Trabajo de Fin de Grado

Conversor de números a texto en inglés

Autor

Néstor Pulido Robaina

Tutor

Francisco Javier Carreras Riudavets

Profesor del Departamento de Informática y Sistemas

ÍNDICE

1. ESTADO ACTUAL Y OBJETIVOS.

Este trabajo consiste en la implementación de un servicio web de conversión de números a texto en el idioma inglés. Normalmente, tendemos a escribir números mediante dígitos, especialmente cuando se trata de números grandes, pero existen normas por las cuales, en determinadas circunstancias, se hace necesario sustituir dígitos por palabras.

Pero antes de comentar dichas normas, procederemos a explicar los conceptos generales que describen la formación de números mediante palabras.

1.1. Conceptos generales sobre la conversión de números a texto en inglés.

Como ocurre en el idioma español, la conversión de números a texto en inglés resulta un proceso sistemático, aunque con algunas excepciones, que puede obtener distintos resultados alternativos. Así, un número puede ser escrito en forma cardinal, ordinal, fraccionaria…, y en determinados contextos puede se escrito con palabras ajenas al ámbito de la numeración, como por ejemplo “century” para referirse a 100 años (un siglo).

A continuación describiremos las formas más usadas de escribir un número en inglés.

1.1.1. Números cardinales.

Los números cardinales expresan cantidades. Del 0 al 19 se escriben según la siguiente tabla:

0	Zero	10	Ten
1	One	11	Eleven
2	Two	12	Twelve
3	Three	13	Thirteen
4	Four	14	Fourteen
5	Five	15	Fifteen
6	Six	16	**Six**teen
7	Seven	17	**Seven**teen
8	Eight	18	**Eight**een
9	Nine	19	**Nine**teen

Tabla T.1.1.1.A

Como se puede apreciar, a partir del número 16 se escribe indicando las unidades seguido de la palabra "teen", eliminando la doble "t" en el caso del número 18 (~~eightteen~~ = eighteen).

A partir de 20, las decenas se escriben según la siguiente tabla:

20	Twenty	60	**Six**ty
30	Thirty	70	**Seven**ty
40	Forty	80	**Eight**y
50	Fifty	90	**Nine**ty

Tabla T.1.1.1.B

Obsérvese también como a partir de 60, las decenas se escriben como las unidades pero seguido de la terminación "ty", eliminando la doble "t" en el caso de 80 (~~eightty~~ = eighty).

Si el número está entre 21 y 99, y el dígito de unidades no es cero, se escribe normalmente como dos palabras separadas por un guión (decenas-unidades):

21	Twenty-one	25	Twenty-five
32	Thirty-two	58	Fifty-eight
64	Sixty-four	79	Seventy-nine
83	Eighty-three	99	Ninety-nine

Tabla T.1.1.1.C

Las centenas son regulares (salvo el hecho de que la palabra "hundred" se mantiene en singular independientemente del número que la preceda):

100	One hundred	125	One hundred twenty-five
200	Two hundred	584	Five hundred eighty-four
…	…	…	…
900	Nine hundred	731	Seven hundred thirty-one

Tabla T.1.1.1.D

Lo mismo ocurre con los millares, donde el número de miles va seguido de la palabra "thousand":

1.000	one thousand	20.000	twenty thousand
2.000	two thousand	21.000	twenty-one thousand
10.000	ten thousand	30.000	thirty thousand
11.000	eleven thousand	85.000	eighty-five thousand

Tabla T.1.1.1.E

Llegados a este punto, hay que decir que existen diferencias entre el inglés británico y el inglés americano a la hora de escribir números. En inglés americano, los números de cuatro dígitos con centenas distintas de cero suelen nombrarse usando múltiples “hundred” combinadas con decenas y unidades, por ejemplo:

- 1001: "One thousand one"
- 1103: "Eleven hundred three"
- 1225: "Twelve hundred twenty-five"
- 4042: "Four thousand forty-two"
- 9999: "Ninety-nine hundred ninety-nine."

En inglés británico, este estilo es común para múltiplos de 100 entre 1000 y 2000, por ejemplo 1500 como “fifteen hundred”, pero no para números más grandes.

En determinados contextos, los americanos pueden pronunciar números de cuatro dígitos con decenas distintas de cero y unidades como pares de números de dos dígitos sin decir “hundred” e insertando la palabra “oh” para las decenas con cero, por ejemplo:

- 2659: "twenty-six fifty-nine"
- 4105: "forty-one oh five"

Probablemente, este uso ha evolucionado a partir del uso para indicar años, por ejemplo, "nineteen eighty-one" (1981). Se evita para números menores de 2500 si el contexto puede generar confusión con la hora del día: “ten ten” o “twelve oh four”. También puede llegar a usarse en un contexto económico dónde se espera que haya parte decimal, por ejemplo al indicar el precio de un artículo.

Los números intermedios se leen de forma diferente dependiendo de su uso. La forma típica de nombrarlos se da cuando los números son usados para contar. Otra forma es cuando se usan como etiquetas. La siguiente tabla muestra distintas alternativas:

	Inglés británico coloquial	**Inglés americano coloquial**	**Inglés británico coloquial (normalmente no usado en América)**
	"How many marbles do you have?"	"What is your house number?"	"Which bus goes to the high street?"
101	"A hundred and one."	"One-oh-one." (Aquí, "oh" se usa para el dígito 0).	"One-oh-one."
109	"A hundred and nine."	"One-oh-nine."	"One-oh-nine."
110	"A hundred and ten."	"One-ten."	"One-one-oh."
117	"A hundred and seventeen."	"One-seventeen."	"One-one-seven."
120	"A hundred and twenty."	"One-twenty."	"One-two-oh", "One-two-zero."
152	"A hundred and fifty-two."	"One-fifty-two."	"One-five-two."
208	"Two hundred and eight."	"Two-oh-eight."	"Two-oh-eight."
334	"Three hundred and thirty-four."	"Three-thirty-four."	"Three-three-four."

Tabla T.1.1.1.F

Nota: Al escribir un cheque, el número 100 siempre se escribe como “one hundred”, no como “a hundred”.

En inglés americano, a muchos estudiantes se les enseña a no usar la palabra “and” en cualquier parte del número, de forma que no es usada antes de las decenas y unidades. En su lugar se usa un delimitador verbal cuando se trata con números compuestos. Por lo tanto, en lugar de "three hundred and seventy-three", se diría "three hundred seventy-three".

Para números superiores a un millón, hay dos formas distintas de nombrarlos:

- **Escala grande** (cada vez menos usada en inglés británico), donde *mil millones* se nombra con la palabra “milliard”, y *un millón de millones* con la palabra “billion”.

- **Escala pequeña** (usada en inglés americano y cada vez más en inglés británico), donde *mil millones* se nombra como “billion” y no se usa la palabra “milliard”.

  La siguiente tabla muestra estas diferencias:

Número	Escala pequeña	Escala grande
1,000,000	one million	one million
1,000,000,000	one billion a thousand million	one milliard a thousand million
1,000,000,000,000	one trillion a thousand billion	one billion a million million
1,000,000,000,000,000	one quadrillion a thousand trillion	one billiard a thousand billion
1,000,000,000,000,000,000	one quintillion a thousand quadrillion	one trillion a million billion
1,000,000,000,000,000,000,000	one sextillion a thousand quintillion	one trilliard a thousand trillion

Tabla T.1.1.1.G

Los números por encima de un *trillón*, en potencias ascendentes de diez, se componen de la siguiente forma: “quadrillion”, “quintillion”, “sextillion”, “septillion”, “octillion”, “nonillion”, “decillion”, “undecillion”, “duodecillion”, “tredecillion”, “quattuordecillion”, y “quindecillion”, que es un uno seguido de 48 ceros. El número más grande de esta lista es *un mili-millón* (“milli-millillion”), que es 10 elevado a 3 000 003.

El término *gúgol* (“googol”) es el nombre de un número acuñado en 1938 por Milton Sirotta, un niño de 9 años, sobrino del matemático estadounidense Edward Kasner. Un *gúgol* es un uno seguido de cien ceros. Por su parte, un *gúgolplex* (“googolplex”) es un uno seguido de un *gúgol* de ceros, esto es, 10 elevado a la *gugol-ésima* potencia.

Aunque el inglés británico ha utilizado tradicionalmente el sistema de escala grande, el uso de la escala pequeña se ha incrementado en los últimos años, hasta el punto de que el gobierno británico y los sitios web de la BBC usan exclusivamente la escala pequeña.

La siguiente tabla muestra cómo se componen números largos en inglés americano:

Número	Escrito	Pronunciado
1,200,000	1.2 million	one point two million
3,000,000	3 million	three million
250,000,000	250 million	two hundred fifty million
6,400,000,000	6.4 billion	six point four billion
23,380,000,000	23.38 billion	twenty-three point three eight billion

Tabla T.1.1.1.H

Con frecuencia, los números grandes se escriben con semiespacios para separar los miles (y, a veces, con espacios normales o apóstrofes) en lugar de comas, para evitar así confusiones en países donde se usa la coma decimal. Por lo tanto, un *millón* se escribe como 1 000 000. En algunas partes, se puede usar un punto como separador de miles, pero entonces el separador decimal debe ser una coma.

1.1.2. Números ordinales.

Los números ordinales se refieren a una posición dentro de una serie. La forma de escribirlos es muy similar a los cardinales, aunque con algunas diferencias.

Del 0 al 19 se escriben según la siguiente tabla:

0th	zeroth	10th	tenth
1st	first	11th	eleventh
2nd	second	12th	twelfth
3rd	third	13th	thirteenth
4th	fourth	14th	fourteenth
5th	fifth	15th	fifteenth
6th	sixth	16th	sixteenth
7th	seventh	17th	seventeenth
8th	eighth	18th	eighteenth
9th	ninth	19th	nineteenth

Tabla T.1.1.2.A

A partir de 20, las decenas se escriben según la siguiente tabla:

20th	twentieth	60th	sixtieth
30th	thirtieth	70th	seventieth
40th	fortieth	80th	eightieth
50th	fiftieth	90th	ninetieth

Tabla T.1.1.2.B

Como hemos visto hasta ahora, a excepción de 1st, 2nd 3rd y 5th, se forman igual que los cardinales pero añadiendo la terminación “th”o “tieth”.

“Zeroth” sólo tiene sentido cuando un recuento empieza en cero, lo cual ocurre en contextos matemáticos o informáticos.

Los números ordinales tales como “21st”, “33rd”, etc., se forman combinando una decena cardinal con una unidad ordinal.

21st	twenty-first	64th	sixty-fourth
25th	twenty-fifth	79th	seventy-ninth
32nd	thirty-second	83rd	eighty-third
58th	fifty-eighth	99th	ninety-ninth

Tabla T.1.1.2.C

Los ordinales más altos no suelen escribirse con palabras, a menos que sean números “redondos” (thousandth, millionth, billionth, etc). Se escriben usando dígitos y letras tal como se describe a continuación. Aquí hay algunas reglas que deberían tenerse en cuenta:

- Los sufijos –th, -st, -nd y –rd se escriben ocasionalmente como superíndices.

- Si el dígito de decenas es 1, se escribe “th” después del número. Por ejemplo: 13th, 19th, 112th, 9,311th.

- Si el dígito de decenas no es 1, se usa la siguiente tabla:

Si el dígito de unidades es:	0	1	2	3	4	5	6	7	8	9
Se escribe después del número:	th	st	nd	rd	th	th	th	th	th	th

Tabla T.1.1.2.D

Por ejemplo: 2nd, 7th, 20th, 23rd, 52nd, 135th, 301st.

Estas abreviaciones ordinales son en realidad contracciones híbridas de un número y una palabra. *1st* es "1" + "st" de "first". Igualmente, "nd" es usada para "second" y "rd" para "third". En el ámbito legal y en algunas publicaciones antiguas, la abreviación ordinal para "second" y "third" es simplemente "d". Por ejemplo: 42d, 33d, 23d.

La práctica de usar "d" para denotar "second" y "third" continúa aún en designaciones de unidades de las fuerzas armadas norteamericanas, por ejemplo "533d Squadron".

Cualquier nombre ordinal que no termine en "first", "second" o "third", termina en "th".

1.1.3. Números fraccionales.

En inglés hablado, los números ordinales también se usan para cuantificar el denominador de una fracción. Así pues, "fifth" puede ser tanto el elemento entre "fourth" y "sixth", como la fracción resultante de dividir la unidad en cinco partes (lo que en español diríamos como "un quinto" o "una quinta parte"). En este último uso, los números ordinales pueden pluralizarse añadiendo la terminación "s" cuando el numerador es mayor que uno: "two sevenths" (2/7).

Existe una excepción, y es cuando el denominador es dos. En este caso se usa el término "half" (en español "mitad"), o bien "halves" (plural de "half" - en español "mitades"). Cuando el denominador es cuatro, se puede escribir con el ordinal "fourth" o bien con el término "quarter".

En cuanto al numerador, simplemente se escribe como un número cardinal.

En la siguiente tabla se muestran algunos ejemplos de números fraccionales:

1/16	one sixteenth
1/10 (ó 0.1)	one tenth
1/8	one eighth
2/10 (ó 0.2)	two tenths
1/4	one quarter (o principalmente en inglés americano one fourth)
3/10 (ó 0.3)	three tenths
1/3	one third
3/8	three eighths
1/2	one half
2/3	two thirds

7/10 (ó 0.7)	seven tenths
3/4	three quarters (o principalmente en inglés americano three fourths)
7/8	seven eighths
15/16	fifteen sixteenths

Tabla T.1.1.3.A

Alternativamente, y para números mayores, se puede decir “one over two” para 1/2, “five over eight” para 5/8, etc. Este “over” es comúnmente utilizado en matemáticas.

Aunque en inglés hablado no es usual expresar números fraccionales con un numerador o denominador muy grande, la norma puede extenderse a cualquier número:

308,120/564,380: “three hundred and eight thousand and one hundred and twenty” *(numerador)* “five hundred and sixty-four thousand and three hundred and eightieths” *(denominador)*.

1.1.4. Números decimales.

Los decimales son una forma de escribir números fraccionales sin tener que escribir una fracción con numerador y denominador. La fracción “7/10” puede ser escrita como el decimal “0.7”, que es el resultado de dividir 7 entre 10. El punto indica que se trata de un decimal. Si el decimal es menor que 1, se coloca un cero delante del punto decimal (“0.7” en lugar de “.7”).

Cuando hablamos de texto, el decimal “0.7” se puede escribir como “seven **tenths**” (en español “siete décimas”) o “zero **point** seven” (“cero punto siete”). Si el decimal es mayor que 1, se puede nombrar la parte entera y la parte decimal unidas con la palabra “and”, de forma que “3.7” se puede escribir como “three **and** seven **tenths**”.

Si la parte decimal tiene dos dígitos, en lugar de “tenths” utilizamos “hundredths”. Por ejemplo, “0.37” se puede escribir como “thirty-seven **hundredths**”, “zero **point** three seven” (inglés británico), o también “zero **point** thirty-seven” (inglés americano). Al igual que antes, si el decimal es mayor que 1 se nombra la parte entera y la parte decimal unidas por “and”. Por ejemplo, “12.37” se puede escribir como “twelve **and** thirty-seven **hundreths**”, “twelve **point** three seven” (inglés británico) o también “twelve **point** thirty-seven” (inglés americano).

Si la parte decimal tiene tres dígitos, utilizamos “thousandths”, tal como se muestra en los siguientes ejemplos:

	Sin “point”	Con “point” (BrE)	Con “point” (AmE)
0.749	“seven hundred forty-nine **thousandths**”	“zero **point** seven four nine”	“zero **point** seven hundred forty-nine”
234.987	“two hundred thirty-four **and** nine hundred eighty-seven **thousandths**”	“two hundred and thirty-four **point** nine eight seven”	“two hundred thirty-four **point** nine hundred eighty-seven”

Tabla T.1.1.4.A

Nota: En la tabla de arriba (y en lo sucesivo), **BrE** indica “British English” (inglés británico) y **AmE** indica “American English” (inglés americano).

En caso de haber ceros después del punto decimal, si no utilizamos “point” omitimos las posiciones correspondientes a dichos ceros:

	Sin “point”	Con “point” (BrE)	Con “point” (AmE)
0.064	“sixty-four **thousandths**”	“zero **point** zero six four”	“zero **point** zero sixty-four”
0.005	“five **thousandths**”	“zero **point** zero zero five”	“zero **point** zero zero five”

Tabla T.1.1.4.B

Nótese que si no se utiliza la palabra “point” y la parte entera es cero, ésta se omite. Por otra parte, mientras en inglés americano es normal utilizar un solo cardinal tras la palabra “point” para referirse a toda la parte decimal, en inglés británico dicha parte se nombra dígito por dígito:

	Ingés británico (BrE)	Inglés americano (AmE)
0.749	“zero **point** seven four nine”	“zero **point** seven hundred forty-nine”

Tabla T.1.1.4.C

En todos los casos en que el decimal es mayor que 1, en lugar de “point” se puede utilizar la palabra “and” para separar la parte entera de la parte decimal, pero en tal caso no debería utilizarse en otras partes del número.

En general, la parte decimal se especifica en función de la posición que ocupe el último dígito decimal, según la siguiente tabla:

POSICIÓN DE LOS DÍGITOS													
millions	hundred thousands	ten thousands	thousands	hundreds	tens	ones	and	tenths	hundredths	thousandths	ten-thousandths	hundred-thousandths	millionths
7	6	5	4	3	2	1	0	1	2	3	4	5	6

Tabla T.1.1.4.D

Veamos algunos ejemplos:

- 5.3 → five and three **tenths** (1 dígito decimal)
- 49.01 → forty-nine and one **hundredth** (2 dígitos decimales)
- 216.231 → two hundred sixteen and two hundred thirty-one **thousandths** (3 dígitos decimales)
- 9,010.0359 → nine thousand, ten and three hundred fifty-nine **ten-thousands** (4 dígitos decimales)
- 76,053.00047 → seventy-six thousand, fifty-three and forty-seven **hundred-thousands** (5 dígitos decimales)
- 229,000.000081 → two hundred twenty-nine thousand and eighty-one **millionths** (6 dígitos decimales)
- 1,729,405.008365 → one million, seven hundred twenty-nine thousand, four hundred five and eight thousand, three hundred sixty-five **millionths**. (6 dígitos decimales)

Nota: Las comas se utilizan como separadores de miles y el punto como indicador del comienzo de la parte decimal.

Otras formas de expresar decimales o fracciones.

Para pequeñas cantidades decimales, existen otras alternativas:

- 1/2 → 0.5 → a half (una mitad)
- 1/3 → 0.3 → a third (un tercio)

- 1/4 → 0.25 → a quarter (un cuarto)
- 1/5 → 0.2 → a fifth (un quinto)
- 1/6 → 0.16 → a sixth (un sexto)

… y así sucesivamente.

1.1.5. Números negativos.

El nombre de un número negativo es igual al nombre correspondiente al número positivo precedido de la palabra "minus" en inglés británico o "negative" en inglés americano. Por tanto, el número -5.2 puede expresarse como "minus five point two", "minus five and two tenths", "negative five point two" o "negative five and two tenths".

1.1.6. Nombres de números largos.

La siguiente tabla muestra nombres de números largos que han sido encontrados en muchos diccionarios de inglés, y por ello tienen un especial reclamo para convertirse en "palabras reales". Los valores de "inglés tradicional" mostrados a continuación, son inusuales en inglés americano, y comienzan a ser extraños en inglés británico, pero sus variantes en otros idiomas son predominantes en muchas zonas de habla no inglesa, incluyendo Europa continental y países de habla hispana en Latino-América.

El idioma inglés también tiene otras muchas palabras, como "zillion", usada informalmente para referirse a cantidades grandes pero sin especificar.

Nombre	Escala Corta (Inglés Americano)	Escala Larga (Inglés Británico)
Million	10^6	10^6
Milliard	---	10^9
Billion	10^9	10^{12}
Trillion	10^{12}	10^{18}
Quadrillion	10^{15}	10^{24}
Quintillion	10^{18}	10^{30}
Sextillion	10^{21}	10^{36}
Septillion	10^{24}	10^{42}
Octillion	10^{27}	10^{48}
Nonillion	10^{30}	10^{54}
Decillion	10^{33}	10^{60}
Undecillion	10^{36}	10^{66}

Duodecillion	10^{39}	10^{72}
Tredecillion	10^{42}	10^{78}
Quattuordecillion	10^{45}	10^{84}
Quindecillion	10^{48}	10^{90}
Sexdecillion (Sedecillion)	10^{51}	10^{96}
Septendecillion	10^{54}	10^{102}
Octodecillion	10^{57}	10^{108}
Novemdecillion (Novendecillion)	10^{60}	10^{114}
Vigintillion	10^{63}	10^{120}
…	…	…
…	…	…
Centillion	10^{303}	10^{600}

Tabla T.1.1.6.A

Aparte de “million”, las palabras de esta lista terminadas en “-illion” son derivadas de añadir prefijos (“bi-“, “tri-“, etc., derivadas del Latín) a la raíz. “Centillion” parece ser el nombre del número más alto terminado en “-illion” que está incluído en estos diccionarios. “Trigintillion”, frecuentemente citada en discusiones de nombres de números largos, no está incluida en ninguno de ellos, así como tampoco está ninguno de los nombres que pueden ser creados fácilmente extendiendo el patrón de nombres (“unvigintillion”, “duovigintillion”, “duoquinguagintillion”, etc.).

Cada número por encima del millón tiene dos posibles nombres: uno para la escala corta, donde los sucesivos nombres difieren por un factor de un millar (10^3), y otro para la escala larga, donde los sucesivos nombres difieren por un factor de un millón (10^6).

Una forma fácil de encontrar el valor de estos números en escala corta es tomar el número indicado por el prefijo (tal como 2 en “billion”, 4 en “quadrillion”, 18 en “octodecillion”, etc.), sumarle uno, y multiplicar el resultado por 3. Por ejemplo, en un “trillion”, el prefijo es “tri”, que significa 3. Sumándole 1 nos da 4. Ahora, multiplicando 4 por 3 nos da 12, que es la potencia a la cual se debe elevar 10 para expresar un “trillion” en escala corta en notación científica. Un “trillion” = 10^{12}.

En el caso de la escala larga, hay que multiplicar el número del prefijo por 6. Por ejemplo, para un “billion”, cuyo prefijo significa 2, multiplicando 2 por 6 nos da 12, que es la potencia a la cual se debe elevar 10 para expresar un “billion”

en escala larga en notación científica. Un "billion" = 10^{12}. Los valores intermedios ("billiard", "trilliard", etc.) pueden ser convertidos del mismo modo, sumándole ½ al número del prefijo y multiplicándolo por 6. Por ejemplo, para "septilliard", cuyo prefijo indica 7, sumándole ½ a 7 y multiplicándolo por 6 nos da 45. Un "septilliard" = 10^{45}.

El procedimiento de nombrar números largos está basado en tomar un número *n* en intervalos de 10^{3n+3} (escala corta) o 10^{6n} (escala larga) y concatenar las raíces latinas para las posiciones de sus unidades, decenas y centenares, junto con el sufijo "-illion". De esta forma, los números hasta $10^{3 \cdot 999+3} = 10^{3000}$ (escala corta) o $10^{6 \cdot 999} = 10^{5994}$ (escala larga) pueden ser nombrados.

	Unidades	**Decenas**	**Centenares**
1	Un	N Deci	NX Centi
2	Duo	MS Viginti	N Ducenti
3	Tre (*)	NS Triginta	NS Trecenti
4	Quattuor	NS Quadraginta	NS Quadringenti
5	Quinqua	NS Quinquaginta	NS Quingenti
6	Se (*)	N Sexaginta	N Sescenti
7	Septe (*)	N Septuaginta	N Septingenti
8	Octo	MX Octoginta	MX Octingenti
9	Nove (*)	Nonaginta	Nongenti

Tabla T.1.1.6.B

(*) ^ Cuando va precedido de un componente marcado como ^{S} o ^{X}, "tre" pasa a ser "tres" y "se" pasa a ser "ses" o "sex"; igualmente, cuando va precedido de un componente marcado como ^{M} o ^{N}, "septe" y "nove" pasan a ser "septem" y "novem" o "septen" y "noven".

Aplicando este procedimiento podemos completar la tabla vista anteriormente:

Valor	**Escala Corta**	**Escala Larga**		**Valor**	**Escala Corta**	**Escala Larga**
…	…	…		…	…	…
10^{66}	Unvigintillion	Undecillion		10^{111}	Sestrigintillion	Octodecilliard
10^{69}	Duovigintillion	Undecilliard		10^{114}	Septentrigintillion	Novendecillion
10^{72}	Tresvigintillion	Duodecillion		10^{117}	Octotrigintillion	Novendecilliard
10^{75}	Quattuorvigintillion	Duodecilliard		10^{120}	Noventrigintillion	Vigintillion
10^{78}	Quinquavigintillion	Tredecillion		10^{123}	Quadragintillion	Vigintilliard
…	…	…		…	…	…

Tabla T.1.1.6.C

Todos los diccionarios incluyen “googol” (10^{100}) y “googolplex” ($10^{10\ e\ 100}$), pero ninguno de ellos incluye nombres mayores de la familia del “googol”, como “googolduplex”, etc. El *Oxford English Dictionary* comenta que “googol” y “googolplex” no tienen un uso formal en matemáticas.

1.1.7. Números especiales.

Algunos números tienen nombres especiales (adicionalmente a sus nombres regulares):

- 0: tiene otros nombres, dependiendo del contexto:
    - *zero:* formato científico
    - *naught / nought:* principalmente uso británico
    - *aught:* principalmente arcaico, pero aún utilizado ocasionalmente cuando un dígito en medio de un número es cero (como "thirty-aught-six", el cartucho del rifle .30-06 Springfield y cualquier asociación de armas que lo utilice)
    - oh: usado al deletrear números (por teléfono, cuentas bancarias, línea de autobús, etc.)
    - nil: en puntuaciones de determinados deportes que usan porterías, como fútbol, rugby, hockey, etc. (uso británico): ("The score is two–nil.")
    - nothing: en puntuaciones de deportes en general (uso americano): ("The score is two–nothing.")
    - null: usado técnicamente para referirse a un objeto o idea relativo con la nada.
    - love: en tenis, bádminton, squash y deportes similares.
    - zilch, nada (del español), zip: usado informalmente para enfatizar la nada; suele usarse especialmente en combinación de uno con otro: ("You know nothing—zero, zip, nada, zilch!") (uso americano)
    - nix: también usado como verbo (uso americano, principalmente)
    - cypher / cipher: arcaico, del francés “chiffre”

  - goose egg: (informal)
  - duck: (usado en cricket)
  - blank: la mitad de una pieza de dominó sin puntos.
- 1:
  - ace: usado en ciertos deportes y juegos, como el tenis o el golf.
  - birdie: usado en golf.
  - solo: español.
  - unit: unidad.
  - linear: el grado de un polinomio si es 1.
  - unity: en matemáticas.
- 2:
  - couple: pareja.
  - pair: par.
  - deuce: la cara de un dado, carta o dominó con dos puntos.
  - “eagle” en golf denota dos golpes menos que un “par”.
  - duo: dúo.
  - quadratic: el grado de un polinomio si es 2.
- 3:
  - trey: la cara de un dado o carta con 3 puntos, canasta de 3 puntos en baloncesto.
  - trio: trío.
  - trips: trío en una mano de póker (3 cartas con el mismo número).
  - cubic: el grado de un polinomio si es 3.
  - albatross: usado en golf.

- 4:
    - cater: la cara de un dado o carta con cuatro puntos.
    - quartet: cuarteto.
    - “quartic” o “biquadratic”: el grado de un polinomio si es 4.
    - quad: abreviatura de “quadruple”.
    - condor: usado en golf.
- 5:
    - cinque: la cara de un dado o carta con 5 puntos.
    - quintet: quinteto.
    - nickel: en americano informal, 5 centavos, pero aplicado a referencias no monetarias.
    - quintic: el grado de un polinomio si es 5.
    - quint: abreviatura de “quintuplet”.
- 6:
    - half a dozen: media docena.
    - sice: la cara de un dado o carta con 6 puntos.
    - sextet: sexteto.
    - “sextic” o “hectic”: el grado de un polinomio si es 6.
- 7:
    - septet: septeto.
    - “septic” or “heptic”: el grado de un polinomio si es 7.
- 8: octet: octeto.
- 9: nonet: noneto.

- 10:
    - a metric dozen: una docena métrica.
    - dime: en americano informal, el valor de 10 centavos, pero aplicado a referencias no monetarias.
    - decet: deceto.
- 11: a banker's dozen.
- 12: a dozen (primera potencia de la base), usado principalmente en comercio.
- 13: a baker's dozen. Una docena + 1.
- 20: a score (primera potencia de la base vigesimal), actualmente arcaico.
- 50: half a century. Medio siglo, también usado en resultados de Cricket.
- 100:
    - a century: un siglo, también usado en resultados de Cricket.
    - a ton, en inglés de la “Commonwealth”, la velocidad de 100 mph o 100 km/h.
- 120: a great hundred: doce decenas, o también “small gross” (diez docenas), ambos arcaicos. A veces también puede ser referido como “duodecimal hundred”.
- 144: a gross. Una docena de docenas (segunda potencia de la base duodecimal), usado principalmente en comercio.
- 1000:
    - a grand, coloquialmente usado en especial cuando se refiere a dinero, también en fracciones y múltiplos, por ejemplo “half a grand”, “two grand”, etc.
    - K, originalmente de la abreviación de kilo-, por ejemplo "He only makes $20K a year."
- 1728: a great gross (a dozen gross, tercera potencia de la base duodecimal), usado principalmente en comercio.

- 10,000: a myriad (a hundred hundred), comúnmente usado en el sentido de un número indefinido muy alto.
- 100,000: a lakh (a hundred thousand), procedente del inglés indio.
- 10,000,000: a crore (a hundred lakh), procedente del inglés indio.
- 10^{100}: googol (1 seguido de 100 ceros), usado en matemáticas.
- 10^{googol}: googolplex (1 seguido de un googol de ceros)
- $10^{googolplex}$: googolplexplex (1 seguido de un googolplex de ceros)

Las combinaciones de números en la mayoría de las puntuaciones deportivas son leídas como en los siguientes ejemplos:

- 1–0 Inglés británico: *one-nil*; Inglés americano: "*one-nothing*", "*one-zip*", o "*one-zero*".
- 0–0 Inglés británico: "*nil-nil*", o rara vez "*nil all*"; Inglés americano: "*zero-zero*" o "*nothing-nothing*", (ocasionalmente "*scoreless*" o "*no score*")
- 2–2 "*two-two*" o "*two all*"; Inglés americano: también "*twos*", "*two to two*", "*even at two*", o "*two up*".

En el caso de las puntuaciones de Tenis, las convenciones de nombres son distintas al resto de deportes.

1.2. Normas para decidir si escribimos dígitos o palabras.

Escribir números con palabras es una práctica que puede parecernos incómoda e innecesaria, pero lo cierto es que en determinadas circunstancias se convierte en algo recomendable, si no necesario.

¿Y cuáles son esas circunstancias? – En el caso que nos ocupa, que es en el idioma inglés, se ha estipulado una serie de normas para decidir, sistemáticamente, si escribimos un número mediante dígitos o mediante palabras. Algunas de estas normas son las siguientes:

- Escribir con palabras los números que puedan escribirse con una o dos palabras, números redondos y números ordinales. En escritura académica general, se debe escribir estos números con palabras:

todos los que estén por debajo de cien (por ejemplo “ninety-nine”), los números redondos (por ejemplo “four hundred”, “two thousand”, etc.), y los números ordinales (por ejemplo “third”, “twenty-fifth”, etc.).

- Escribir con palabras los números que comienzan una frase. Si esto resulta muy incómodo, es preferible reconstruir la frase para evitar empezar con el número. En el caso de fechas sí está permitido.

- Escribir con palabras números aproximados y algunas horas del día. En escritura académica no técnica, se puede escribir con palabras números aproximados a una cifra real, incluyendo fracciones (por ejemplo “about half the students”, “hundreds of times”, etc), así como medias horas y cuartos de hora (por ejemplo “six o’clock”, “quarter past seven”, “midday”, “half past six”, etc.).

- Evitar confusiones entre números dentro de una frase. Cuando se usan dos números juntos o se reparten dentro de una frase, es necesario utilizar palabras en alguno de ellos (por ejemplo “The computer laboratory hast 24 thirty-centimetre monitors”).

- Escribir con palabras números ordinales en nombres, y también en nombres numéricos de calles (por ejemplo “The Third Reich”, “The Fourth Estate”, “Sixth Avenue”, etc).

- Escribir con palabras en documentos legales u oficiales, donde la claridad es esencial (por ejemplo “the sum of nine hundred and forty-three pounds sterling”, “a distance of no less than two hundred yards from the plaintiff”, etc).

- Escribir dígitos en lugar de palabras para números mayores que cien y en las siguientes situaciones:

    - Dinero: Usar dígitos para cantidades exactas (por ej. 24.28€), pero usar dígitos y palabras para cantidades redondas y grandes (por ej. 15 million).

    - Edad: Usar dígitos para especificar la edad de una persona (por ej. 58 years old).

- Medidas: Usar dígitos con el símbolo de medida (por ej. 6 cm).
- Porcentajes: Usar “55%”, “55 percent” o “fifty-five percent”.
- Fracciones y decimales: En el caso de las fracciones, se puede escribir con dígitos o palabras. Si se usan palabras, se debe unir la parte fraccional con un guión (por ej. 2/3 o “two-thirds”). En el caso de los decimales, dar las cantidades exactas en dígitos (por ej. 0.45).
- Encuestas: Escribir los resultados de la encuesta con dígitos.
- Puntuaciones de juegos y deportes: Escribir las puntuaciones con dígitos.
- Estadísticas: Usar dígitos para describir información estadística.
- Direcciones: Usar dígitos para indicar una dirección (por ej. “47 Marston Street”).
- Nombres de carreteras: Usar dígitos para indicar una carretera (por ej. “A40”, “M25”, etc).
- Épocas: Elegir dentro de una variedad de formatos, pero ser consistente (por ej. “the eighteen century” o “the 18th century”, “from the 1960s to the 1990s”).
- Fechas: Usar este orden (día/mes/año) consistentemente (por ej. “Tuesday 23 February 2008”).
- Horas del día: Elegir dentro de una variedad de formatos, pero ser consistente (por ej. “9 am” o “9.00 am” o “8.22 pm”). Si no se está usando “am” o “pm”, escribir la hora con palabras (por ej. “the eight-thirty bus”, “four o’clock in the afternoon”). Para mediodía y media noche, escribir con palabras (“midday” o “midnight” respectivamente).
- Períodos: Usar dígitos (por ej. “pages: 56-74”, “115-117”, “streets 36-99 Spa St”).

- Secciones de un libro: Usar dígitos para referirnos a las secciones de libros (por ej. “volume 5”, “chapter 6”, “page 45”, etc).

1.3. Objetivos a cubrir en este trabajo.

Como hemos visto, si queremos manejar correctamente el idioma, es necesario saber escribir números con palabras. Obviamente, escribir números pequeños parece bastante fácil, pero… ¿y si tuviéramos que escribir números más grandes? –No es, ni muchísimo menos, algo habitual, sobre todo teniendo en cuenta la norma que hemos comentado de escribir con dígitos los números mayores que cien, pero simplemente por una cuestión de curiosidad, o bien por ampliar nuestros conocimientos sobre el idioma, merece la pena disponer de una herramienta que realice estas complejas combinaciones de palabras por nosotros.

Por otra parte, tal como vimos en el apartado anterior sobre conceptos generales, un número puede escribirse de varias formas: en escala corta, escala grande, utilizando comas o palabras “and” como separadores, etc. Normalmente cada persona está acostumbrada a escribir números siempre de la misma manera, pero conocer otras formas alternativas puede resultar útil e interesante, como por ejemplo el uso de la palabra “oh” por parte de los americanos para referirse a un cero.

Con todo ello, la publicación de este servicio web pretende cubrir los siguientes objetivos:

- Dar a conocer las distintas alternativas a la hora de escribir un número con palabras.

- Enseñar cómo concatenar correctamente las palabras que componen un número, independientemente de su tamaño.

- Aportar una herramienta de comprobación ortográfica en la escritura de números.

- Servir de apoyo a otras aplicaciones informáticas.

2. JUSTIFICACIÓN DE LAS COMPETENCIAS ESPECÍFICAS CUBIERTAS.

En la elaboración de este trabajo se han cubierto las áreas o competencias específicas que se detallan a continuación.

2.1. Trabajo de fin de grado.

- **TFG01**: Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

  Este trabajo ha sido desarrollado exclusivamente por su autor, bajo la supervisión del tutor del mismo. En él se han aplicado conceptos y metodologías estudiadas a lo largo de la carrera, especialmente en el ámbito de la programación orientada a objetos.

2.2. Comunes a la ingeniería informática.

- **CII06**: Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.

  El planteamiento y diseño de los mecanismos para llevar a cabo el proceso de conversión ha sido crucial para obtener una aplicación eficiente y sostenible. Buena parte del tiempo total del trabajo se ha empleado en la búsqueda de las mejores soluciones posibles a la cuestión planteada.

- **CII07**: Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.

  A la hora de diseñar las estructuras de datos se han tenido en cuenta factores como la rapidez de ejecución, el consumo de recursos y la estructuración coherente y organizada de la información que se maneja.

- **CII014**: Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.

  Al tratarse de un servicio web que, de cumplirse las expectativas de demanda, va a ser ampliamente utilizado por muchas personas a nivel mundial, ha sido necesaria la implementación de parte del código utilizando técnicas de programación paralela, que sin lugar a dudas otorgan al servicio un tiempo de respuesta menor al que tendría con una programación tradicional.

2.3. Ingeniería del Software.

- **IS01**: Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.

  El servicio web desarrollado cumple perfectamente los requisitos planteados, devolviendo un amplio conjunto de resultados que se obtienen en un corto periodo de tiempo más que razonable.

2.4. Sistemas de información.

- **SI01**: Capacidad de integrar soluciones de tecnologías de la información y las comunicaciones y procesos empresariales para satisfacer las necesidades de información de las organizaciones, permitiéndoles alcanzar sus objetivos de forma efectiva y eficiente, dándoles así ventajas competitivas.

  Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. El servicio desarrollado en este trabajo pretende cubrir las necesidades que puedan surgir en organizaciones de distintos ámbitos: educativo, económico, social, etc.

2.5. Tecnologías de la información.

- **TI06**: Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.

  Como ya hemos comentado, hemos desarrollado una solución al problema planteado a través de un servicio web. Ello implica tener, al menos, un mínimo conocimiento de lo que es un servicio web y de las posibilidades que ofrece a través de Internet.

3. APORTACIONES.

Podemos decir que con este trabajo se hacen aportaciones en los siguientes ámbitos:

- **Educativo**: éste es el campo donde radica la principal aportación del trabajo realizado. El servicio no sólo puede resultar útil para estudiantes del idioma inglés; incluso aquellas personas para las que dicho idioma es su lengua materna pueden considerarlo interesante, dado que permite resolver la compleja construcción de números de gran tamaño con palabras, lo cual en ocasiones se convierte en una tarea tremendamente incómoda. Es más, cuando se manejan cifras extremadamente grandes, los términos asociados a dichas cifras son desconocidos por la mayor parte de las personas ajenas al entorno de las matemáticas.

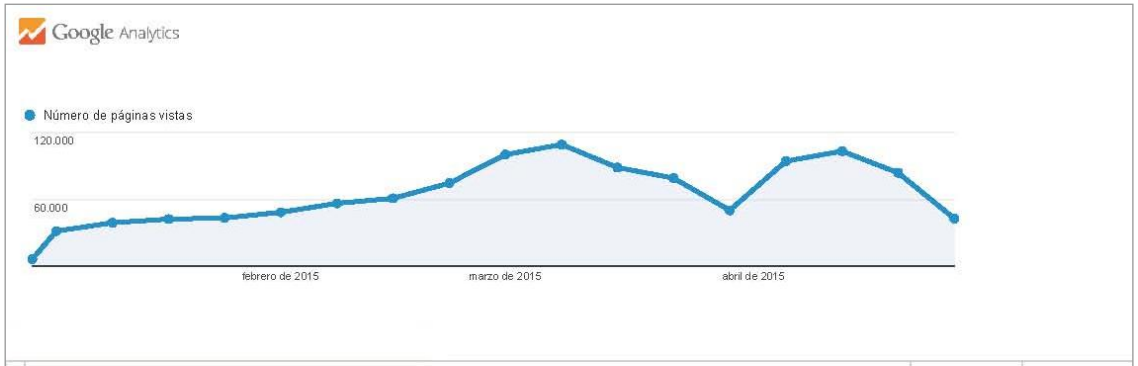
- **Económico**: puede llegar a ser realmente útil a la hora de rellenar cheques bancarios, ya que en éstos se exige indicar los importes tanto con números como con letras.

- **Jurídico/Administrativo**: en documentos oficiales o de carácter legal, es habitual especificar cifras tanto con dígitos como con palabras, con objeto de ser lo más claros y concisos posible.

- **Social**: puede ser utilizado conjuntamente con otras aplicaciones para la conversión automática a voz de números en textos. Este tipo de aplicaciones están en auge y son cada vez más importantes, especialmente para personas con discapacidades.

Prueba de que un servicio de estas características puede llegar a tener bastante éxito la tenemos en la web "Números TIP" (*Text & Information Processing*), desarrollada por el tutor de este trabajo, *Francisco Javier Carreras Riudavets*, que convierte números a texto en español.

Las estadísticas obtenidas por el servicio *Google Analytics* para la medición de visitas a dicha web no dejan lugar a dudas:



	Título de la página	Número de páginas vistas	Número de páginas vistas únicas
		1.150.382 % del total: 31,29 % (3.676.688)	527.966 % del total: 40,63 % (1.299.307)
1.	Números TIP - Convertir número a texto o letras	**784.262** (68,17 %)	264.437 (50,09 %)
2.	Números TIP - Convertir el número XXXVI a texto.	**29.031** (2,52 %)	18.635 (3,53 %)
3.	Números TIP - Convertir el número 12345678901234567890123456789012345678901234567890 a texto.	**21.795** (1,89 %)	17.270 (3,27 %)
4.	Números TIP - Convertir el número 1635828711189135853469972734343255555821953675 a texto.	**11.599** (1,01 %)	9.533 (1,81 %)
5.	Números TIP - Convertir el número 97898762911321937426853972581387258842685093808583900138206104309 14.07303643113467090522 a texto.	**8.586** (0,75 %)	7.049 (1,34 %)
6.	Números TIP - Convertir el número 4125123212312321255566565555124564564567897817191981848427817841654156841212456456456789781719198184842781784165415684 a texto.	**4.917** (0,43 %)	4.173 (0,79 %)
7.	Números TIP - Convertir el número 2879899876543234566778888899990090988787667564543543423134546576887998808787675645434243646575 8 a texto.	**4.694** (0,41 %)	4.028 (0,76 %)
8.	Números TIP - Lista número 5	**4.661** (0,41 %)	3.352 (0,63 %)
9.	Números TIP - Convertir el número 12345678912345678789234567891345678901234546578924346789 4 a texto.	**4.166** (0,36 %)	3.243 (0,61 %)
10.	Números TIP - Convertir el número MMXI a texto.	**4.129** (0,36 %)	2.453 (0,46 %)

Figura F.3.A

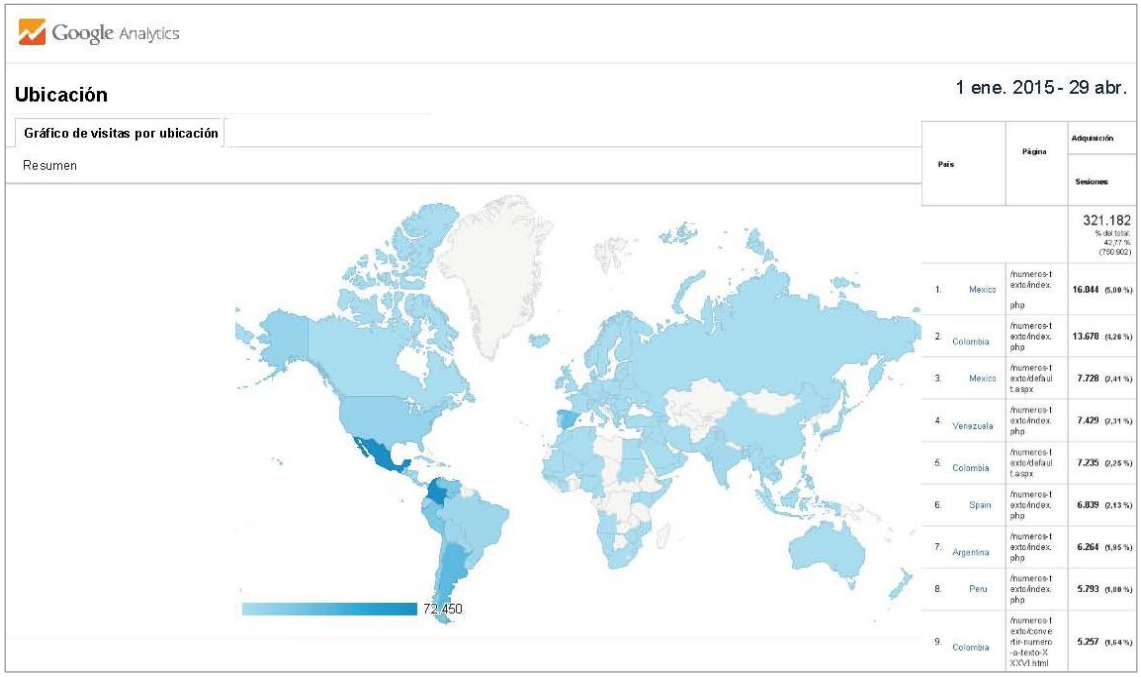


Figura F.3.B

La figura F.3.A muestra el número de visitas registradas desde Enero hasta Abril de 2015, ambos inclusive. En total, 1 150 382 visitas. Por otra parte, la figura F.3.B muestra la distribución de esas visitas en el mismo periodo a nivel mundial, siendo los países de habla hispana los que más visitas registran, especialmente México. Este dato confirma la utilidad de este tipo de servicios incluso para aquellas personas que, en teoría, dominan el idioma.

Estas estadísticas nos otorgan una justificación más para el desarrollo de este trabajo, y teniendo en cuenta que el idioma inglés es el más hablado en todo el mundo, es de esperar que el número de visitas a una web que utilice este servicio llegue a ser similar o incluso mayor que el registrado para la conversión a texto en español.

4. REQUISITOS E INSTALACIÓN.

Para la correcta instalación y posterior funcionamiento del servicio, es necesario disponer de un servidor con los siguientes componentes:

- *Microsoft Internet Information Services* (*IIS*) instalado localmente en el equipo.

- *.NET Framework* versión 3.5 o posterior.

Los pasos a seguir para instalar el servicio en el servidor son los siguientes:

1. Abrir el Administrador de *Internet Information Services* (*IIS*).

2. Hacer clic con el botón derecho en el Sitio Web por defecto y pulsar "Agregar aplicación"

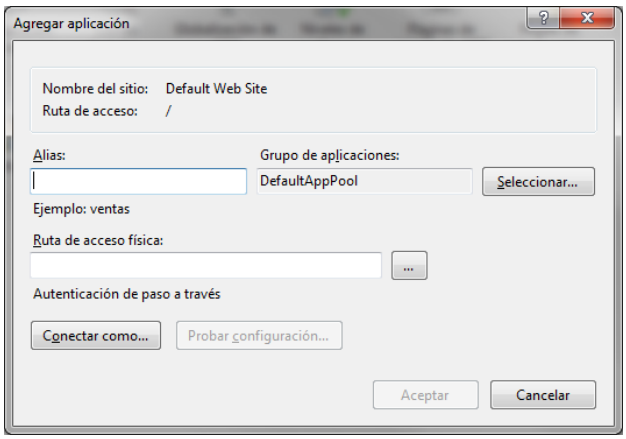


Figura F.4.A

3. Introducir un alias, por ejemplo "Conversor" y especificar la ruta de la carpeta que contiene los archivos del servicio. Si se produce un error al intentar conectar con el servicio, podría tratarse de un problema con los permisos de la carpeta. En tal caso, cambiar el usuario pulsando el botón "Conectar como…".

5. DISEÑO.

El proyecto está compuesto por una clase, llamada "Conversor", que a su vez contiene otras dos clases, y que devuelve distintos resultados alternativos de la conversión de un número a texto en inglés. Estos resultados vienen acompañados de una descripción o categoría que indica el tipo de resultado en cuestión, por ejemplo, si se trata de inglés británico o inglés americano.

5.1. Entrada y salidas del proceso.

El proceso comienza analizando el formato de un número escrito con dígitos y acaba devolviendo múltiples textos, cada uno resultado de un tipo distinto de conversión, o bien, en su caso, una lista con aquellos errores que hayan surgido al intentar hacer alguna conversión.

Para enlazar cualquier aplicación con este servicio, es necesario conocer el formato de la entrada y de las salidas que genera. A continuación pasamos a describirlas.

5.1.1. Entrada.

Como se detallará más adelante, la clase *Conversor* contiene un método llamado *Convertir*, que es el encargado de realizar todo el proceso, y que requiere un parámetro de entrada de tipo texto, que es el número que se desea convertir, escrito con dígitos numéricos. El número máximo de dígitos admitidos varía dependiendo del formato del número. Si el número está expresado en notación científica, se considerará tanto la longitud de su base y exponente, como su longitud una vez convertido a notación decimal. También se admiten números negativos, números fraccionarios, e incluso números romanos, en cuyo caso se realiza una conversión a número entero y se trata como tal.

El número introducido debe cumplir con al menos uno de los siguientes formatos, dentro de los cuales se debe seguir estrictamente el orden indicado de sus partes:

- **Número entero**

  Parte 1: [OPCIONAL]: un signo + o un signo - (y sólo uno).

  Parte 2: [OBLIGATORIA]: de 1 a 603 dígitos numéricos.

- **Número decimal**

  Parte 1: [OPCIONAL]: un signo + o un signo - (y sólo uno).

  Parte 2: [OBLIGATORIA]: de 1 a 603 dígitos numéricos.

  Parte 3: [OBLIGATORIA]: un punto (separador decimal) seguido de 1 a 306 dígitos numéricos.

- **Número fraccionario**

  Parte 1: [OPCIONAL]: un signo + o un signo - (y sólo uno).

  Parte 2: [OBLIGATORIA]: de 1 a 603 dígitos numéricos.

Parte 3: [OBLIGATORIA]: un signo / para separar el Numerador del Denominador.

Parte 4: [OPCIONAL]: un signo + o un signo - (y sólo uno).

Parte 5: [OBLIGATORIA]: de 1 a 603 dígitos numéricos.

- **Número en notación científica**

  Parte 1: [OPCIONAL]: un signo + o un signo - (y sólo uno).

  Parte 2: [OBLIGATORIA]: De 1 a 603 dígitos numéricos.

  Parte 3: [OPCIONAL]: un punto (separador decimal) seguido de 1 a 306 dígitos numéricos.

  Parte 4: [OBLIGATORIA]: una letra "e" o una "E" (y sólo una).

  Parte 5: [OPCIONAL]: un signo + o un signo - (y sólo uno).

  Parte 6: [OBLIGATORIA]: de 1 a 3 dígitos numéricos.

- **Número romano**

  Formato habitual de los números romanos. Se permite que las letras estén escritas en minúscula.

5.1.2. Salidas.

Como hemos comentado, el método *Convertir* realiza todo el proceso de conversión. Dicho proceso se descompone en varias tareas de conversión que se ejecutan simultáneamente, confluyendo todas en una integración de resultados que da lugar al conjunto de resultados definitivos, y que constituye el valor devuelto por el método.

El tipo de datos devuelto se corresponde con la subclase pública *Res_Def*, que será descrita más adelante. Si por algún motivo surgieran errores en el proceso, las descripciones de los mismos pueden obtenerse mediante la propiedad *Errores*, de tipo *ArrayList*, dentro de dicha subclase.

Por lo tanto, la aplicación que vaya a publicar los resultados debería comprobar previamente si se ha generado algún error.

5.1.3. Tipos de resultados.

Los tipos de resultados obtenidos dependen de la longitud y formato del número a convertir. El límite del servicio está fijado en los *1 000 centillones menos uno*, tanto en escala corta como en escala larga. Esto provoca una asimetría entre ambos tipos de resultados, ya que en escala corta sólo podemos llegar hasta los 306 dígitos (10^{306}), mientras que en escala larga podemos llegar nada más y nada menos que hasta los 603 dígitos (10^{603}).

La siguiente figura representa gráficamente la relación entre el número de dígitos y los resultados que se pueden obtener:

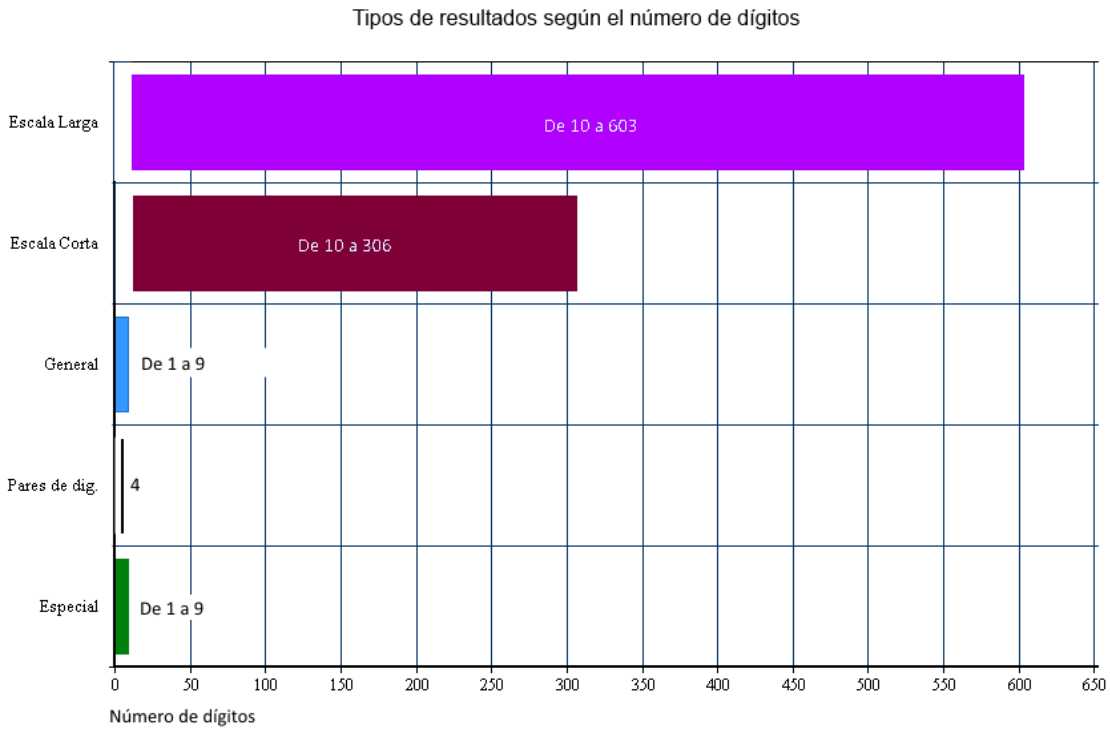


Figura F.5.1.3.A

A continuación veremos de una forma más detallada los distintos tipos de resultados que se pueden dar según el formato del número:

Formato del número	**Longitud (núm. de dígitos)**	**Tipo de resultado**						
		Especial (*)	General (**)		Por pares de dígitos (***)	Escala corta		Escala larga	
			Cardinal / Decimal	Ordinal	Cardinal	Cardinal / Decimal	Ordinal	Cardinal / Decimal	Ordinal
Entero	Entre 1 y 3	SÍ	SÍ	SÍ	NO	NO	NO	NO	NO
	4	SÍ	SÍ	SÍ	SÍ	NO	NO	NO	NO
	Entre 5 y 9	SÍ	SÍ	SÍ	NO	NO	NO	NO	NO
	Entre 10 y 306	NO	NO	NO	NO	SÍ	SÍ	SÍ	SÍ
	Entre 307 y 603	NO	NO	NO	NO	NO	NO	SÍ	SÍ
Decimal	P. Entera: Entre 1 y 3 P. Decimal: Entre 1 y 306	NO	SÍ	NO	NO	NO	NO	NO	NO
	P. Entera: Entre 4 y 9 P. Decimal: Entre 1 y 306	NO	SÍ	NO	NO	NO	NO	NO	NO
	P. Entera: Entre 10 y 306 P. Decimal: Entre 1 y 306	NO	NO	NO	NO	SÍ	NO	SÍ	NO
	P. Entera: Entre 307 y 603 P. Decimal: Entre 1 y 306	NO	NO	NO	NO	NO	NO	SÍ	NO
Fraccionario	Numerador: Entre 1 y 603 Denominador: Entre 1 y 603	Se tratan el Numerador y Denominador con formato de Entero y se combinan los resultados de Cardinal del Numerador con los de Ordinal del Denominador. Quedan excluidos los resultados de tipo “Especial” y “Por pares de dígitos”. Si se puede obtener el resultado de la división entre el Numerador y el Denominador, éste se trata con formato Decimal y se añaden sus resultados a los del Número Fraccionario.							
Notación científica	Se convierte el número a Notación decimal y se trata como tal.								
Romano	Se convierte el número a Entero y se trata como tal.								

Tabla T.5.1.3.A

(*) Resultado especial: A determinados números se les asigna un nombre relacionado con el contexto en el que se utilizan, por ejemplo, en deportes.
(**) Resultado general: Consideramos un resultado como “general” cuando el resultado en Escala Corta es el mismo que en Escala Larga.
(***) Por pares de dígitos: En inglés americano, los números de 4 dígitos suelen nombrarse por pares de dígitos en determinados contextos. Por ejemplo: 1974 → “Nineteen seventy-four”

5.2.Vista general del proceso de conversión.

Antes de describir la estructura de la clase *Conversor*, conviene comprender cómo se ha planteado el mecanismo de conversión. Existen varios "caminos" alternativos hasta la obtención de resultados, dependiendo del formato del número introducido.

La siguiente figura ilustra el diagrama de flujo de la conversión:

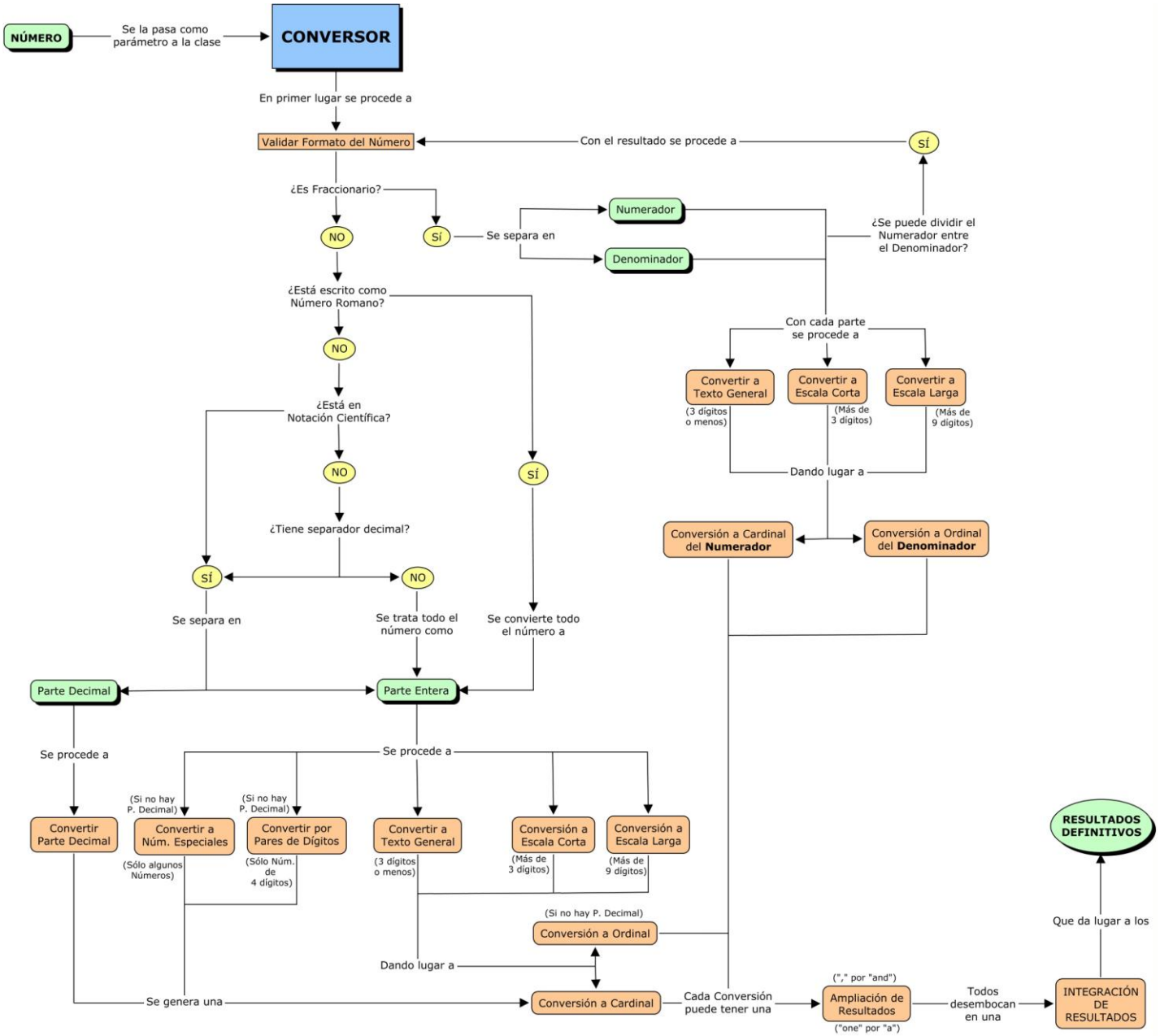

NÚMERO
Se la pasa como parámetro a la clase
CONVERSOR
En primer lugar se procede a
Validar Formato del Número
Con el resultado se procede a
SÍ
¿Es Fraccionario?
NO
SÍ
Se separa en
Numerador
Denominador
¿Se puede dividir el Numerador entre el Denominador?
¿Está escrito como Número Romano?
NO
¿Está en Notación Científica?
NO
¿Tiene separador decimal?
SÍ
NO
SÍ
Se separa en
Se trata todo el número como
Se convierte todo el número a
Con cada parte se procede a
Convertir a Texto General
(3 dígitos o menos)
Convertir a Escala Corta
(Más de 3 dígitos)
Convertir a Escala Larga
(Más de 9 dígitos)
Dando lugar a
Conversión a Cardinal del Numerador
Conversión a Ordinal del Denominador
Parte Decimal
Parte Entera
Se procede a
Se procede a
(Si no hay P. Decimal)
Convertir Parte Decimal
Convertir a Núm. Especiales
(Sólo algunos Números)
(Si no hay P. Decimal)
Convertir por Pares de Dígitos
(Sólo Núm. de 4 dígitos)
Convertir a Texto General
(3 dígitos o menos)
Conversión a Escala Corta
(Más de 3 dígitos)
Conversión a Escala Larga
(Más de 9 dígitos)
(Si no hay P. Decimal)
Conversión a Ordinal
Dando lugar a
Se genera una
Conversión a Cardinal
Cada Conversión puede tener una
(“,” por “and”)
Ampliación de Resultados
(“one” por “a”)
Todos desembocan en una
INTEGRACIÓN DE RESULTADOS
Que da lugar a los
RESULTADOS DEFINITIVOS

Figura F.5.2.A

El proceso puede describirse brevemente en los siguientes pasos:

1. Desde una aplicación externa se crea una instancia de la clase *Conversor*.

2. Se llama al método *Convertir* de dicha clase, pasándole como parámetro el número a convertir.

A partir de ese momento, el método *Convertir* realiza las siguientes tareas:

3. Se valida el formato del número. Si el formato no es correcto, se genera un error y se finaliza el proceso. En caso contrario, se analiza el número de la siguiente forma:

   - Si no es un formato numérico: se comprueba si se trata de un **número romano**, en cuyo caso se convierte a entero. Si no, se genera un error y finaliza el proceso.

   - Si se detecta un signo de división (considerado como indicador de que es un número fraccionario) se descompone en **numerador** y **denominador**. Si la longitud de ambos permite obtener el resultado de la división, se procede a validar el formato de dicho resultado, siguiendo estos mismos pasos.

   - Si se detecta un punto (considerado como separador decimal), se descompone en **parte entera** y **parte decimal.**

   - También se admiten **números negativos** y **notación científica**. En este último caso, se realiza una conversión a notación decimal. Si un número no fraccionario no tiene parte decimal, se considera como parte entera el número completo.

4. Con cada una de las partes devueltas por la validación (parte entera y parte decimal cuando el número no es fraccionario, o bien numerador y denominador cuando sí lo es), se realizan simultáneamente tareas de conversión a texto en distintos formatos, según proceda: números especiales, escala corta, escala larga, decimal, etc. Estas conversiones pueden, a su vez, generar diversos tipos de resultados, como texto cardinal, texto ordinal, números romanos, etc. En la mayor parte de los casos se amplía el conjunto de resultados obtenidos, por ejemplo sustituyendo *comas* por palabras “and”, que es la forma utilizada

habitualmente en inglés británico, o bien sustituyendo “one” por “a” al comienzo del texto, si se trata de un número que empieza por un dígito de unidad o centena igual a uno.

5. Tras esperar a que finalicen todas las conversiones, se envían los resultados obtenidos a un proceso de *Integración de Resultados*, que se encarga de combinar unos con otros de forma coherente y devolverlos agrupados por categorías, que indican qué tipo de resultados contienen.

De nuevo, desde la aplicación externa:

6. Se comprueba si ha habido algún error en la conversión. Si no lo hay, se obtienen los resultados agrupados por categorías que fueron devueltos por el proceso de integración.

5.3.Estructura de la clase *Conversor*.

Tal como hemos adelantado, el servicio consta únicamente de una clase llamada *Conversor*, que encapsula todos los componentes necesarios para el proceso de conversión.

En la siguiente figura podemos ver un diagrama general de su estructura:

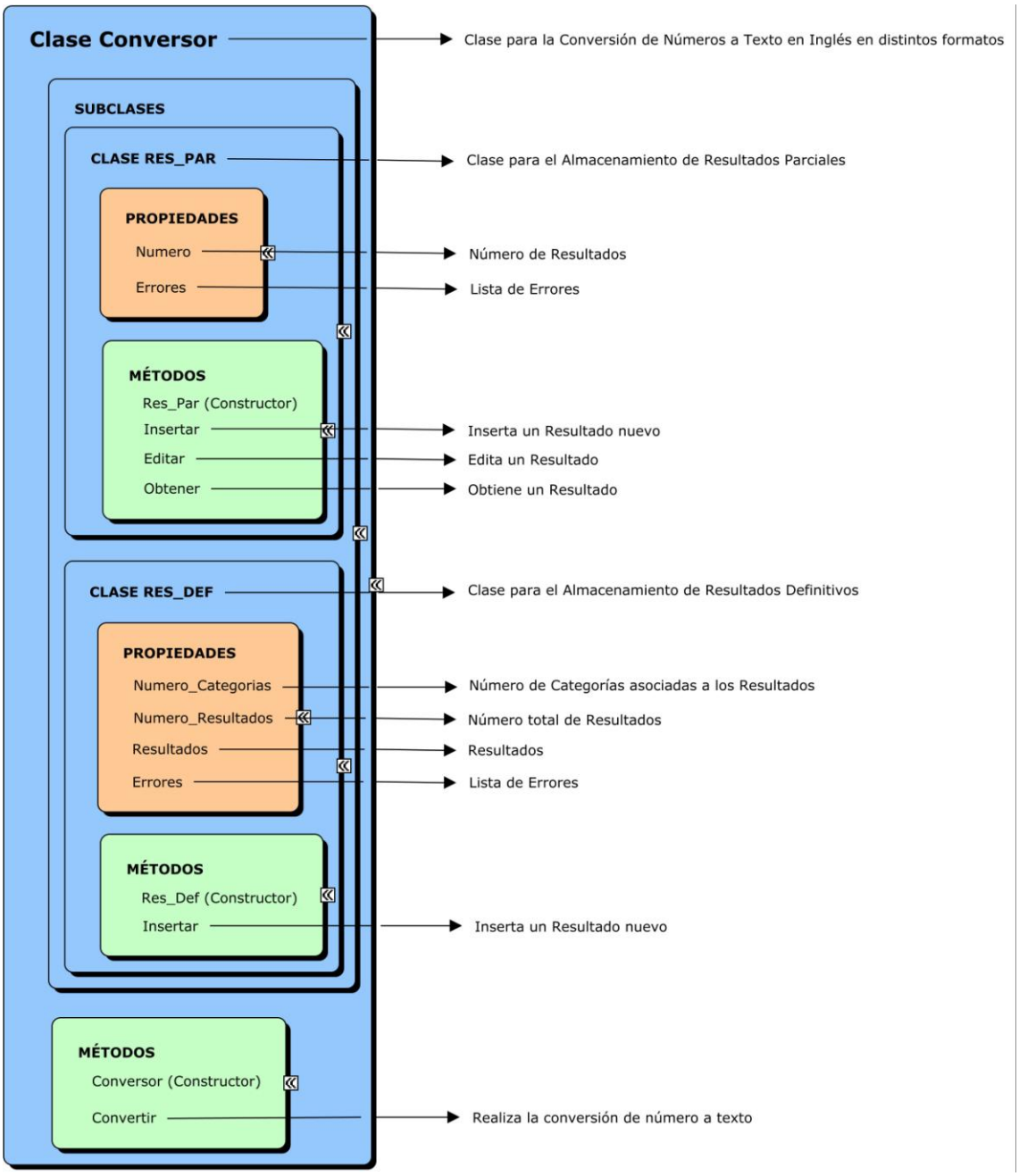


Figura F.5.3.A

Como se puede apreciar, contiene dos subclases: ***Res_Par***, que es privada y se encarga del almacenamiento de resultados parciales, y ***Res_Def***, que es pública y se encarga del almacenamiento de los resultados definitivos.

Por otra parte, contiene dos métodos: el constructor, que no requiere parámetros, y el método *Convertir*, que es el encargado de inicializar y lanzar el proceso de conversión. Este método devuelve una instancia de la clase *Res_Def* con los resultados definitivos de la conversión.

Aunque no se muestran en la figura, por tratarse de una descripción a nivel general, la clase *Conversor* también consta de **definiciones de tipos, variables globales, constantes y funciones locales**.

A continuación se detallan todos los componentes.

5.3.1. Clase *Res_Par*.

Esta subclase es la encargada de almacenar los **resultados parciales** que generan las distintas tareas de conversión. Es de ámbito privado, y por tanto sólo es accesible desde dentro de la clase *Conversor*.

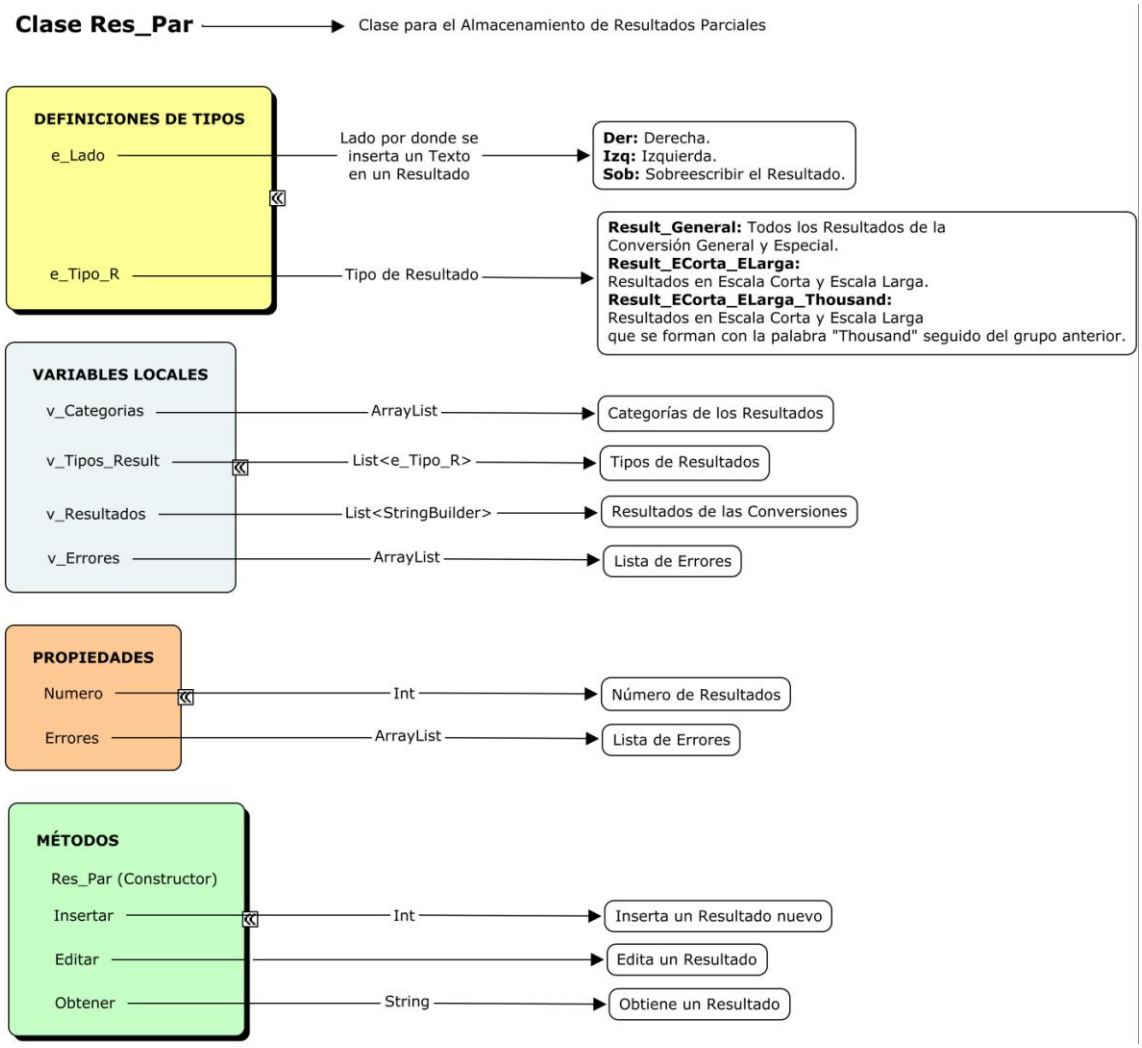


Figura F.5.3.1.A

- El tipo enumerado *e_Lado* indica el lado por el que se produce la concatenación en la cadena de un resultado (izquierda, derecha o sobreescribir).

- El tipo enumerado *e_Tipo_R* indica el tipo de resultado:
    - *Result_General*: todos los resultados en la conversión a número general y/o especial.
    - *Result_ECorta_ELarga*: resultados en escala corta y escala larga.
    - *Result_ECorta_ELarga_Thousand*: resultados en escala corta y escala larga que se forman con la palabra “thousand” seguido del grupo anterior.

- Además del constructor, esta clase contiene tres métodos:
    - *Insertar*: inserta un nuevo resultado.
    - *Editar*: edita un resultado insertándole un texto por la izquierda o derecha, o bien reemplazándolo completamente por el texto.
    - *Obtener*: devuelve un resultado, opcionalmente junto con su categoría y tipo.

5.3.2. Clase *Res_Def*.

Esta subclase es la encargada de almacenar los **resultados definitivos**, procedentes de la integración de resultados parciales. Es de ámbito público, y por tanto accesible desde una aplicación externa.

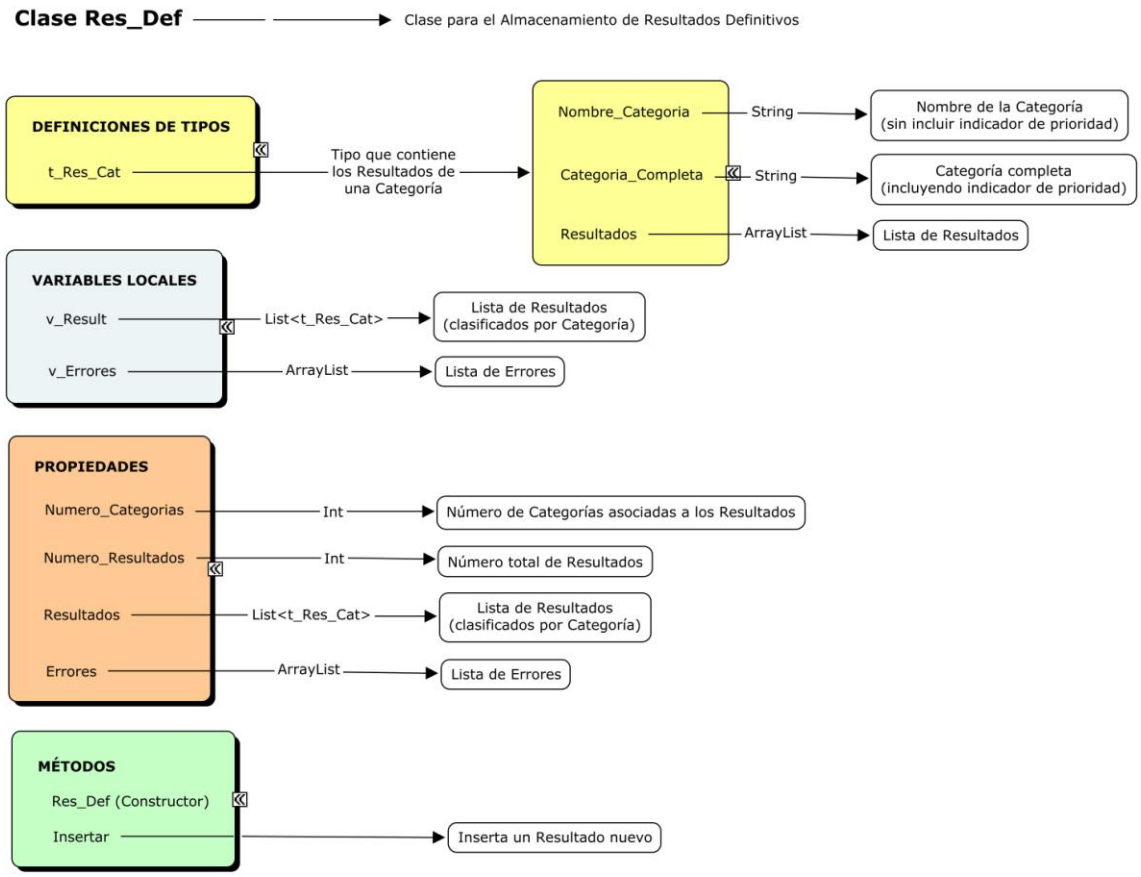


Figura F.5.3.2.A

El tipo estructurado *t_Res_Cat* permite almacenar los resultados por categoría. Está compuesto por tres campos:

- *Nombre_Categoria* (*String*): indica el nombre de la categoría.
- *Categoria_Completa* (*String*): indica el nombre de la categoría incluyendo un indicador de prioridad, cuya utilidad comentaremos a continuación.
- *Resultados* (*ArrayList*): contiene la lista de resultados de la categoría.

Como se puede apreciar en la figura, existen dos propiedades que devuelven un valor entero:

- *Numero_Categorias*: Número de categorías asociadas a los resultados.
- *Numero_Resultados*: Número total de resultados.

La propiedad *Resultados* contiene una lista con los resultados clasificados por categoría. Se trata de una lista de tipo *t_Res_Cat*, que como vimos en la figura anterior está compuesta por los campos:

- *Nombre_Categoria*: Nombre de la categoría asociada a los Resultados. Cada categoría puede contener subcategorías, y éstas a su vez más subcategorías. El carácter # se usa como delimitador de subcategorías, de tal forma que el valor de esta propiedad consiste en una cadena de palabras separadas por #.

  Por ejemplo, el valor "Cardinal#Escala Corta#BrE" indica la siguiente jerarquía de categorías:

  - Cardinal ……(Categoría Principal)
    - ◊ Escala Corta……..(Subcategoría de Cardinal)
      - BrE (British English)……..(Subcategoría de Escala Corta)

- *Categoria_Completa*: Este campo contiene el mismo valor que *Nombre_Categoria*, pero incluyendo un indicador de prioridad. El nombre de cada categoría viene definido mediante una constante en el código con el valor [*n*]*Nombre_Categoria*, donde [*n*] es el indicador de prioridad. Dicho indicador permite que las categorías, con sus correspondientes resultados, se almacenen y devuelvan en un orden concreto. Por ejemplo, los resultados de la categoría *[1]Cardinal#[1][Escala corta]* aparecen antes en la lista que los de la categoría *[1]Cardinal#[2][Escala larga]* o que los de la categoría *[2]Ordinal#[1][Escala corta]*.

- *Resultados*: Array con los resultados correspondientes a la categoría indicada.

- La propiedad *Errores*, que normalmente está vacía, puede contener una lista con los errores que se pudieran originar durante el proceso de conversión.

- Además del constructor, esta clase contiene el método *Insertar*, que se utiliza para insertar un resultado nuevo. Los resultados se insertan en una lista de tipo *t_Res_Cat*, de forma que quedan agrupados y

ordenados por categoría siguiendo el criterio comentado en la descripción del campo *Categoria_Completa*.

Tal como comentamos en la sección "Entrada y Salidas del Proceso", el método *Convertir* de la clase *Conversor* devuelve una instancia de la clase *Res_Def* con los resultados definitivos. Conviene, por tanto, poner ejemplos reales de lo que devolvería dicho método para algunos números, facilitando así la comprensión de la estructura obtenida.

A continuación se muestra la estructura de resultados obtenida por el método *Convertir* para los siguientes números:

a. **530,765,255,405** (figura F.5.3.2.B)

b. **-325/621** (figura F.5.3.2.C)

c. **2.51907e24** (figura F.5.3.2.D)

Resultados para el número 530,765,255,405

v_Result_Def.Resultados	Count = 7
[0]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[1]Cardinal#[1]Escala corta#[1]BrE"
Nombre_Categoria	"Cardinal#Escala corta#BrE"
Resultados	Count = 2
[0]	"five hundred and thirty billion, seven hundred and sixty-five million, two hundred and fifty-five thousand and four hundred and five"
[1]	"five hundred and thirty thousand million, seven hundred and sixty-five million, two hundred and fifty-five thousand and four hundred and five"
[1]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[1]Cardinal#[1]Escala corta#[2]AmE"
Nombre_Categoria	"Cardinal#Escala corta#AmE"
Resultados	Count = 2
[0]	"five hundred thirty billion, seven hundred sixty-five million, two hundred fifty-five thousand, four hundred five"
[1]	"five hundred thirty thousand million, seven hundred sixty-five million, two hundred fifty-five thousand, four hundred five"
[2]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[1]Cardinal#[2]Escala larga#[1]BrE"
Nombre_Categoria	"Cardinal#Escala larga#BrE"
Resultados	Count = 2
[0]	"five hundred and thirty milliard, seven hundred and sixty-five million, two hundred and fifty-five thousand and four hundred and five"
[1]	"five hundred and thirty thousand million, seven hundred and sixty-five million, two hundred and fifty-five thousand and four hundred and five"
[3]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[2]Ordinal#[1]Escala corta#[1]BrE"
Nombre_Categoria	"Ordinal#Escala corta#BrE"
Resultados	Count = 2
[0]	"five hundred and thirty billion, seven hundred and sixty-five million, two hundred and fifty-five thousand and four hundred and fifth"
[1]	"five hundred and thirty thousand million, seven hundred and sixty-five million, two hundred and fifty-five thousand and four hundred and fifth"
[4]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[2]Ordinal#[1]Escala corta#[2]AmE"
Nombre_Categoria	"Ordinal#Escala corta#AmE"
Resultados	Count = 2
[0]	"five hundred thirty billion, seven hundred sixty-five million, two hundred fifty-five thousand, four hundred fifth"
[1]	"five hundred thirty thousand million, seven hundred sixty-five million, two hundred fifty-five thousand, four hundred fifth"
[5]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[2]Ordinal#[2]Escala larga#[1]BrE"
Nombre_Categoria	"Ordinal#Escala larga#BrE"
Resultados	Count = 2
[6]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[5]Romano"
Nombre_Categoria	"Romano"
Resultados	Count = 1

Figura F.5.3.2.B

Resultados para el número -325/621 = -0.523349436392915

v_Result_Def.Resultados	Count = 6
[0]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[3]Decimal#[1]General#[1]Sin 'point'#[1]BrE"
Nombre_Categoria	"Decimal#General#Sin 'point'#BrE"
Resultados	Count = 1
[0]	"minus five hundred twenty-three trillion, three hundred forty-nine billion, four hundred thirty-six million, three hundred ninety-two thousand, nine hundred fifteen quadrillionths"
[1]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[3]Decimal#[1]General#[1]Sin 'point'#[2]AmE"
Nombre_Categoria	"Decimal#General#Sin 'point'#AmE"
Resultados	Count = 1
[0]	"negative five hundred twenty-three trillion, three hundred forty-nine billion, four hundred thirty-six million, three hundred ninety-two thousand, nine hundred fifteen quadrillionths"
[2]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[3]Decimal#[1]General#[2]Con 'point'#[1]BrE"
Nombre_Categoria	"Decimal#General#Con 'point'#BrE"
Resultados	Count = 1
[0]	"minus zero point five two three three four nine four three six three nine two nine one five"
[3]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[3]Decimal#[1]General#[2]Con 'point'#[2]AmE"
Nombre_Categoria	"Decimal#General#Con 'point'#AmE"
Resultados	Count = 1
[0]	"negative zero point five hundred twenty-three trillion, three hundred forty-nine billion, four hundred thirty-six million, three hundred ninety-two thousand, nine hundred fifteen"
[4]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[4]Fraccionario#[1]General#[1]BrE"
Nombre_Categoria	"Fraccionario#General#BrE"
Resultados	Count = 1
[0]	"minus three hundred and twenty-five six hundred and twenty-firsts"
[5]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[4]Fraccionario#[1]General#[2]AmE"
Nombre_Categoria	"Fraccionario#General#AmE"
Resultados	Count = 1
[0]	"negative three hundred twenty-five six hundred twenty-firsts"

Figura F.5.3.2.C

Resultados para el número 2.51907e24

v_Result_Def.Resultados	Count = 6
[0]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[1]Cardinal#[1]Escala corta#[1]BrE"
Nombre_Categoria	"Cardinal#Escala corta#BrE"
Resultados	Count = 2
[0]	"two septillion, five hundred and nineteen sextillion, seventy quintillion"
[1]	"two thousand sextillion, five hundred and nineteen thousand quintillion, seventy thousand quadrillion"
[1]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[1]Cardinal#[1]Escala corta#[2]AmE"
Nombre_Categoria	"Cardinal#Escala corta#AmE"
Resultados	Count = 2
[0]	"two septillion, five hundred nineteen sextillion, seventy quintillion"
[1]	"two thousand sextillion, five hundred nineteen thousand quintillion, seventy thousand quadrillion"
[2]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[1]Cardinal#[2]Escala larga#[1]BrE"
Nombre_Categoria	"Cardinal#Escala larga#BrE"
Resultados	Count = 2
[0]	"two quadrillion, five hundred and nineteen trilliard, seventy trillion"
[1]	"two million trillion, five hundred and nineteen thousand trillion, seventy million billion"
[3]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[2]Ordinal#[1]Escala corta#[1]BrE"
Nombre_Categoria	"Ordinal#Escala corta#BrE"
Resultados	Count = 2
[0]	"two septillion, five hundred and nineteen sextillion, seventy quintillionth"
[1]	"two thousand sextillion, five hundred and nineteen thousand quintillion, seventy thousand quadrillion"
[4]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[2]Ordinal#[1]Escala corta#[2]AmE"
Nombre_Categoria	"Ordinal#Escala corta#AmE"
Resultados	Count = 2
[0]	"two septillion, five hundred nineteen sextillion, seventy quintillionth"
[1]	"two thousand sextillion, five hundred nineteen thousand quintillion, seventy thousand quadrillion"
[5]	{Conversor_Numeros_Texto.Conversor.Res_Def.t_Res_Cat}
Categoria_Completa	"[2]Ordinal#[2]Escala larga#[1]BrE"
Nombre_Categoria	"Ordinal#Escala larga#BrE"
Resultados	Count = 2
[0]	"two quadrillion, five hundred and nineteen trilliard, seventy trillionth"
[1]	"two million trillion, five hundred and nineteen thousand trillion, seventy million billion"

Figura F.5.3.2.D

5.3.3. Definiciones de tipos.

La clase *Conversor* contiene las siguientes declaraciones de tipos de datos:

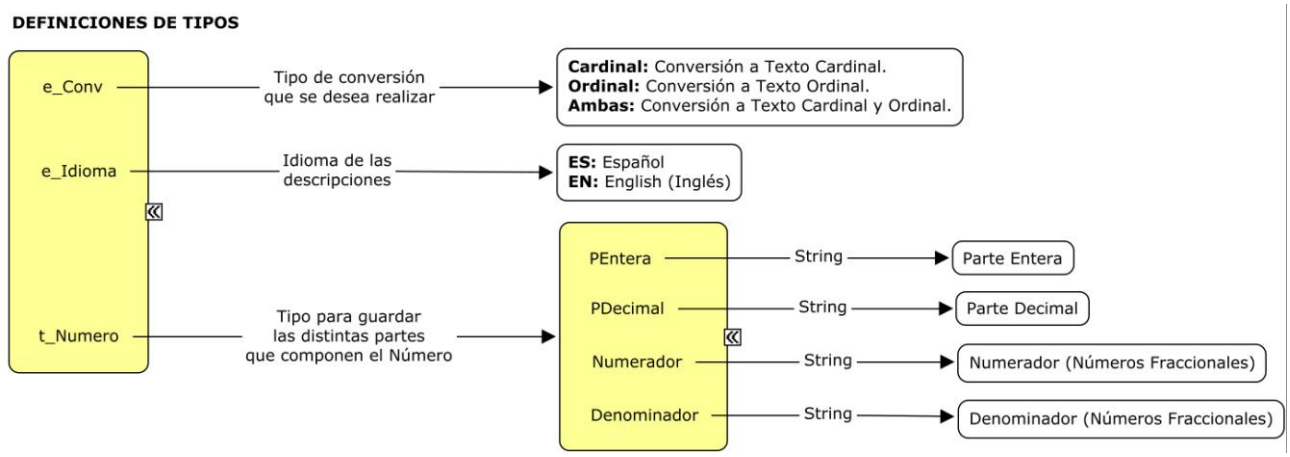


Figura F.5.3.3.A

- El tipo enumerado *e_Conv* indica el tipo de conversión a realizar, siendo sus posibles valores *Cardinal*, *Ordinal* o *Ambas*. Dependiendo de las características del número a convertir se ejecutan unos tipos de conversión u otros.

- El tipo enumerado *e_Idioma* indica el idioma de las descripciones, tanto de los resultados como de los errores.

- El tipo estructurado *t_Numero* se utiliza para guardar las distintas partes que pueden componer un número. Sus campos son *PEntera* (Parte Entera) y *PDecimal* (Parte Decimal), que se utilizan para números enteros y decimales, y *Numerador* y *Denominador*, que se utilizan para números fraccionarios.

5.3.4. Variables globales.

La clase *Conversor* contiene las siguientes variables globales:

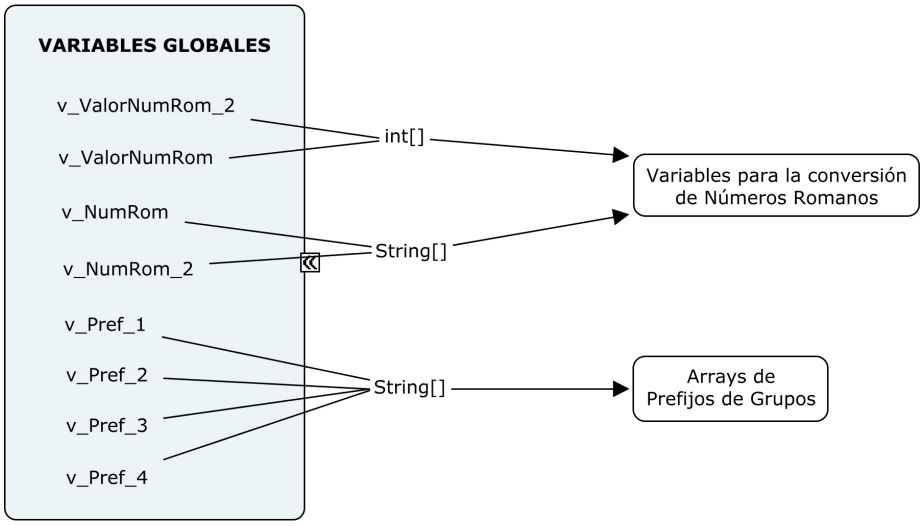


Figura F.5.3.4.A

- Los *arrays* *v_ValorNumRom*, *v_ValorNumRom_2*, *v_NumRom* y la variable *v_NumRom_2* se utilizan para la conversión de números romanos a números enteros y viceversa.

- Los *arrays* *v_Pref_1*, *v_Pref_2*, *v_Pref_3* y *v_Pref_4* se utilizan como prefijos en la construcción dinámica del *array* de **Textos de Grupos de Dígitos**, que se explicará detalladamente en la sección de Desarrollo.

5.3.5. Funciones locales.

A continuación se detallan las funciones locales de la clase:

Nombre	Validar_Formato
Descripción	Comprueba si el texto original tiene un formato correcto, y, en caso afirmativo, separa el número en parte entera y parte decimal, o bien en numerador y denominador.
Argumentos de Entrada	• ae_Numero (String): Texto original del número (dígitos) • ae_Idioma (e_Idioma): Idioma de las descripciones de error.
Argumentos de Salida	• as_Numero (t_Numero): Número separado en sus correspondientes partes. • as_Negativo (Boolean): Indica si el número es negativo. • as_Errores (ArrayList): Lista de errores.
Valor devuelto	(Boolean): Verdadero si el formato es correcto, y falso en caso contrario.

Tabla T.5.3.5.A

Nombre	Convertir_Notacion_Decimal
Descripción	Convierte un número de Notación Científica a Notación Decimal.
Argumentos de Entrada	• ae_Numero (String): Número a convertir.
Argumentos de Salida	• as_Errores (ArrayList): Lista de errores.
Valor devuelto	(String): Número en Notación Decimal.

Tabla T.5.3.5.B

Nombre	Concatenar_And
Descripción	Concatena la palabra “and” en un texto de número.
Argumentos de Entrada	• ae_Texto_Numero (String): Texto de Número. • ae_Texto_Grupos (String[] – Opcional): Textos de Grupos de Dígitos.
Argumentos de Salida	• as_Errores (ArrayList): Lista de errores.
Valor devuelto	(String): Texto de Número con palabras “and” concatenadas.

Tabla T.5.3.5.C

Nombre	Eliminar_Ceros_NS
Descripción	Elimina los ceros no significativos de un número: si es una parte entera los que estén por la izquierda; si es una parte decimal, los que estén por la derecha. Además, si el número comienza por un signo + o un -, elimina dicho signo.
Argumentos de Entrada	• ae_Numero (String): Número. • ae_Decimal (Boolean): Indica si se trata de una parte decimal.
Argumentos de Salida	• as_Errores (ArrayList): Lista de errores.
Valor devuelto	(String): Número sin signo ni ceros no significativos.

Tabla T.5.3.5.D

Nombre	Obtener_Grupo_Cardinal
Descripción	Obtiene la conversión a cardinal de un grupo de 3 dígitos (o menos).
Argumentos de Entrada	• ae_Digitos_Grupo (String): Dígitos del Grupo.
Argumentos de Salida	• as_Resultado (String): Resultado de la Conversión. • as_Errores (ArrayList): Lista de errores.
Valor devuelto	(Boolean): Verdadero si la conversión tiene éxito, y falso en caso contrario.

Tabla T.5.3.5.E

Nombre	Obtener_Grupo_Ordinal
Descripción	Obtiene la conversión a ordinal de un grupo de 3 dígitos (o menos).
Argumentos de Entrada	• ae_Digitos_Grupo (String): Dígitos del Grupo.
Argumentos de Salida	• as_Resultado (String): Resultado de la Conversión. • as_Errores (ArrayList): Lista de errores.
Valor devuelto	(Boolean): Verdadero si la conversión tiene éxito, y falso en caso contrario.

Tabla T.5.3.5.F

Nombre	Cargar_Textos_Grupos
Descripción	Carga los textos de grupos en un *array*.
Argumentos de Entrada	• ae_NumGrupos (int): Número de Grupos.
Argumentos de Salida	• as_Textos_Grupos (String[]): Array con los Textos de los Grupos. • as_Errores (ArrayList): Lista de errores.
Valor devuelto	Ninguno.

Tabla T.5.3.5.G

Nombre	Gen_Result_Especial
Descripción	Genera los resultados de la conversión de la parte entera a números especiales.
Argumentos de Entrada	• ae_ PEntera (String): Parte entera del número. • ae_Idioma (e_Idioma): Idioma de las descripciones.
Argumentos de Salida	Ninguno.
Valor devuelto	(Res_Par): Resultados de la conversión.

Tabla T.5.3.5.H

Nombre	Gen_Result_Par_Dig
Descripción	Genera los resultados de la conversión de la parte entera a pares de dígitos.
Argumentos de Entrada	• ae_ PEntera (String): Parte entera del número. • ae_Idioma (e_Idioma): Idioma de las descripciones.
Argumentos de Salida	Ninguno.
Valor devuelto	(Res_Par): Resultados de la conversión.

Tabla T.5.3.5.I

Nombre	Gen_Result_General
Descripción	Genera los resultados de la conversión de la parte entera a texto general.
Argumentos de Entrada	• ae_ PEntera (String): Parte entera del número. • ae_Tipo_Result (e_Conv): Tipos de resultados a obtener. • ae_Negativo (Boolean): Indica si el número es negativo.
Argumentos de Salida	Ninguno.
Valor devuelto	(Res_Par): Resultados de la conversión.

Tabla T.5.3.5.J

Nombre	Gen_Result_ECorta
Descripción	Genera los resultados de la conversión de la parte entera a escala corta.
Argumentos de Entrada	• ae_ PEntera (String): Parte entera del número. • ae_Tipo_Result (e_Conv): Tipos de resultados a obtener. • ae_Negativo (Boolean): Indica si el número es negativo. • ae_Idioma (e_Idioma): Idioma de las descripciones.
Argumentos de Salida	Ninguno.
Valor devuelto	(Res_Par): Resultados de la conversión.

Tabla T.5.3.5.K

Nombre	Gen_Result_ELarga
Descripción	Genera los resultados de la conversión de la parte entera a escala larga.
Argumentos de Entrada	• ae_PEntera (String): Parte entera del número. • ae_Tipo_Result (e_Conv): Tipos de resultados a obtener. • ae_Negativo (Boolean): Indica si el número es negativo. • ae_Idioma (e_Idioma): Idioma de las descripciones.
Argumentos de Salida	Ninguno.
Valor devuelto	(Res_Par): Resultados de la conversión.

Tabla T.5.3.5.L

Nombre	Gen_Result_Decimal
Descripción	Genera los resultados de la conversión de la parte decimal.
Argumentos de Entrada	• ae_PDecimal (String): Parte decimal del número. • ae_Idioma (e_Idioma): Idioma de las descripciones.
Argumentos de Salida	Ninguno.
Valor devuelto	(Res_Par): Resultados de la conversión.

Tabla T.5.3.5.M

Nombre	Integrar_Resultados
Descripción	Integra y combina los resultados parciales obtenidos y los devuelve en un único conjunto.
Argumentos de Entrada	• ae_Idioma (e_Idioma): Idioma de las descripciones. • ae_Negativo (Boolean): Indica si el número es negativo. • ae_Result_PEntera_Num (List<Res_Par>): Resultados de la parte entera o numerador si el número es fraccionario. • ae_Result_PDecimal (Res_Par - Opcional): Resultados de la parte decimal. • ae_Result_Den (List<Res_Par> - Opcional): Resultados del denominador (números fraccionarios). • ae_Numerador (String – Opcional): Numerador (números fraccionarios). • ae_Denominador (String – Opcional): Denominador (números fraccionarios). • ae_Romanos (String – Opcional): Números romanos.
Argumentos de Salida	Ninguno.
Valor devuelto	(Res_Def): Conjunto de Resultados Definitivos.

Tabla T.5.3.5.N

Nombre	Conv_Entero_Romano
Descripción	Convierte un número entero a romano.
Argumentos de Entrada	• ae_PEntera (String): Parte entera del número.
Argumentos de Salida	• as_Errores (ArrayList): Lista de errores.
Valor devuelto	(String): Número escrito en romano.

Tabla T.5.3.5.O

Nombre	Conv_Romano_Entero
Descripción	Convierte un número romano a entero.
Argumentos de Entrada/Salida	• as_NumRomano (String): Número escrito en romano. • as_Errores (ArrayList): Lista de errores.
Valor devuelto	(int): Valor del número.

Tabla T.5.3.5.P

Nombre	Formatear_Parte_Numero
Descripción	Formatea una parte de un número, separando cada grupo de tres dígitos con un espacio.
Argumentos de Entrada	• ae_Parte (String): Parte del número.
Valor devuelto	(String): Número formateado.

Tabla T.5.3.5.Q

Nombre	Obtener_Digitos_Decimales
Descripción	Obtiene la conversión a texto de un conjunto de dígitos decimales (formato BrE con “point”).
Argumentos de Entrada	• ae_Digitos (String): Dígitos decimales.
Argumentos de Salida	• as_Resultado (String): Resultado de la conversión. • as_Errores (ArrayList): Lista de errores.
Valor devuelto	(Boolean): Verdadero si la conversión es correcta.

Tabla T.5.3.5.R

Nombre	Logs
Descripción	Escribe en un fichero las consultas realizadas al servicio.
Argumentos de Entrada	• ae_Numero (String): Número consultado. • ae_Error (int): Código de error.
Valor devuelto	Ninguno.

Tabla T.5.3.5.S

5.3.6. Métodos.

Nombre	Conversor (Constructor)
Descripción	Constructor implícito de la clase (no implementado en el código).
Argumentos de Entrada	Ninguno.
Argumentos de Salida	Ninguno.
Valor devuelto	Instancia de la clase Conversor.

Tabla T.5.3.6.A

Nombre	Convertir
Descripción	Crea una instancia de la clase e inicializa variables.
Argumentos de Entrada	• ae_Numero (String): Número a convertir en texto. • ae_Idioma (e_Idioma): Idioma de las descripciones.
Argumentos de Salida	• as_Numero_Formateado (String): Número formateado. • as_NumCategorias (int): Número de categorías de los resultados obtenidos. • as_NumResultados (int): Número total de resultados obtenidos.
Valor devuelto	(Res_Def): Conjunto de Resultados Definitivos.

Tabla T.5.3.6.B

6. DESARROLLO.

Para la implementación del servicio se ha empleado el lenguaje C#. El código desarrollado contempla las siguientes características:

- Programación orientada a objetos: implementación de clases.
- Expresiones regulares: uso de patrones de caracteres para la validación del formato del número introducido por el usuario.
- Concurrencia: ejecución de múltiples tareas de conversión simultáneamente, con lo que se obtiene mayor velocidad en la obtención de resultados.
- Fraccionamiento y concatenación: fraccionamiento del número que se desea convertir en grupos de tres dígitos para obtener resultados por partes y concatenación de dichas partes.
- Construcción dinámica de términos con prefijos y sufijos: construcción, en tiempo de ejecución, de los términos asociados a los grupos de tres dígitos usando una serie de prefijos y sufijos.
- Integración de resultados: combinación e integración de los resultados parciales para obtener los resultados definitivos.
- Mejora de rendimiento a través de la clase *StringBuilder*: uso de objetos de esta clase, en lugar de objetos tipo *String*, en aquellos casos donde hay repetidas modificaciones de cadenas de texto.

Veamos ahora con detalle de qué forma se han empleado dichas características.

6.1. Programación orientada a objetos.

Tras explicar el diseño de la clase *Conversor*, queda claro que se trata de una programación orientada a objetos, donde encontramos métodos y propiedades entre otros componentes. Además encontramos un anidamiento de clases, ya que la clase *Conversor* contiene otras dos subclases: *Res_Par* y *Res_Def*.

6.2. Expresiones regulares.

Antes de iniciar las tareas de conversión sobre un número, es indispensable verificar que el formato de dicho número es correcto. Una forma de hacerlo, sin tener que desarrollar demasiado código, es usando *expresiones regulares*.

Una expresión regular es una secuencia de caracteres que forma un patrón de búsqueda, principalmente utilizada para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones. La secuencia puede estar formada por uno o más literales de carácter, operadores o estructuras.

La lista de caracteres que se pueden utilizar para definir una expresión regular es bastante extensa, por lo que no procede detallarla por completo en este documento. Sin embargo, sí que podemos desglosar y explicar las expresiones regulares utilizadas para verificar el formato del número introducido por el usuario. La función *Validar_Formato*, de la clase *Conversor*, utiliza tres patrones distintos para esta verificación; uno para números no fraccionarios, otro para números fraccionarios, y otro para números romanos. A continuación pasamos a detallarlos:

- Números no fraccionarios:

  (\+?|\-?)\d{1,603}(\.\d{1,306})?((e|E)(\+?|\-?)\d{1,3})?

Caracteres	Opcional	Descripción
(\+?|\-?)	Sí	Un signo + o un signo – (y sólo uno).
\d{1,603}	No	De 1 a 603 dígitos numéricos.
(\.\d{1,306})?	Sí	Un punto (separador decimal) seguido de 1 a 306 dígitos numéricos.
((e|E)(\+?|\-?)\d{1,3})?	Sí	Una letra "e" o una "E" (y sólo una), seguida OPCIONALMENTE por un signo + o un signo - (y sólo uno), seguido de 1 a 3 dígitos numéricos para expresar Notación Científica.

Tabla T.6.2.A

- Números fraccionarios:

  (\+?|\-?)\d{1,603}\/(\+?|\-?)\d{1,603}

Caracteres	Opcional	Descripción
(\+?|\-?)	Sí	Un signo + o un signo – (y sólo uno).
\d{1,603}	No	De 1 a 603 dígitos numéricos.
\ /	No	Un signo / para separar el Numerador del Denominador.
(\+?|\-?)	Sí	Un signo + o un signo – (y sólo uno).
\d{1,603}	No	De 1 a 603 dígitos numéricos.

Tabla T.6.2.B

- Números romanos:

```
^(?i:(?=[MDCLXVI])((M{0,3})((C[DM])|(D?C{0,3}))?((X[LC])|(L?XX{0,2})
|L)?((I[VX])|(V?(II{0,2}))|V)?))$
```

Caracteres	Opcional	Descripción
^	-	La coincidencia debe comenzar al principio de la cadena.
?i:	-	Se omite la distinción entre mayúsculas y minúsculas.
(?=[MDCLXVI])	-	Lista de caracteres admitidos.
(M{0,3})	No	De 0 a 3 letras M.
((C[DM])|(D?C{0,3}))?	Sí	Una C seguida de una D o una M (o ambas), o bien una D seguida de 0 a 3 C's.
((X[LC])|(L?XX{0,2})|L)?	Sí	Una X seguida de una L o una C (o ambas), o bien una L (opcional) seguida de cero a dos XX, o bien una L.
((I[VX])|(V?(II{0,2}))|V)?	Sí	Una I seguida de una V o una X (o ambas), o bien una V (opcional) seguida de cero a dos II , o bien una V.
$	-	Final de línea.

Tabla T.6.2.C

Si el número se ajusta a alguno de estos patrones, se da por válido y se descompone en parte entera y parte decimal (si no es fraccionario), o bien en numerador y denominador (si es fraccionario). En el caso de los números romanos, se convierte a entero y se trata como tal.

6.3. Concurrencia.

Hemos diseñado un servicio web, y por lo tanto cabe la posibilidad de que la demanda por parte de clientes sea muy alta. Ello exige que el proceso de conversión sea lo más rápido y eficiente posible, y dado que dicho proceso se descompone en tareas que pueden ejecutarse independientemente, resulta

obvio pensar que la ejecución simultánea de dichas tareas puede aportar una reducción considerable en el tiempo de respuesta.

La biblioteca TPL (*Task Parallel Library*, o *Biblioteca de Procesamiento Paralelo basado en tareas*) de *.NET Framework* se basa en el concepto de tarea, que representa una operación asincrónica. En cierta forma, una tarea recuerda a un subproceso o elemento de trabajo, pero en un nivel más alto de abstracción. El término "paralelismo de tareas" hace referencia a la ejecución simultánea de una o varias tareas independientes. Las tareas proporcionan dos ventajas fundamentales:

- Un uso más eficaz y más escalable de los recursos del sistema.

  En segundo plano, las tareas se ponen en cola, con algoritmos que determinan y ajustan el número de subprocesos y que ofrecen el equilibrio de carga para maximizar el rendimiento. Esto hace que las tareas resulten relativamente ligeras y que, por tanto, pueda crearse un gran número de ellas para habilitar un paralelismo pormenorizado.

- Un mayor control mediante programación del que se puede conseguir con un subproceso o un elemento de trabajo.

  Las tareas y el marco que se crea en torno a ellas proporcionan un amplio conjunto de API's que admiten el uso de esperas, cancelaciones, continuaciones, control robusto de excepciones, estado detallado, programación personalizada, etc.

Por estos dos motivos, en .NET Framework, TPL es la API preferida para escribir código multiproceso, asincrónico y paralelo.

Existen dos formas de crear y ejecutar tareas: implícitamente, mediante el método *Parallel.Invoke*, al que hay que pasarle un delegado *Action* por cada elemento de trabajo, o explícitamente, con objetos *Task*. Esta última alternativa permite un mayor control de la ejecución de tareas y también que éstas puedan devolver un valor, motivos por los cuales se ha elegido como método de implementación del paralelismo que necesitamos.

Como hemos comentado a lo largo de este documento, el método *Convertir* es el encargado de ejecutar las tareas de conversión y devolver los resultados obtenidos por éstas. En la siguiente lista se muestran dichas tareas,

independientes unas de otras, lo que nos permite asignar un objeto *Task* del tipo *Res_Par* (Resultados Parciales) a cada una de ellas.

T_Especiales	Tarea para convertir a números especiales
T_Pares_Dig	Tarea para convertir a Pares de dígitos
T_General	Tarea para convertir a Número General (3 dígitos o menos)
T_ECorta	Tarea para convertir a Escala Corta (o a Número General si tiene menos de 10 dígitos)
T_ELarga	Tarea para convertir a Escala Larga (10 dígitos o más)
T_Decimal	Tarea para convertir la Parte Decimal de un Número.
T_Romanos	Tarea para convertir a números romanos.
T_Integ_No_Fracc	Tarea para la integración de resultados de números No Fraccionarios.
T_General_Num	Tarea para convertir a Número General (3 dígitos o menos) el Numerador de un Número Fraccionario.
T_ECorta_Num	Tarea para convertir a Escala Corta (o a Número General si tiene menos de 10 dígitos) el Numerador de un Número Fraccionario.
T_ELarga_Num	Tarea para convertir a Escala Larga (10 dígitos o más) el Numerador de un Número Fraccionario.
T_General_Den	Tarea para convertir a Número General (3 dígitos o menos) el Denominador de un Número Fraccionario.
T_ECorta_Den	Tarea para convertir a Escala Corta (o a Número General si tiene menos de 10 dígitos) el Denominador de un Número Fraccionario.
T_ELarga_Den	Tarea para convertir a Escala Larga (10 dígitos o más) el Denominador de un Número Fraccionario.
T_Integ_Fracc	Tarea para la integración de resultados de números Fraccionarios.

Tabla T.6.3.A

Dependiendo de las características del número, se lanzan unas tareas u otras. El flujo es el siguiente:

- Si se trata de un número no fraccionario:
    - Se lanzan simultáneamente las tareas *T_General*, *T_ECorta* y *T_ELarga*.
    - Si el número es positivo y no tiene parte decimal, se lanzan simultáneamente también las tareas *T_Especiales*, *T_Pares_Dig* y *T_Romanos*. En caso contrario se lanza la tarea *T_Decimal*.
    - Se espera a que finalicen todas las tareas y se integran todos sus resultados mediante la función *Integrar_Resultados*.

- Si se trata de un número fraccionario:

    - Si las longitudes del numerador y el denominador lo permiten, se realiza la división entre ambos y el resultado se trata como no fraccionario.
    - Se lanzan simultáneamente las tareas *T_General_Num, T_ECorta_Num, T_ELarga_Num, T_General_Den, T_ECorta_Den y T_Elarga_Den.* Si el número pudo convertirse a no fraccionario, se lanzan simultáneamente también las tareas correspondientes.
    - Se espera a que finalicen todas las tareas y se integran todos sus resultados mediante la función *Integrar_Resultados*.

El hecho de que se lance una tarea no implica que ésta devuelva resultados. Cada tarea controla internamente diversos aspectos, como los errores que se puedan producir en la conversión o la longitud del número. Por ejemplo, si un número tiene tan solo 6 dígitos, la tarea *T_ELarga* no devuelve resultados.

6.4. Fraccionamiento y concatenación.

Todas las tareas de conversión devuelven sus resultados en instancias de la clase *Res_Par* (Resultados Parciales). Esta clase permite el almacenamiento paralelo de distintos tipos de resultados. Así por ejemplo, la tarea para convertir el número a texto en escala larga genera múltiples resultados: cardinal en inglés británico, ordinal en inglés británico, cardinal en inglés americano, ordinal en inglés americano, e incluso dos variantes dentro de cada tipo.

Pero los resultados no siempre se obtienen de forma directa; de hecho, cuando el número tiene más de tres dígitos se van formando mediante la concatenación de fragmentos, cada uno de los cuales se corresponde con la conversión a texto de un grupo de tres dígitos o menos. Esto ocurre, por ejemplo, en las tareas para convertir a texto en escala corta y en escala larga.

El procedimiento genérico que siguen estas tareas puede resumirse en los siguientes pasos:

1. Se inicializan variables, entre ellas la de almacenamiento de resultados parciales, que será devuelta por la función asociada a la tarea.

2. Se inicializa un *array* con los términos correspondientes a los grupos de tres dígitos (esto se detallará en el próximo apartado).

3. Se descompone sucesivamente la parte entera del número en grupos de tres dígitos, desde los menos significativos hasta los más significativos, o lo que es lo mismo, de derecha a izquierda. Esto implica un bucle con una iteración por cada grupo. Con cada uno se hace lo siguiente:

    a. Si se trata del primer grupo de dígitos, se obtiene la conversión del grupo a Cardinal (si procede) y a Ordinal (también si procede). Estos resultados se guardan en variables temporales.

    b. En caso contrario, es decir, para el resto de grupos, siempre se obtiene la conversión del grupo a Cardinal, ya que los Ordinales sólo influyen en el primero. El resultado se guarda en otra variable temporal.

    c. Si se trata del tercer grupo de dígitos, se guardan los resultados acumulados hasta el momento en dos variables auxiliares (una para Cardinal y otra para Ordinal). Esto es necesario porque a partir del tercer grupo se genera una nueva variante de resultados que coinciden hasta este punto con los actuales.

    d. Si aún no se ha almacenado ningún resultado, se almacenan las conversiones a Cardinal y Ordinal del grupo actual junto con el término correspondiente al grupo (generado en el paso 2), en la variable de almacenamiento, concatenando por la derecha las correspondientes conversiones del primer grupo, que fueron almacenadas en variables temporales.

    e. En caso contrario, se concatenan por la izquierda las conversiones del grupo actual junto con el término correspondiente al grupo, a los resultados obtenidos en el paso anterior.

    f. Si se trata del tercer grupo de dígitos, se almacenan las conversiones a Cardinal y Ordinal del grupo actual junto con la palabra “thousand”, seguida del término correspondiente al

grupo anterior, y se les concatenan por la derecha los resultados que fueron guardados en el paso c. De esta forma tenemos una nueva variante: "thousand" + Término del grupo anterior. A partir de este momento se seguirán concatenando las nuevas conversiones por la izquierda.

4. Una vez terminadas todas las iteraciones, se procede con una ampliación de los resultados actuales. Por cada resultado obtenido:

    a. Si procede hacerlo, se inserta un nuevo resultado sustituyendo el primer "one" por "a". Por ejemplo: "**one** thousand five hundred…" → "**a** thousand five hundred…".

    b. Se inserta un nuevo resultado concatenando las partes del texto con la palabra "and". Esto da lugar al formato británico. Por ejemplo: "one thousand five hundred…" → "one thousand **and** five hundred".

Para aclarar más estos pasos, veamos un ejemplo. Utilicemos uno de los números que vimos en el apartado sobre la clase *Res_Def*:

530,765,255,405

- Paso 1: Inicialización de variables.

- Paso 2: Inicialización del *array* de términos de grupos. En este caso: {"thousand", "million", "billion"}, si se trata de escala corta, o bien {"thousand", "million", "milliard"}, si se trata de escala larga.

- Paso 3: Se descompone el número en grupos de tres dígitos (el grupo más significativo puede tener menos de tres). Cada iteración queda como se muestra a continuación:

  Ejemplo: con la categoría **Cardinal – Escala Corta – AmE**:

Iteración	Grupo	Resultados
1	405	• four hundred five
2	255	• two hundred fifty-five thousand, four hundred five
3	765	• seven hundred sixty-five million, two hundred fifty-five thousand, four hundred five
4	530	• five hundred thirty **billion**, seven hundred sixty-five

		million, two hundred fifty-five thousand, four hundred five • five hundred thirty **thousand million**, seven hundred sixty-five million, two hundred fifty-five thousand, four hundred five

Tabla T.6.4.A

Ejemplo: con la categoría **Ordinal – Escala Larga – BrE**:

Iteración	Grupo	Resultados
1	405	• four hundred fifth
2	255	• two hundred fifty-five thousand, four hundred fifth
3	765	• seven hundred sixty-five million, two hundred fifty-five thousand, four hundred fifth
4	530	• five hundred thirty **milliard**, seven hundred sixty-five million, two hundred fifty-five thousand, four hundred fifth • five hundred thirty **thousand million**, seven hundred sixty-five million, two hundred fifty-five thousand, four hundred fifth

Tabla T.6.4.B

Nota: Aunque se trata de inglés británico, la concatenación con "and" no se hace desde el primer momento, sino en la ampliación de resultados.

- Paso 4: Se amplían los resultados. Hasta este momento las categorías de resultados son:

  - Cardinal – Escala Corta – AmE
  - Ordinal – Escala Corta – AmE
  - Cardinal – Escala Larga – BrE (sin concatenar aún con "and")
  - Ordinal – Escala Larga – BrE (sin concatenar aún con "and")

  Concatenando sus resultados con "and" (Por ej: "five hundred **and** thirty…") se obtienen estas dos nuevas categorías:

  - Cardinal – Escala Corta – BrE
  - Ordinal – Escala Corta – BrE

  Y se corrigen:

  - Cardinal – Escala Larga – BrE
  - Ordinal – Escala Larga – BrE

En total, 6 categorías con 2 resultados cada una, por lo que tenemos un total de 12 resultados. Si el primer o tercer dígito del grupo más significativo hubiese sido 1, se añadirían nuevos resultados sustituyendo el primer "one…" por "a…".

Por ejemplo, para el número **130,765,255,405:**

"**one** hundred and thirty…" → "**a** hundred and thirty…"

El conjunto de resultados definitivo es el mostrado en el apartado sobre la clase *Res_Def*.

6.5. Construcción dinámica de términos con prefijos y sufijos.

Cada grupo de tres dígitos, salvo el menos significativo, lleva asociado un término que define su peso o ponderación dentro del número. Así, el segundo grupo se corresponde con el término "thousand" (mil), el tercero con "million" (millón), el cuarto con "billion" (billón), y así sucesivamente.

Para proporcionar la mayor flexibilidad posible al código y no tener que insertar mediante texto fijo todos estos términos, que son muchos, teniendo en cuenta el número máximo de dígitos permitido, se ha implementado una función que construye y devuelve, en tiempo de ejecución, un *array* con los términos necesarios para definir el número. Únicamente se precisa la siguiente inicialización de cuatro *arrays* de Prefijos:

```
v_Pref_1 = {"m", "b", "tr", "quadr", "quint", "sext", "sept", "oct", "non"};
v_Pref_2 = {"dec", "vi", "tri", "quadra", "quinqua", "sexa", "septua", "octo", "nona"};
v_Pref_3 = {"un", "duo", "tre", "quattuor", "quin", "sex", "septen", "octo", "noven"};
v_Pref_4= {"un", "duo", "tres", "quattuor", "quinqua", "sex", "septen", "octo", "noven"};
```

Esta función se llama *Cargar_Textos_Grupos*, y se utiliza tanto para generar texto en cardinal como en ordinal, e incluso para la parte decimal, añadiendo en este último caso el sufijo "th" o "ths" a cada término. El límite está en el término "centillion" (303 dígitos en Escala Corta). A continuación describimos el algoritmo que sigue dicha función:

- Se definen las siguientes constantes:
    - K_Suf_1 = “illion”
    - K_Suf_2 = “gintillion”
    - K_Ths = “ thousand“
    - K_Cent = “centillion”

- Se redimensiona el *array* con el número de grupos que componen el número (número de dígitos / 3). Cada elemento del *array* contendrá el término asociado a un grupo de tres dígitos.

- Se inicializa sólo el primer elemento del *array* con la constante *K_Ths* (“thousand”), ya que la raíz de este término es distinta al resto.

- El resto de términos se construye concatenando los prefijos y sufijos definidos anteriormente de la siguiente manera:
    - Si es un grupo entre los dígitos 6 y 30 (desde “Million” hasta “Nonillion”):
      Terminos (Ind_G) = “ “ + v_Pref_1(Ind_G – 1) + K_Suf_1

Índice Grupo	Núm. Digitos	Prefijo	Sufijo	Término
1	6	M	illion	Million
2	9	B		Billion
3	12	Tr		Trillion
4	15	Quadr		Quadrillion
…	…	…		…
9	30	Non		Nonillion

Tabla T.6.5.A

    - Si es un grupo entre los dígitos 33 y 60 (desde “Decillion” hasta “Novendecillion”):
        - Si el Resto de dividir Ind_G entre 10 es igual a 0:
          Terminos(Ind_G) = “ “ + v_Pref_2(0) + K_Suf_1
        - Si no:
          Terminos (Ind_G) = “ “ + v_Pref_3(Resto(Ind_G/10) – 1) + v_Pref_2((Ind_G/10) – 1) + K_Suf_1

Índice Grupo	Núm. Digitos	Prefijo	Sufijo	Término
10	33	Dec	illion	Decillion
11	36	Un + Dec		Undecillion
12	39	Duo + Dec		Duodecillion

13	42	Tre + Dec		Tredecillion
…	…	…		…
19	60	Noven + Dec		Novendecillion

Tabla T.6.5.B

- Si es un grupo entre los dígitos 63 y 300:
  - Si el Resto de dividir Ind_G entre 10 es igual a 0:
    Terminos (Ind_G) = " " + v_Pref_2((Ind_G/10) – 1) + K_Suf_2
  - Si no:
    Terminos (Ind_G) = " " + v_Pref_4(Resto(Ind_G/10) – 1) + v_Pref_2((Ind_G/10) – 1) + K_Suf_2

Índice Grupo	**Núm. Dígitos**	**Prefijo**	**Sufijo**	**Término**
20	63	Vi	gintillion	Vigintillion
21	66	Un + Vi		Unvigintillion
22	69	Duo + Vi		Duovigintillion
23	72	Tre + Vi		Trevigintillion
…	…	…		…
99	300	Noven + Nona		Novennonagintillion

Tabla T.6.5.C

- Si el grupo se corresponde con el dígito 303 ("Centillion"):
  Terminos (Ind_G) = " " + K_Cent. → Centillion

NOTAS:

- *Terminos* es el nombre del *array* que se devuelve.
- *Ind_G* es el índice de un elemento en el array, comenzando desde 0.

Estos son los términos empleados en **Escala Corta**. Para la **Escala Larga**, se utiliza el mismo *array*, pero entre medio de cada par de términos se introduce uno igual al primero de ellos cambiando el sufijo "-illion" por "-illiard":

Thousand (3)	Million (6)	**Milliard** (9)	Billion (12)	**Billiard** (15)
Trillion (18)	**Trilliard** (21)	Quadrillion (24)	**Quadrilliard** (27)	…

Tabla T.6.5.D

6.6. Integración de resultados.

Como vimos en el apartado sobre concurrencia, la obtención independiente de resultados parciales por parte de cada tarea nos lleva inexorablemente a una integración de resultados que dará lugar al conjunto definitivo.

La función *Integrar_Resultados* se encarga de integrar y combinar los resultados parciales obtenidos y devolverlos en un único conjunto, a través de una instancia de la clase *Res_Def* (Resultados Definitivos).

El algoritmo que implementa esta función es el siguiente:

- Se inicializan variables, entre ellas la de almacenamiento de los resultados definitivos, que será devuelta por la función.
- Si el número es **no fraccionario**:
    - Para cada resultado de la **parte entera**:
        - Si hay errores, se incluyen en la lista de errores definitiva y se pasa al siguiente resultado.
        - Se obtiene el resultado y la categoría a la que pertenece.
        - Si el número es negativo, se antepone la palabra “negative” si la categoría corresponde a inglés americano, o “minus” si corresponde a inglés británico.
        - Para cada resultado de la **parte decimal** (si la hay):
            - Si hay errores, se incluyen en la lista de errores definitiva y se pasa al siguiente resultado.
            - Se obtiene el resultado y la categoría a la que pertenece.
            - Si la categoría de la parte decimal es “con punto”, se concatena la parte entera con la parte decimal y se inserta como resultado definitivo. Si la categoría de la parte decimal es “sin punto” y la

categoría de la parte entera no es de inglés británico (que ya utiliza “and” y por tanto no se puede concatenar), también se concatena e inserta como resultado definitivo.

- Si no hay **parte decimal**, simplemente se inserta el resultado como definitivo.

- Si el número es **fraccionario**:
  - Para cada resultado del **numerador**:
    - Si hay errores, se incluyen en la lista de errores definitiva y se pasa al siguiente resultado.
    - Se obtiene el resultado y la categoría a la que pertenece.
    - Si el número es negativo, se antepone la palabra “negative” si la categoría corresponde a inglés americano, o “minus” si corresponde a inglés británico.
    - Para cada resultado del **denominador**:
      - Si hay errores, se incluyen en la lista de errores definitiva y se pasa al siguiente resultado.
      - Se obtiene el resultado y la categoría a la que pertenece.
      - Si la combinación Numerador - Denominador no es válida, por tener formatos incompatibles, se descarta y se pasa al siguiente resultado.
      - Si el denominador es 2 o 4, se añaden, excepcionalmente, nuevos resultados alternativos empleando los términos “half” o “quarter” respectivamente.

- Si el denominador es mayor que 1, se concatena una “s” por la derecha al resultado.

- Si la categoría es de inglés británico, se concatena un “and” o “and a” entre numerador y denominador. Si no, simplemente se concatena un espacio entre ambos.

- Finalmente se devuelven los resultados definitivos. En el caso de que existan resultados fraccionales y no fraccionales, se devuelven ambos.

6.7. Mejora de rendimiento a través de la clase *StringBuilder*.

Los objetos de tipo *String* son invariables. Esto quiere decir que cada vez que se usa uno de sus métodos, se crea un nuevo objeto de cadena en la memoria, que necesita una nueva asignación de espacio para dicho objeto.

Esta circunstancia hace que en los casos en que sea necesario hacer repetidas modificaciones en una cadena, la sobrecarga asociada a la creación de un nuevo objeto *String* pueda disminuir el rendimiento.

Por su parte, la clase *StringBuilder* permite modificar una cadena sin crear un nuevo objeto, y por ello puede mejorar considerablemente el rendimiento al concatenar muchas cadenas en un bucle.

Es por ello que en nuestro caso, donde precisamente necesitamos realizar continuas concatenaciones, se hace altamente recomendable utilizar objetos *StringBuilder* para obtener un rendimiento óptimo.

Uno de los puntos críticos del código en este sentido es en el método *Editar* de la clase *Res_Par*, donde se producen las concatenaciones de los resultados parciales, y por ello empleamos una lista de tipo *StringBuilder*.

7. CONCLUSIONES Y TRABAJOS FUTUROS.

En vista de todo lo expuesto, se pueden obtener las siguientes conclusiones:

- La escritura de números con palabras en el idioma inglés es un proceso sistemático, aunque con determinadas excepciones.

- En la actualidad existen dos corrientes claramente diferenciadas: el inglés británico y el inglés americano, cada una de ellas con sus peculiaridades.

- Dichas peculiaridades, como por ejemplo el uso de escala corta o escala larga, dan lugar a que un mismo número pueda ser escrito de distintas formas.

- Aunque los términos que se emplean para la designación de números constituyen un conjunto finito, éstos permiten referirse hasta números extremadamente grandes, prácticamente inmanejables para el ser humano.

Si como es de esperar, esta utilidad tiene una aceptación similar a la ya desarrollada para la conversión a texto en español, sería más que razonable considerar la idea de extenderla a otros idiomas, especialmente a los más hablados en el mundo, como francés, alemán, etc.

8. FUENTES DE INFORMACIÓN.

La información recopilada en este documento ha sido obtenida a partir de las siguientes fuentes:

- Numbers in academic writing: https://www.une.edu.au/__data/assets/pdf_file/0012/10803/WC_Numbers-in-academic-writing.pdf

- House Style (Oxford University Press): http://global.oup.com/uk/academic/authors/AuthorGuidelinesMain/HouseStyle/#lev19

- Numbers and dates (Oxford Dictionaries): http://www.oxforddictionaries.com/secondary/harts_rules/11-1-numbers-general-principles

- English numerals (Wikipedia): http://en.wikipedia.org/wiki/English_numerals

- Generador de números ordinales y cardinales en ingles (hasta 66 dígitos): http://www.ego4u.com/en/cram-up/vocabulary/numbers/generator

- Numbers Vocab (EnglishClub): https://www.englishclub.com/vocabulary/numbers.htm

- Names of large numbers: http://en.wikipedia.org/wiki/Names_of_large_numbers

- English names of the first 10000 powers of 10: http://www.isthe.com/chongo/tech/math/number/tenpowere.html

- Ordinal Numbers: http://www.ego4u.com/en/cram-up/vocabulary/numbers/ordinal

- Name of Decimals: http://www.aaamath.com/nam.htm#topic8

- Read and Write Decimals: http://www.mathgoodies.com/lessons/decimals/read_write.html

- Introduction to Decimals: http://cstl.syr.edu/fipse/Decunit/intrdec/intrdec.htm

- Lenguaje de expresiones regulares: https://msdn.microsoft.com/es-es/library/az24scfc%28v=vs.110%29.aspx

- Paralelismo de tareas: https://msdn.microsoft.com/es-es/library/dd537609%28v=vs.110%29.aspx

- Utilizar la clase *StringBuilder* en *NET.Framework*: https://msdn.microsoft.com/es-es/library/2839d5h5%28v=vs.110%29.aspx