CODE:ME
UCZYMY PROGRAMOWANIA

{256}

TYDZIEŃ PROGRAMISTY

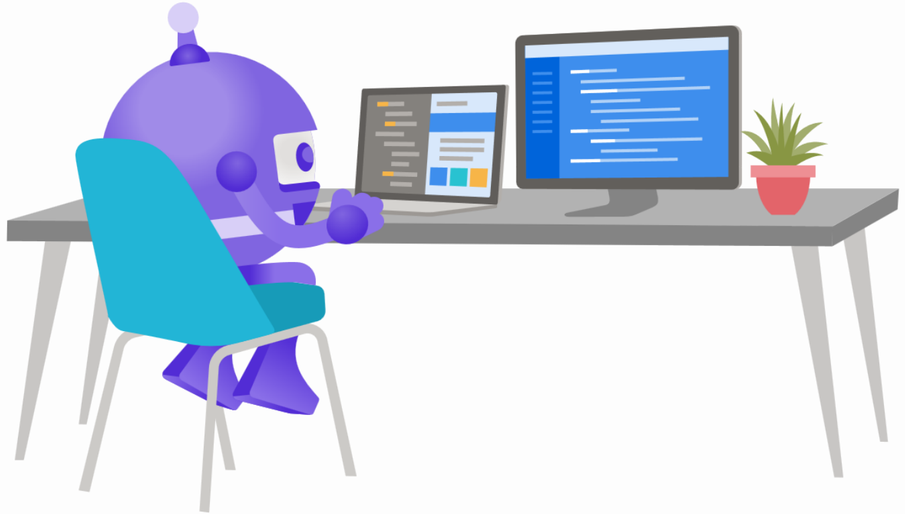
WRZESIEŃ 2022

CODE:ME
UCZYMY PROGRAMOWANIA

{256}
TYDZIEŃ PROGRAMISTY

Wprowadzenie do C#

Stwórzmy wspólnie pierwszą aplikację!



06.09.2022

CODE:ME
UCZYMY PROGRAMOWANIA

{256}
TYDZIEŃ PROGRAMISTY

Materiały z prezentacji

https://github.com/mechelewskim/warsztat-csharp-2022-09-06

CODE:ME
UCZYMY PROGRAMOWANIA

{256}
TYDZIEŃ PROGRAMISTY

Kilka słów o mnie

Mateusz Mechelewski

- programista C#
- absolwent Politechniki Warszawskiej
- aplikacje webowe
- branża finansowa

https://www.linkedin.com/in/mateusz-mechelewski/

CODE:ME
UCZYMY PROGRAMOWANIA

{256}
TYDZIEŃ PROGRAMISTY

Plan prezentacji

1. Dlaczego C#?

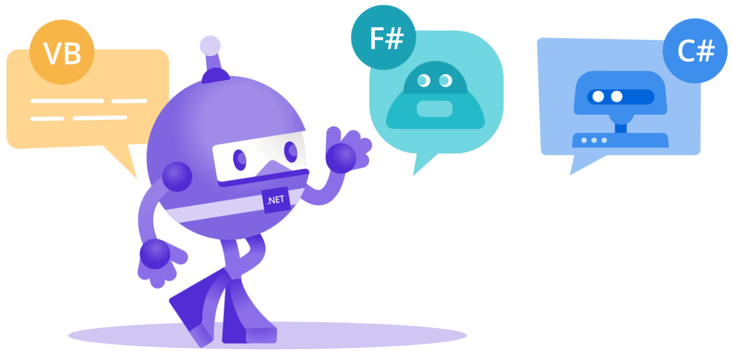
2. Podstawy C#

3. Stwórzmy aplikację!

4. Co dalej?

Platforma dotnet

- Środowisko uruchomieniowe
  - Windows
  - Linux
  - macOS



Kiedyś

.NET FRAMEWORK

- opracowany przez Microsoft
- wydany w 2002 roku
- współpracujący jedynie z systemem Windows
- niedostępny kod źródłowy
- nierekomendowany dla nowych aplikacji

20 LAT
PLATFORMY DOTNET

Teraz

DOTNET (.NET CORE)

- opracowany przez Microsoft wraz ze społecznością (.NET Foundation)
- wydany w 2016 roku
- współpracujący z systemami Windows, Linux oraz macOS
- otwarty kod źródłowy (https://github.com/dotnet)
- rekomendowany dla nowych aplikacji
- rozwijany na podstawie konsultacji ze społecznością

Dlaczego dotnet?

1. **Otwartość**

   Darmowa platforma, dostęp do kodów źródłowych

2. **Zastosowania**

   Twórz aplikacje dla Windows, Linux, macOS, Android, iOS, web

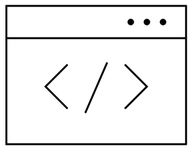
3. **Społeczność**

   Top 5 języków na GitHubie, ponad 1.5 mln programistów

4. **Wydajność**

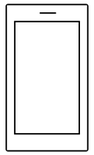
   Społeczność skupiona na poprawie wydajności

Zastosowania



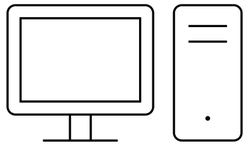
WEB

ASP.NET Core
Blazor



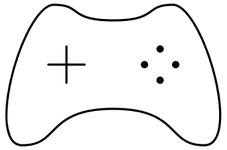
MOBILE

.NET MAUI
Xamarin



DESKTOP

.NET MAUI
WPF / WinForms

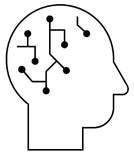


GAMING

Unity

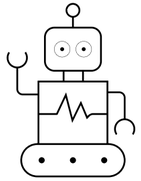
CLOUD

Azure



Artificial Intelligence

ML.NET



Internet of Things

ARM32/64

Zastosowania komercyjne

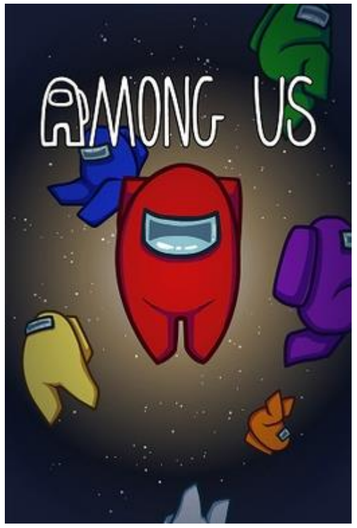
Microsoft Bing

Azure
Active Directory

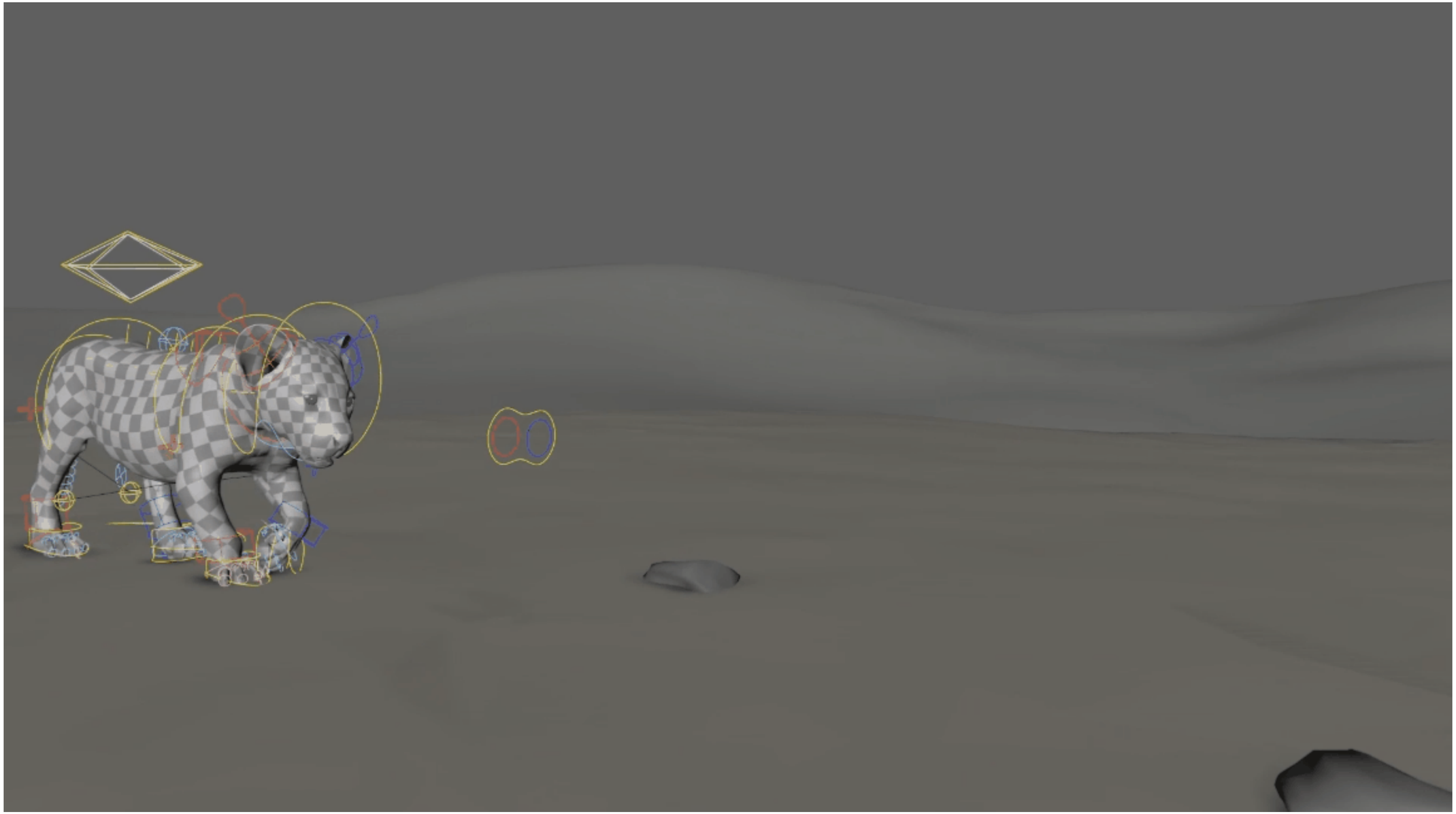
https://dotnet.microsoft.com/platform/customers

Unity jako silnik graficzny

Unity®



https://dotnet.microsoft.com/apps/games
https://unity.com/madewith



Azure jako obsługa trybów multiplayer

Microsoft Flight Simulator

ML.NET - sztuczna inteligencja

Sentiment analysis

Analyze the sentiment of customer reviews using a binary classification algorithm.

Product recommendation

Recommend products based on purchase history using a matrix factorization algorithm.

Price prediction

Predict taxi fares based on parameters such as distance traveled using a regression algorithm.

Customer segmentation

Identify groups of customers with similar profiles using a clustering algorithm.

Object detection

Recognize objects in an image using an ONNX deep learning model.

Fraud detection

Detect fraudulent credit card transactions using a binary classification algorithm.

Sales spike detection

Detect spikes and changes in product sales using an anomaly detection model.

Image classification

Classify images (for example, broccoli vs. pizza) using a TensorFlow deep learning model.

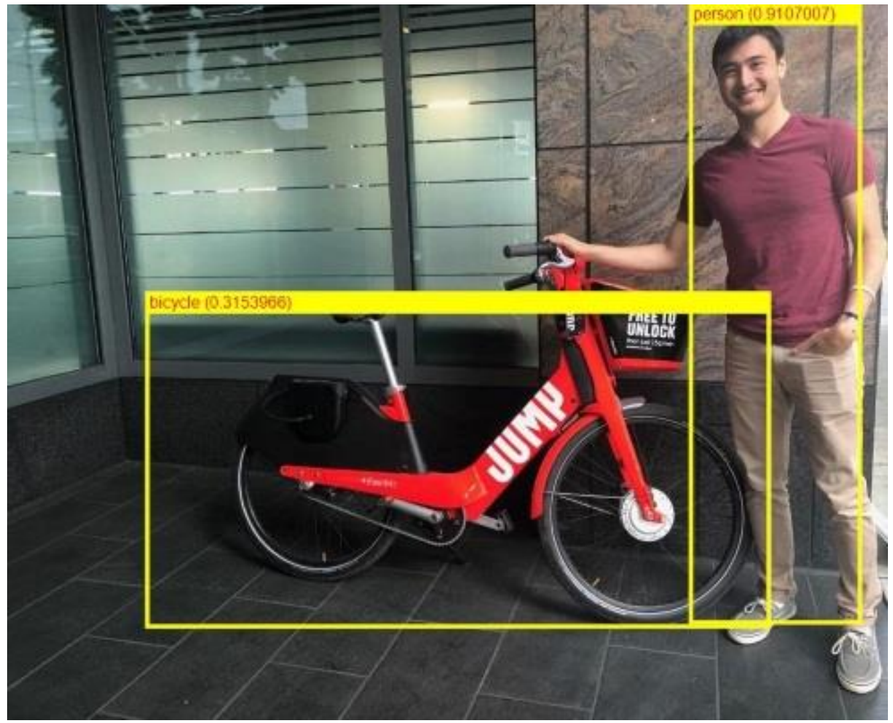
Sales forecasting

Forecast future sales for products using a regression algorithm.

ML.NET - sztuczna inteligencja

FACILITY MANAGEMENT

Microsoft Defender



Power BI

SCANCAM
scancam.com.au

Zaczynamy!

Visual Studio

Visual Studio Code

Visual Studio for Mac

JetBrains Rider

Darmowe do użytku niekomercyjnego
https://visualstudio.microsoft.com
Windows, macOS, Linux

Płatny*
https://jetbrains.com/rider
Windows, macOS, Linux

Przydatne narzedzia

.NET Interactive Notebooks:

https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.dotnet-interactive-vscode

dotnet fiddle:

https://dotnetfiddle.net

Typy zmiennych

C# to język:

- silnie typowany (jawne wskazanie typu każdej zmiennej)
- kompilowany (weryfikacja błędów przed wydaniem nowej wersji)

```
1 int myNum = 5;                    // Liczby całkowite
2 long myBigNum = 50000;            // Duże liczby całkowite (kiedy int nie jest wystarczający)
3 double myDoubleNum = 5.99;        // Liczba rzeczywista (niska precyzja obliczeń)
4 decimal myDecimalNum = 5.99m;     // Liczba rzeczywista z wysoką precyzją (pieniądze)
5 bool myBool = true;               // Wartość logiczna (prawda lub fałsz)
6 char myLetter = 'D';              // Pojedynczy znak
7 string myText = "Hello";          // Wartość tekstowa
8 var stringVariable = "Hello";     // Wartość tekstowa (na podstawie czasu kompilacji)
9 var intVariable = 5;              // Liczba całkowita (na podstawie czasu kompilacji)
```

Formatowanie napisów

```
 1 string a = "Hello";
 2 string b = "World";
 3
 4 string c = a + " " + b;                  //Hello World
 5 string d = $"Program mówi: {a} {b}";     //Program mówi: Hello World
 6 string e = " Hello World ";
 7 string f = e.Trim();                     //Hello World
 8 string g = c.Substring(6);               //World
 9 string h = c.Substring(0, 5);            //Hello
10 string i = a.ToUpper();                  //HELLO
11 string j = a.ToLower();                  //hello
12
13 int length = a.Length;              //5
14 int index1 = a.IndexOf('o');        //4
15 int index2 = a.IndexOf('l');        //2
16 int index3 = a.LastIndexOf('l');    //3
17
18 bool start = a.StartsWith('H');     //true
19 bool end   = a.EndsWith('o');       //true
```

Operacje w konsoli

```
1 Console.BackgroundColor = ConsoleColor.Magenta; //kolor tła
2 Console.ForegroundColor = ConsoleColor.Yellow;  //kolor tekstu
3
4 Console.WriteLine("Hello World!");              //wypisanie linii napisu
5 string input = Console.ReadLine();              //odczytanie napisu
6 Console.WriteLine(input);                       //wypisanie linii napisu
7
8 int number = int.Parse(input);                  //zamiana tekstu na liczbę
```

Operacje na liczbach

```
 1 int a = 5;
 2 int b = 6;
 3
 4 int c = a + b;              //11
 5 int d = a - b;              //-1
 6 int e = a * b;              //30
 7 int f = a / b;              //uwaga: 0 (dzielenie całkowite)
 8 int g = a % b;              //5 (reszta z dzielenia)
 9 int h = a ^ b;              //uwaga: operacja bitowa
10 double i = Math.Pow(a, b); //15625 (potęgowanie)
11
12 double j = (double) b;
13
14 double k = a / g;           //0.8333333333333334
15 double l = (double) a / b; //0.8333333333333334
```

Operacje na liczbach

```
int a = int.MaxValue;    // 2147483647
int b = int.MinValue;    //-2147483648

int c = a + 1;           //-2147483648

long d = long.MaxValue;  // 9223372036854775807
long e = long.MinValue;  //-9223372036854775808

long f = a;
int g = d; //error: Cannot implicitly convert type 'long' to 'int'.
```

Instrukcje warunkowe

```
 1 int age = 17;
 2
 3 if (age >= 18)
 4 {
 5     // więcej lub równe 18 lat
 6 }
 7 else if (age > 12)
 8 {
 9     // mniej niż 18 lat
10     // i więcej niż 12 lat
11 }
12 else
13 {
14     // wszystkie pozostałe przypadki
15 }
16
17 bool condition = true; // jawne przypisanie wartości true lub false
18 condition = age >= 21; // warunek logiczny jako wartość bool
19
20 if (condition)
21 {
22     // warunek spełniony
23 }
```

Kolekcje

```
int[] numbers = new int[] { 0, 1, 2, 3 };                //tablica liczb
string[] texts = new string[] { "abc", "def", "ghi" };   //tablica napisów

int number = numbers[0]; //0
numbers[1] = 5;          //numbers = { 5, 1, 2, 3 }

List<int> listOfNumbers = new List<int>();
listOfNumbers.Add(0);
listOfNumbers.Add(1);
listOfNumbers.Add(2);
listOfNumbers.Remove(1);

List<string> listOfTexts = new()
{
    "abc",
    "def",
    "ghi"
};

Dictionary<string, int> dictionary = new()
{
    { "Opel", 1 },
    { "Audi", 2 }
};

int opel = dictionary["Opel"]; //1
```

Pętle

```
1 //Pętla for
2 for (int i = 0; i < 3; i++)
3 {
4     Console.WriteLine("Liczba to: " + i);
5 }
6
7 //Liczba to: 0
8 //Liczba to: 1
9 //Liczba to: 2
10
11
12 //Pętla foreach
13 int[] numbers = new int[] { 0, 1, 2, 3 };
14
15 int sum = 0;
16
17 foreach (var number in numbers)
18 {
19     sum += number;
20 }
21
22 Console.WriteLine("Suma cyfr z tablicy to: " + sum);
23 //Suma cyfr z tablicy to: 6
```

CODE:ME
UCZYMY PROGRAMOWANIA

{256}
TYDZIEŃ PROGRAMISTY

Klasy i metody

```
 1 public class Person
 2 {
 3     public DateTime Birthday { get; set; }
 4
 5     public string FirstName { get; set; }
 6
 7     public string LastName { get; set; }
 8
 9     public int GetAge()
10     {
11         var today = DateTime.Today;
12         var age = today.Year - Birthday.Year;
13         return age;
14     }
15 }
```

Klasy i metody

```
1 var person = new Person
2 {
3     FirstName = "Marek",
4     LastName = "Nowak",
5     Birthday = new DateTime(2004, 12, 01)
6 };
7
8 int age = person.GetAge();
9
10 Console.WriteLine(age); //18
```

Klasy i metody

```
 1 public class Person
 2 {
 3     public DateTime Birthday { get; set; }
 4
 5     public string FirstName { get; set; }
 6
 7     public string LastName { get; set; }
 8
 9     public int GetAge()
10     {
11         var today = DateTime.Today;
12         var age = today.Year - Birthday.Year;
13
14         // Go back to the year in which the person was born in case of a leap year
15         if (Birthday.Date > today.AddYears(-age))
16         {
17             age -= 1;
18         }
19
20         return age;
21     }
22 }
```

LINQ

```
 1 public class Person
 2 {
 3     public int NumberOfChildren { get; set; }
 4
 5     public string Name { get; set; }
 6 }
 7
 8 Person[] people =
 9     {
10         new Person { NumberOfChildren = 2, Name = "Nowak" },
11         new Person { NumberOfChildren = 2, Name = "Kowalski" },
12         new Person { NumberOfChildren = 0, Name = "Basiak" }
13     };
14
15 var sortedPeople = people
16     .OrderBy(x => x.NumberOfChildren)
17     .ThenBy(x => x.Name);
18
19 foreach (var person in sortedPeople)
20 {
21     Console.WriteLine($"{person.Name} ({person.NumberOfChildren})");
22 }
23
24 //Basiak (0)
25 //Kowalski (2)
26 //Nowak (2)
```

LINQ

```
 1 public class Person
 2 {
 3     public int NumberOfChildren { get; set; }
 4
 5     public string Name { get; set; }
 6 }
 7
 8 Person[] people =
 9     {
10         new Person { NumberOfChildren = 2, Name = "Nowak" },
11         new Person { NumberOfChildren = 2, Name = "Kowalski" },
12         new Person { NumberOfChildren = 0, Name = "Basiak" }
13     };
14
15 var parents = people
16     .Where(x => x.NumberOfChildren > 0)
17     .OrderBy(x => x.Name);
18
19 foreach (var person in parents)
20 {
21     Console.WriteLine($"{person.Name} ({person.NumberOfChildren})");
22 }
23
24 //Kowalski (2)
25 //Nowak (2)
```

Obsługa wyjątków

```
1 int[] numbers = new int[] { 0, 1, 2, 3 };
2
3 int a = numbers[4]; //System.IndexOutOfRangeException: Index was outside the bounds of the array.
4
5 int a = 0;
6 int b = 2 /0; //System.DivideByZeroException: Attempted to divide by zero.
```

Obsługa wyjątków

```
try
{
    string input = Console.ReadLine();
    int a = int.Parse(input);

    int b = 2 / a;
}
catch (FormatException)
{
    Console.WriteLine("Wprowadzona nie została liczba");
}
catch (DivideByZeroException)
{
    Console.WriteLine("Dzielenie przez zero");
}
catch (Exception e)
{
    Console.WriteLine("Wystąpił inny błąd " + e.Message);
}
```

Obsługa wyjątków

Dobre praktyki obsługi wyjątków:

- umieszczanie potencjalnego kodu w bloki try-catch
- obsługa w formie wyjątków jedynie nadzwyczajnych sytuacji
- używanie wyjątków zamiast zwracania kodów błędów
- używanie najlepiej pasującego typu wyjątku

Obsługa wyjątków jest procesem o dużej złożoności czasowej.

Gotowe biblioteki (NUGET)

Pakiety NuGet:

- nie musisz pisać wszystkiego samemu
- wykorzystaj gotowe biblioteki (zaufane, ze wsparciem)
- łatwie dzielenie się kodem z innymi programistami

Wykorzystaj zewnętrzne biblioteki na przykład do:

- formatowania wyglądu konsoli
- wysyłki wiadomości e-mail
- połączenia aplikacji z innymi aplikacjami

Gotowe biblioteki (NUGET)

```using FluentEmail.Core;

var email = Email
    .From("from@mail.com")
    .To("to@mail.com", "Bob")
    .Subject("Hows it going, Bob")
    .Body("Yo Bob, long time no see!")
    .Send();

Console.WriteLine(email.Successful);
```

https://www.nuget.org/stats/packages

DEMO

Aplikacja konsolowa

CODE:ME
UCZYMY PROGRAMOWANIA

{256}
TYDZIEŃ PROGRAMISTY

Stwórzmy aplikację!

Co dalej?

Oficjalne materiały szkoleniowe:

https://dotnet.microsoft.com/learn

https://docs.microsoft.com/users/dotnet/collections/yz26f8y64n7k07?WT

C# for begginers:

https://www.youtube.com/playlist?list=PLdo4fOcmZ0oVxKLQCHpiUWun7vlJJvUiN

Kanały Youtube:

https://www.youtube.com/c/Elfocrash

https://www.youtube.com/c/NDCConferences

Roadmapa rozwoju

Programista dotnet:

https://github.com/Elfocrash/.NET-Backend-Developer-Roadmap

Architekt oprogramowania:

https://github.com/justinamiller/SoftwareArchitect

Certyfikaty Microsoft

Poprzednia ścieżka egzaminacyjna:

https://www.seattlepro.com/wp-content/uploads/docs/certification/MSL_Commercial_Certification_Roadmap_2015_03_04.pdf

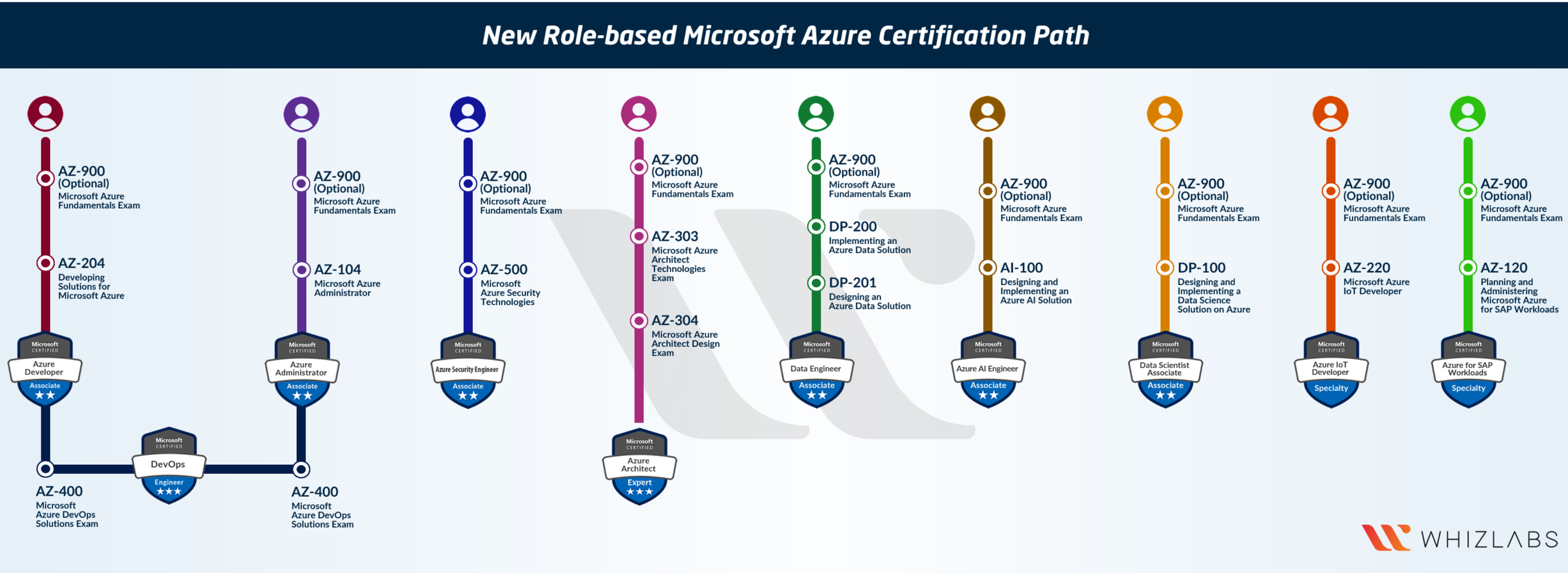
Ścieżka zdecydowanie nierekomendowana do rozpoczęcia

Zalety certyfikatów dla pracowników i pracodawcy:
https://www.altkomakademia.pl/baza-wiedzy/blog/zalety-certyfikatow-microsoft-mcse-mcsd-dla-pracownikow-pracodawcy/

CODE:ME
UCZYMY PROGRAMOWANIA

{256}
TYDZIEŃ PROGRAMISTY

Certyfikaty Microsoft



WHIZLABS

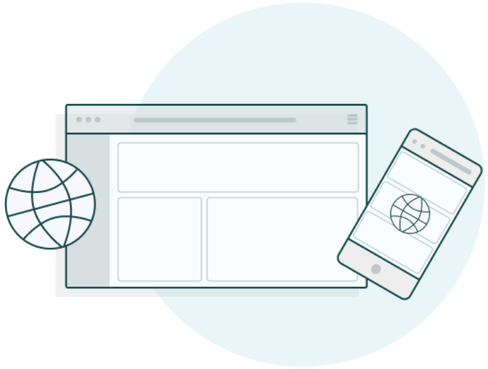
Aktualne trendy

- migracja istniejących aplikacji z .NET Framework do .NET
- wykorzystanie .NET 6 (ze względu na długoterminowe wsparcie LTS)
- rozwiązania chmurowe
- SaaS jako model biznesowy
- wsparcie dla Dockera, Kubernetes
- Infrastructure as Code + przenośność aplikacji
- Blazor oraz .NET MAUI

https://enlear.academy/top-9-net-trends-to-dominate-in-2022-65ea14d24f56

WEB czy desktop?

ASP.NET Core



Web apps

Cross platform

Access anywhere

No installation

VS.

.NET MAUI

Desktop apps

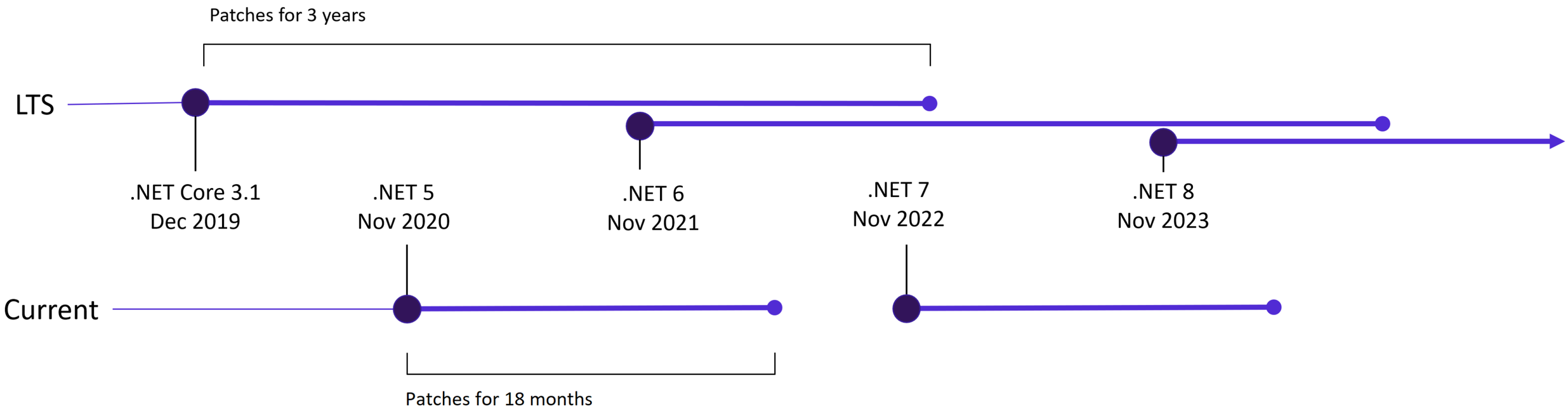
OS Features

Offline support

Installed on device

https://blog.scottlogic.com/2020/07/22/web-or-desktop.html

Oficjalne wsparcie



CODE:ME
UCZYMY PROGRAMOWANIA

{256}
TYDZIEŃ PROGRAMISTY

Dziękuję!

CODE:ME
UCZYMY PROGRAMOWANIA

{256}
TYDZIEŃ PROGRAMISTY

Materiały z prezentacji

https://github.com/mechelewskim/warsztat-csharp-2022-09-06

CODE:ME

{256}

TYDZIEŃ PROGRAMISTY