Linq: Language Integrated Query

¿Qué es Linq?

- Linq es una librería que se usa para ejecutar consultas en C#.
- Es la manera *declarativa* de manipular colecciones en .NET.
- Está implementado como una serie de métodos que se aplican sobre la interface IEnumerable<T>

Kahoot pin number: 1963228

¿Y qué es IEnumerable?

IEnumerable es una interfaz que garantiza que una clase sea ***Iterable.*** Significa que una clase que implementa IEnumerable puede ser pensada como una secuencia de objetos.

Ejemplos: List, array, Stack, Queue

Kahoot pin number: 1963228

Ejemplos LINQ

Son métodos para realizar **consultas** a una secuencia objetos.

Cuales de los siguientes creen que son métodos LINQ?

Kahoot time!

Kahoot pin number: 1963228

Sintaxis LINQ

Existen dos tipos de sintaxis Linq: sintaxis query y sintaxis method.

Ejemplo query sintax:

```
using System.Linq;

    ...

List<string> animalNames = new List<string>
    {"perro", "gato", "elefante", "jirafa", "ruiseñor"};

IEnumerable<string> longAnimalNames =
    from name in animalNames
    where name.Length >= 6
    orderby name.Length
    select name;
```

Sintaxis LINQ

Existen dos tipos de sintaxis Linq: sintaxis query y sintaxis method.

Ejemplo method sintax:

```
using System.Linq;

    ...

List<string> animalNames = new List<string>
    {"perro", "gato", "elefante", "jirafa", "ruiseñor"};


IEnumerable<string> longAnimalNames =
    animalNames
    .Where(name => name.Length >= 6)
    .OrderBy(name => name.Length);
```

Expresiones lambda

Una expresión lambda es una forma conveniente de definir una función anónima, que puede ser utilizada como una variable

```
Func<int, int> multiplicarPor5 = num => num * 5;

int result = multiplicarPor5(7);
```

```
Func<int, int> multiplicarPor5 = num =>
{
    int product = num * 5;
    return product;
};

int result = multiplicarPor5(7);
```

Ejemplos

Método First()

```
List<double> doubles = new List<double> { 2.0, 2.1, 2.2, 2.3 };
double whatsThis = doubles.First()
```

```
List<double> doubles = new List<double> { 2.0, 2.1, 2.2, 2.3 };
double whatsThis = doubles.First(val => val >2.3);
```

Y Find ?

```
List<double> doubles = new List<double> { 2.0, 2.1, 2.2, 2.3 };
double whatsThis = doubles.FirstOrDefault(val => val >2.3);
```

Ejemplos

Skip() y Take()

```
List<bool> bools = new List<bool> { true, false, true, true, false };

IEnumerable<bool> result = bools.Take(3);
```

```
List<bool> bools = new List<bool> { true, false, true, true, false };

IEnumerable<bool> result = bools.Skip(2);
```

Ejemplos

FirstWhile() y TakeWhile()

```
List<int> ints = new List<int> { 1, 2, 4, 8, 4, 2, 1 };

IEnumerable<int> result = ints.TakeWhile(theInt => theInt <5);
```

```
List<int> ints = new List<int> { 1, 2, 4, 8, 4, 2, 1 };

IEnumerable<int> result = ints.SkipWhile(theInt => theInt !=4);
```

Ejemplos

Where()

```
List<int> ints = new List<int> { 1, 2, 4, 8, 4, 2, 1 };

IEnumerable<int> result = ints.Where(theInt => theInt ==2 || theInt == 4);
```

Y FindAll ?

Ejemplos

OrderBy()

```
List<string> strings = new List<string> { "first", "then", "and then",
"finally" };

IEnumerable<string> result = strings.OrderBy(str => str.Length);

IEnumerable<string> result = strings.OrderBy(str => str[2]);
```

Ejemplos

Count()

```
IEnumerable<string> strings = new List<string> { "first", "then", "and then",
"finally" };

int result = strings.Count();
```

```
IEnumerable<string> strings = new List<string> { "first", "then",
"and then", "finally" };

int result = strings.Count(str => str.Contains("then"));
```

Ejemplos

Min() y Max()

```
IEnumerable<int> ints = new List<int> { 2, 2, 4, 6, 3, 6, 5 };
int result = ints.Max();
```

```
IEnumerable<string> strings = new List<string> { "1.2", "1.3",
"1.5", "0.6" };
float result = strings.Min(str => float.Parse(str));
```

Ejemplos

Sum()

```
IEnumerable<int> ints = new List<int> { 2, 2, 4, 6 };
int result = ints.Sum();
```

```
IEnumerable<int> ints = new List<int> { 2, 2, 4, 6 };
int result = ints.Sum(val => val * val);
```

Ejemplos

Select()

```
IEnumerable<string> strings = new List<string> { "one", "two", "three",
"four" };
IEnumerable<int> result = strings.Select(str => str.Length);
```

¿Preguntas?

Más métodos

https://docs.microsoft.com/en-us/dotnet/api/system.linq.enumerable?redirectedfrom=MSDN&view=net-5.0#methods