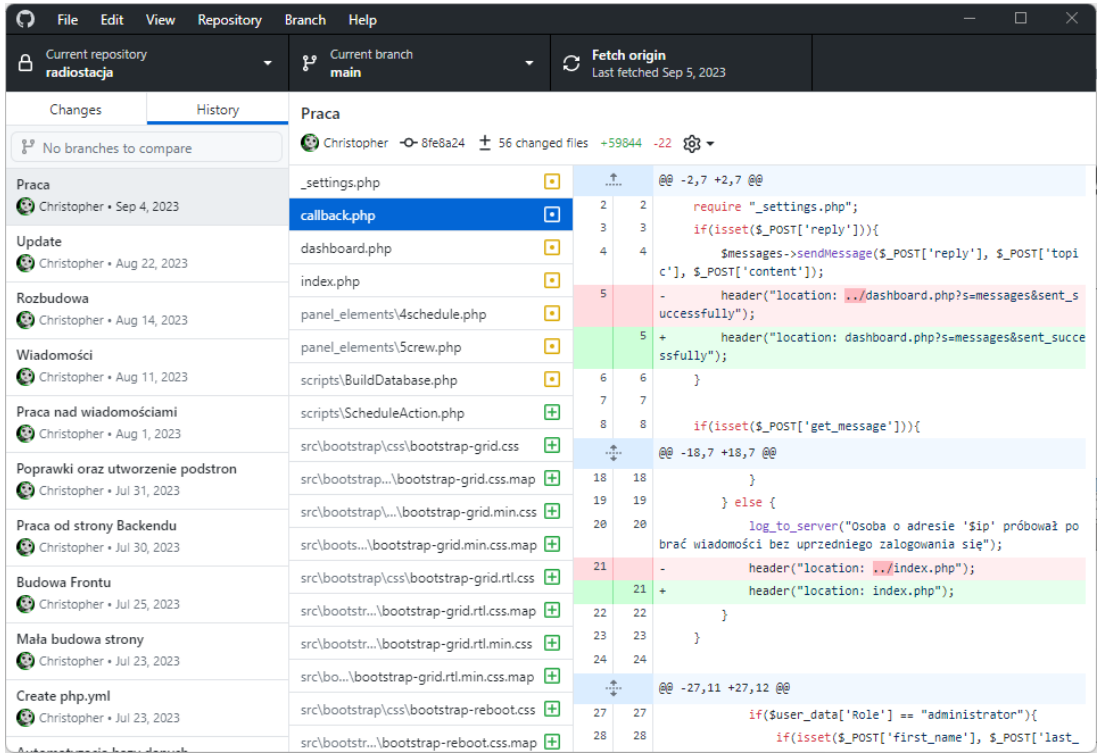
- Zasadą tworzenia dobrego kodu jest również znajomość algorytmów oraz struktur danych, aby w łatwiejszy oraz czystszy sposób móc zapisywać fragmenty kodu, które nie wymagają dużej ilości czasu

- Przykładowe struktury danych:
  - Listy: List<type>
  - Kolejki: Queue<type>
  - Stosy: Stack<type>
  - Słowniki: Dictionary<type, type>

- Przykładowe algorytmy sortowania, wyszukiwania:
  - Sortowanie bąbelkowe
  - Sortowanie przez wstawiania
  - Wyszukiwanie liniowe

IM WIĘCEJ OSÓB TYM... BLIŻEJ DO BAŁAGANU

- Wiadomo, dobrze jest pracować wspólnie nad projektem, aczkolwiek jeżeli mamy bardzo dużą ilość osób przy projekcie to przy nieodpowiedniej organizacji zespołu możemy spowodować bałagan w pracy, przez co czystość kodu może być bliska zeru. Dlatego dobrze jest rozplanować swoją prace.
- Dobrze jest również korzystać z systemów kontroli wersji np. Git dzięki czemu bez żadnych problemów możemy pracować w kilka osób jednocześnie

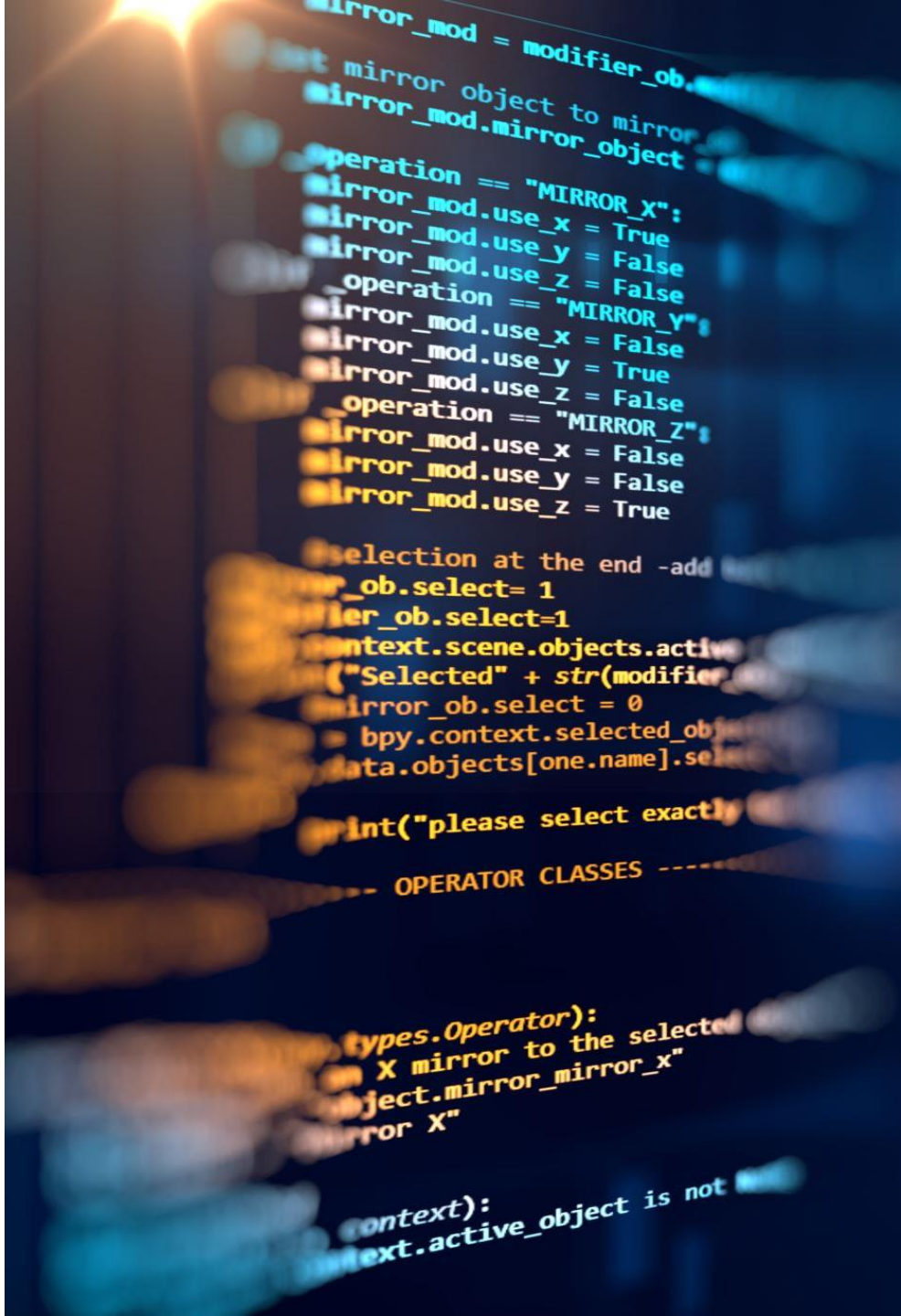


BEZPIECZEŃSTWO PRZEDE WSZYSTKIM

- Zanim wypuścisz swój kod w strefę Internetu, upewnij się, że nie zawiera ona luk, które może spowodować niebezpieczeństwo dla Twojego serwera np. poprzez SQL Injection. Pamiętaj więc o poprawnej walidacji oraz oczyszczania danych, aby nie zawierała niebezpiecznych znaczków, haszowaniu danych za pomocą „soli”. Salt sprawia, że nawet dwa identyczne hasła będą miały inne hasze. Ciągle też aktualizuj swoje zabezpieczenia i kontroluj zdarzenia z kodu.

JEŻELI KOD U CIEBIE DZIAŁA, TO NIE ZNACZY, ŻE ZADZIAŁA U KLIENTA…

- Niestety smutna prawda. Aby upewnić się, że nasz kod działa poprawnie musimy zadbać o optymalizacje kodu oraz testowanie kodu. Możemy zautomatyzować testowanie tworząc **testy jednostkowe** *(Polega na testowaniu indywidualnych komponentów w izolacji od reszty systemu)*, **testy integracyjne** *(testowanie interakcji pomiędzy różnymi komponentami lub modułami systemu),* **testy akceptacyjne** *(Polega na testowaniu całego systemu pod kątem spełnienia wymagań biznesowych i oczekiwań klienta)*, **testy wydajnościowe** *(Ocena wydajności systemu podczas różnych obciążeń, zapytań lub operacji)* oraz **testy bezpieczeństwa** *(Testowanie systemu pod kątem podatności na ataki i zagrożenia bezpieczeństwa).*



ETYKA RÓWNIEŻ OBOWIĄZUJE W PROGRAMOWANIU

1. **Prywatność**
   - Niezależnie od tego, czy piszesz kod dla aplikacji mobilnych, internetowych czy desktopowych, zawsze powinieneś respektować prywatność użytkowników. Nie powinieneś zbierać ani przechowywać prywatnych danych użytkowników bez ich zgody. Jeśli musisz zbierać dane, powinieneś jasno poinformować użytkowników o celu zbierania danych i ich wykorzystaniu.
2. **Bezpieczeństwo danych**
   - Zapewnij bezpieczeństwo danych, z którymi masz do czynienia. Używaj odpowiednich metod szyfrowania, zabezpieczaj hasła, unikaj trzymania poufnych danych w tekście jawnym, a także dbaj o zabezpieczenie przed atakami typu SQL Injection czy Cross-Site Scripting (XSS).
3. **Prawo autorskie**
   - Nie używaj kodu źródłowego innych programistów bez ich zgody, chyba że kod jest udostępniony jako otwarte źródło (open source) i stosujesz się do zasad licencji. Unikaj plagiatu i szanuj prawa autorskie innych twórców.
4. **Dostępność i Właśnie Społeczne**
   - Twórz oprogramowanie, które jest dostępne dla wszystkich użytkowników, włączając osoby niepełnosprawne. To oznacza, że Twoje aplikacje powinny być łatwe w obsłudze dla osób z różnymi rodzajami niepełnosprawności. Twórz technologie, które poprawiają jakość życia i przynoszą korzyści społeczeństwu

JEST ETYKA, JEST I PRAWO

1. **Przestrzeganie Licencji Oprogramowania**
   - Pamiętaj, że korzystając z oprogramowania, musisz przestrzegać zasad i warunków zawartych w licencji. Niewłaściwe korzystanie z oprogramowania może prowadzić do konsekwencji prawnych
2. **Ochrona Danych Osobowych**
   - W przypadku, gdy Twoja aplikacja zbiera lub przetwarza dane osobowe, musisz przestrzegać obowiązujących przepisów dotyczących ochrony danych osobowych, takich jak ogólne rozporządzenie o ochronie danych (GDPR) w Unii Europejskiej.
3. **Bezpieczeństwo Sieciowe**
   - Unikaj działań, które mogą być uznane za próby hakerstwa, ataków DoS czy naruszenia bezpieczeństwa sieci. Przestrzegaj przepisów dotyczących cyberbezpieczeństwa.
4. **Zgodność ze Standardami Branżowymi**
   - Jeśli tworzysz oprogramowanie dla określonej branży (np. opieka zdrowotna, finanse), musisz przestrzegać standardów i regulacji obowiązujących w tej dziedzinie. Niewłaściwe działanie może prowadzić do poważnych konsekwencji prawnych.

PRZYJAŹNIE DLA UŻYTKOWNIKA

- Przy projektowaniu aplikacji, pamiętaj aby była ona przyjazna dla użytkowników, czyli prosta w obsłudze, ale z przejrzystym interfejsem. Najlepiej jest zbierać opinie oraz poznać zdania użytkowników, co chcieliby zmienić, czy wprowadzić do aplikacji. Pamiętaj, dla kogo to tworzysz

ROZWÓJ ZAWODOWY TO TEŻ PODSTAWA DOBREGO KODU

- Nie zatrzymuj się w miejscu. Poznawaj nowości technologiczne związane z Twoim środowiskiem czy zapoznaj się z nową przestrzenią. Korzystaj z różnych źródeł, takich jak GitHub, StackOverflow czy uczestnicz na różnych konferencjach technologicznych lub na hackathonach.