UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

CONVERSOR DE NÚMEROS A TEXTO EN FRANCÉS

Grado en Ingeniería Informática

Tutorizado por:
Tutor oficial: Francisco Javier Carreras Riudavets
Cotutora oficial: Florence Yolande Gerard Lojacono

Carlos Martel Lamas – 44747470G

Las Palmas de Gran Canaria a 1 de junio de 2018

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D/Dª **Carlos Martel Lamas**, autor del Trabajo de Fin de Título **Conversor de números a texto en francés**, correspondiente a la titulación **Grado de Ingeniería Informática**,
en colaboración con la empresa/proyecto (indicar en su caso) _____

S O L I C I T A

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

[ ] Se ha iniciado o hay intención de iniciarlo (defensa no pública).
[x] No está previsto.

Y para que así conste firma la presente.

Las Palmas de Gran Canaria, a 1 de junio de 2018

El estudiante

MARTEL LAMAS CARLOS - 44747470G
Firmado digitalmente por MARTEL LAMAS CARLOS - 44747470G
Fecha: 2018.06.01 10:30:57 +01'00'

Fdo.: Carlos Martel Lamas 44747470G

A rellenar y firmar **obligatoriamente** por el/los tutor/es

En relación a la presente solicitud, se informa:

[x] Positivamente

[ ] Negativamente
(la justificación en caso de informe negativo deberá incluirse en el TFT05)

CARRERAS RIUDAVETS FRANCISCO JAVIER - 42826682S
2018.06.01 11:14:23 +01'00'

Fdo.: _____

DIRECTOR DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

1 Contenido

2 Estado actual y objetivos

Este trabajo consiste en el desarrollo de un servicio web para convertir un número escrito en cifras (1, 2, 3, …) en su equivalente texto en francés (“un”, “deux”, “trois”, …). Hoy en día estamos más que lanzados a escribir números en su forma de dígito (escribimos normalmente 2018, no “Dos mil dieciocho”), sobre todo en el caso de que éstos sean grandes. El problema está en que en algunos casos es necesario expresarnos por palabras y no por dígitos. Esto se da, por ejemplo, en el caso de nosotros querer expresar un número muy grande (véase el número 93546184615834345) en palabras en vez de en una hoja de papel donde su forma numérica nos valdría.

Me gustaría comentar antes de comenzar los entresijos de lo que se lleva a cabo en estos casos, algunos puntos a tener en cuenta a la hora de formar números mediante palabras en francés.

2.1 Conversión de números a texto en francés

Al igual que en el español, el inglés y un gran abanico de idiomas, la conversión de un número a su análogo en texto resulta sistemático a la par que, a veces, incluso mecánico. También hay que tener en cuenta que, aplicando algunas excepciones podemos obtener diferentes variantes igualmente válidas.

Al igual que en el castellano, un número en francés puede ser escrito de diversas formas: cardinal, ordinal, fraccionario, … En algunos casos mucho más cotidianos de lo que nos podría parecer en un principio, estos números pueden ser escritos de una manera muy diferente a lo que en un principio se haría.

Las maneras más comunes de escribir números en francés es la siguiente:

2.1.1 Números cardinales

Los números cardinales expresan cantidades, algunos de estos quedan representados en la siguiente tabla:

0	Zèro	5	Cinq	10	Dix	15	Quinze
1	Un	6	Six	11	Onze	16	Seize
2	Deux	7	Sept	12	Douze	17	Dix-sept
3	Trois	8	Huit	13	Treize	18	Dix-huit
4	Quatre	9	Neuf	14	Quatorze	19	Dix-neuf

TABLA 1- NÚMEROS CARDINALES 0-19

Si nos fijamos bien, a partir del número 17 en adelante el número en sí se forma con el nombre de la decena (en este caso, diez => “dix”), un guion y el nombre de la unidad.

A partir de la primera decena, las siguientes son:

20	Vingt	60	Soixante
30	Trente	70	Soixante-dix
40	quarante	80	Quatre-vingts
50	Cinquante	90	Quatre-vingt-dix

TABLA 2 - DECENAS DE NÚMEROS CARDINALES

Cabe notar que el 70 se puede identificar como 60 + 10 (sesenta + diez = setenta) y que el 80 y el 90 también tienen una forma relativamente parecida (80 = 4*20; 90 = 4*20+10).

Los números cardinales que van desde el 20 hasta el 99 tienen un buen abanico de peculiaridades:

- Los números terminados en 1 (21, 31, 41, …) hasta el 61 se escriben de la forma [*decena*] + “et un”.

21	Vingt-et-un	41	Quarante-et-un
31	Trente-et-un	51	Cinquante-et-un

TABLA 3 - DECENAS TERMINADAS EN 1 CARDINAL

- En la séptima y novena decena los números se tratan de tal forma que es como si estuviéramos contando a partir de diez mediante la forma [*decena*] + (forma cardinal [*unidad+10*]) mientras que las demás los números se tratan de la forma [*decena*] + forma cardinal[*unidad*].

72	Soixante-et-onze	84	Quatre-vingt-quatre
76	Soixante-et-seize	93	Quatre-vingt-treize
81	Quatre-vingt-un	98	Quatre-vingt-dix-huit

TABLA 4 - ÚLTIMAS DECENAS CARDINALES

- Para los números compuestos por dos cifras, la unión se hace mediante un guion (“-”) o bien mediante la partícula “et” en el caso de que termine en 1 en 21, 31, 41, 51 y 61. En el caso del 81 sólo se usa el guion (“-”).

Las centenas se tratan de manera regular, teniendo en cuenta que **no** es invariable, sino que hay una forma singular (“cent”) y plural cuando hay más de una centena (“cents”). En el caso de que el dígito de la centena sea diferente de uno (1) se usará entonces la forma del plural.

Cabe destacar que si no se trata de la centena sola (100, 200, 300) sino de un número compuesto por centena-decena-unidad, la centena se trata en singular, aunque su dígito sea diferente de uno (1).

100	Cent	164	Cent-soixante-quatre
200	Deux-cents	584	Cinq-cent-quatre-vingt-quatre
…	…	…	…
900	Neuf-cents	999	Neuf-cent-quatre-vingt-dix-neuf

TABLA 5 - CENTENAS CARDINAL

Por otra parte, los millares son regulares. No tienen forma plural. Simplemente Se acompaña el dígito con la partícula “mille” para simbolizar que estamos hablando de millares.

1000	Mille	1068	Mille-soixante-huit
2000	Deux-mille	5824	Cinq-mille-huit-cent-vingt-quatre
…	…	…	…
9000	Neuf-mille	9999	Neuf-mille-neuf-cent-quatre-vingt-dix-neuf

TABLA 6 - MILLARES CARDINAL

Si pasamos a hablar de cifras algo mayores (como es en el caso de los millones), estos tienen tanto forma singular cuando su dígito relacionado es el uno (1) como una forma plural cuando es mayor a este. Se les aplica a estos números las mismas reglas que las centenas.

1000000	Un million	1000654	Un million six cent cinquante-quatre
2000000	Deux millions	2258963	Deux millions deux cent cinquante-huit mille neuf cent soixante-trois
…	…	…	…
9000000	Neuf millions	9999999	Neuf millions neuf cent quatre-vingt-dix-neuf mille neuf cent quatre-vingt-dix-neuf

TABLA 7 - MILLONES CARDINAL

A partir de aquí es donde hay que hacer un inciso, ya que existen dos maneras en las que se puede escribir el número. A estas formas se les denomina “escalas” y son la escala grande y la escala pequeña.

En el francés la escala oficial es la grande, aunque es un tema de debate, ya que ambas se usan prácticamente de la misma forma y son populares en casi la misma medida.

Gracias a la influencia inglesa, esta escala pequeña está cobrando cada día algo más de importancia.

Sin más, aquí va una breve explicación de ambas:

- **Escala grande (oficial en el francés):** es una escala donde mil millones (1 000 000 000) se nombra como "milliard", y a un millón de millones (1 000 000 000 000) se llama "billion". Como curiosidad, siempre se da esta alternancia entre la terminación "-ion" e "-iard". Los términos que acaban en "-ion" siempre son escalas múltiplos de 6 (10^6, 10^{12}, …) y los que acaban en "-iard" son aquellos cogidos escalas múltiplos de 3 (10^9, 10^{15}, …)
- **Escala pequeña (cada vez gozando de mayor popularidad):** es una escala donde mil millones (1 000 000 000) se nombra como "billion", y a un millón de millones (1 000 000 000 000) se llama "trillion".

La siguiente tabla ilustra de una manera mucho más directa estas diferencias:

	Escala pequeña	**Escala grande**
1 000 000	Un million	Un million
1 000 000 000	Un billion	Un milliard
1 000 000 000 000	Un trillion	Un billion
1 000 000 000 000 000	Un quadrillion	Un billiard
1 000 000 000 000 000 000	Un quintillion	Un trillion
1 000 000 000 000 000 000 000	Un sextillion	Un trilliard
…	…	…

TABLA 8 - NÚMEROS SUPERIORES A UN MILLÓN CARDINAL

Como nota final, desde 1990 todos los números han sustituido los espacios que lo componen por guiones "-".

2.1.2 Números ordinales

Los números ordinales son esenciales a la hora de listar cualquier tipo de secuencia de objetos y poder designar posiciones dentro de una serie. La forma en la que se escriben estos números es muy parecida a su forma cardinal, aunque a veces hay que apreciar unas grandes diferencias.

0	Zèroième	5	Cinquième	10	Dixième	15	Quinzième
1	Premier (m) – Première (f) - Unième	6	Sixième	11	Onzième	16	Seizième
2	Deuxième – Second (m) – Seconde (f)	7	Septième	12	Douzième	17	Dix-septième

3	Troisième	8	Huitième	13	Treizième	18	Dix-huitième
4	Quatrième	9	Neuvième	14	Quatorzième	19	Dix-neuvième

TABLA 9 - 19 PRIMEROS NÚMEROS ORDINALES

Algo curioso de los números ordinales en francés es que, a diferencia del español, en el francés sí que existe una denominación concreta para el ordinal cero (0).

Estos números ordinales, si nos fijamos bien, se han formado acoplando la partícula "-ième" al final del número cardinal aplicando previamente una serie de reglas.

- Aquellos números cuyo cardinal acaba en "-e" se elimina y se sustituye por "-ième".
- Hay dos números en concreto (y sus derivados) que necesitan de reglas especiales:
  1. **Cinq:** se le añade una "u" al final del dígito para luego acoplarle sufijo "-ième" (cinq >> cinquième). Esto se aplica a todos los números que terminen en cinco (5).
  2. **Neuf:** en este caso se sustituye la "f" final por una "v" antes de añadir la partícula ordinal. Esto se aplica a todos los números que acaban en nueve (9).
- En el caso del uno (1) éste tiene tanto masculino ("premier") como femenino ("première") y solo se utiliza para designar al primero de cualquier lista u orden. En cualquier otro caso (21, 31, 41, …) se utiliza la variante "unième".
- El segundo elemento también tiene una apreciación especial, ya que se usa la palabra "second" si y solo si se trata de una lista de dos elementos. Como en el caso del número uno (1), también tiene masculino ("second") y femenino ("seconde").

Con todo esto, podemos asumir que un número ordinal es la suma de su equivalente cardinal y una partícula después de aplicar una serie de reglas.

21	Vingt et unième	32	Trente-deuxième
81	Quatre-vingt-unième	95	Quatre-vingt-quinzième
101	Cent unième	852	huit cent cinquante-deuxième
1003	Mille troisième	2500	Deux mille cinq centsième

TABLA 10 - EJEMPLOS NÚMEROS ORDINALES

2.1.3 Números fraccionarios

Un número fraccional se compone de dos partes. Un numerador (normalmente en forma cardinal) y un denominador (normalmente en forma ordinal). Por lo tanto, si nos fijamos solo en el denominador, un número ordinal no solo representa la posición dentro de un orden

establecido, sino también la división de la unidad en esas partes iguales. En estos casos podemos pluralizar, por así decirlo, el denominador añadiendo una "-s" al final.

En el caso del francés algunos denominadores reciben nombres especiales, estos bien son pocos, pero hay que tenerlos en cuenta siempre.

- 1/2: no se usa el ordinal "deuxièmes" sino "demi" en el caso masculino o "demie" en el caso femenino.
- 1/3: no se usa el ordinal "troisième" sino la palabra "tiers" tanto en el masculino como en el femenino. Es invariante.
- 1/4: por último, en este casi tampoco se usa su ordinal "quatrième" sino la forma "quart" (con su correspondiente pluralización).
- Los demás números mayores a cuatro (4) son usados mediante su forma ordinal con su correspondiente pluralización.

El numerador, como se comentó antes, se trata siempre de un número en su forma cardinal.

1/2	Un demi (m) – une demie (f)
1/10	Un dixième
1/16	Un seizième
1/8	Un huitième
5/6	Cinq sixièmes
3/8	Trois huitièmes
4/10	Quatre dixièmes
15/16	Quinze seizièmes
2/3	Deux tiers
1/4	Un quart
5/4	Cinq quarts
2/10	Deux dixièmes
45/85	Quarante-cinq quatre-vingt-cinquièmes
23/45	Vingt-trois quarante-cinquièmes
6/100	Six centièmes

TABLA 11 - NÚMEROS FRACCIONARIOS

Por norma general no se suelen expresar de manera oral fracciones demasiado grandes debido a su dificultad a la hora de separar el numerador y el denominador. Pero, en el caso de requerirse, la norma explicada anteriormente se puede extender a cualquier número.

2.1.4 Números decimales

Los números decimales son aquellos resultados de llevar a cabo una división entre dos números (una fracción). Estos números tienen dos (2) partes: una parte entera y una parte decimal. Estos conjuntos están separados normalmente por un punto (.). En el caso de que la parte entera del número sea menor que uno (1), se le añade un cero (0) a la izquierda de este (se escribiría como “0.5” en vez de “.5”).

La forma en la que se pronuncia un número decimal es bastante sencilla, dado que se forma mediante la unión de la forma cardinal mediante la partícula “virgule” como si de en el castellano habláramos de la coma (,) o el punto “.”.

De manera opcional se recomienda también añadir al final del número el orden del decimal (décimas, centésimas, …) para una mayor claridad a la hora de mostrar los resultados.

Posición de los dígitos						
Centaines	Dizaines	Unités	virgule	dixièmes	centièmes	millièmes
3	2	1	0	1	2	3

TABLA 12 DECIMALES

Algunos ejemplos pueden resultar de gran ayuda llegados a este punto:

1. 5.3: cinq virgule trois **dixièmes (1 decimal)**
2. 64.21: soixante-quatre virgule vingt et un **centièmes (2 decimales)**
3. 3 214.65: trois mille deux cent quatorze virgule soixante-cinq **centièmes (2 decimales)**
4. 98 785.321 5: quatre-vingt-dix-huit mille sept cent quatre-vingt-cinq virgule trois mille deux cent quinze **dix-millièmes (4 decimales)**
5. 654 987.321 654: six cent cinquante-quatre mille neuf cent quatre-vingt-sept virgule trois cent vingt et un mille six cent cinquante-quatre **millionièmes (6 decimales)**
6. 3 216 549.321 654 987: trois millions deux cent seize mille cinq cent quarante-neuf virgule trois cent vingt et un millions six cent cinquante-quatre mille neuf cent quatre-vingt-sept **billionièmes (9 decimales)**

En cualquier caso, en el momento en el que la parte entera del número sea cero (0), la podemos omitir y dejar solo la parte decimal o bien omitiendo la parte correspondiente al orden del decimal:

1. 0.2:
    a. Deux **dixièmes**
    b. Zèro virgule deux
2. 0.56:
    a. Cinquante-six **centièmes**
    b. Zèro virgule cinquante-six
3. 0.235:
    a. Deux cent trente-cinq **millièmes**
    b. Zèro virgule deux cent trente-cinq
4. 0.123 456 789:
    a. cent vingt-trois millions quatre cent cinquante-six mille sept cent quatre-vingt-neuf **billionièmes**
    b. zèro virgule cent vingt-trois millions quatre cent cinquante-six mille sept cent quatre-vingt-neuf

2.1.5 Números negativos

Los números negativos se usan a la hora de expresar cantidades menores que cero (0).

El nombre de los números negativos es bastante simple, ya que basta con agregar la partícula “moins” al principio del dígito (como si de un prefijo se tratara) y ya lo habríamos convertido a un número negativo.

2.1.6 Nombres de los números realmente largos

A partir de ciertas cifras los números adquieren, de una manera u otra, un tamaño considerable. Para poder nombrarlos se ha tenido que formar una serie de reglas a la hora de nombrar estas magnitudes. No son de uso común, aunque, debido a los estudios de la ciencia en el ámbito cósmico o con precisiones cada vez más y más exactas, se ha visto necesaria una recopilación de algunos de los números largos que se han encontrado en algunos diccionarios de francés y enciclopedias.

En cuanto a las cantidades no mostradas, en algunos estudios se les trata como “zillions” de manera informal a la hora de hablar de cantidades inusualmente grandes sin especificar su valor.

Número	Escala pequeña	Escala grande
10^3	Mille	Mille
10^6	Million	Million
10^9	Billion	Milliard
10^{12}	Trillion	Billion
10^{15}	Quadrillion	Billiard
10^{18}	Quintillion	Trillion
10^{21}	Sextillion	Trilliard
10^{24}	Septillion	Quadrillion
10^{27}	Octillion	Quadrilliard
10^{30}	Nonillion	Quintillion
10^{33}	Decillion	Quintilliard
10^{36}	Undecillion	Sextillion
10^{39}	Duodecillion	Sextilliard
10^{42}	Tredecillion	Septillion
10^{45}	Quattuordecillion	Septilliard

10^{48}	Quindecillion	Octillion
10^{51}	Sexdecillion	Octilliard
10^{54}	Septendecillion	Nonillion
10^{57}	Octodecillion	Nonilliard
10^{60}	Novemdecillion	Décillion
10^{63}	Vigintillion	Décilliard
10^{66}	Unvigintillion	Undécillion
10^{69}	Duovigintillion	Undécilliard
10^{72}	Trevigintillion	Duodécillion
10^{75}	quattuorvigintillion	Duodécilliard
10^{78}	Quinvigintillion	Trédécillion
10^{81}	Sexvigintillion	Trédécilliard
10^{84}	Septenvigintillion	Quattuordécillion
10^{87}	Octovigintillion	Quattourdécilliard
10^{90}	Novemvigintillion	Quindécillion
10^{93}	Trigintillion	Quindécilliard

10^{96}	Untrigintillion	Sexdécillion
10^{99}	Duotrigintillion	Sexdécilliard
10^{102}	Tretrigintillion	Septendécillion
10^{105}	Quattuortrigintillion	Septendécilliard
10^{108}	Quintrigintillion	Octodécillion
10^{111}	Sextrigintillion	Octodécilliard
10^{114}	Septentrigintillion	Nonidécillion
10^{117}	Octotrigintillion	Nonidécilliard
10^{120}	Novemtrigintillion	Vigintillion
…	…	…
10^{303} (peq), 10^{600}(grande)	Centillion	Centillion

TABLA 13 - NOMBRES DE LOS GRANDES NÚMEROS

A partir de aquí la formación de los números se basa en añadir prefijos (“bi”, “tre”, …) a la raíz. El número más grande que se ha podido encontrar documentado es el “Centillion”.

Como se puede observar, cada número por encima del millón tiene una forma en la escala pequeña y otra en la escala grande. En la escala grande los nombres se diferencian en factores de un millón (cambia el nombre cada 10^6) y en la escala pequeña los números se diferencian en factores de un millar (cambia el nombre cada 10^3)

2.2 Cuando escribimos números mediante dígitos o su equivalente en palabras.

Normalmente nunca nos hacemos una pregunta como esta, pero a veces es necesario tener que saber discernir entre los momentos en los que es necesario escribir un número sólo a través de su representación numérica (a través de sus dígitos: 1, 2, 3, …) o bien comunicarnos mediante su forma en palabras (uno, dos, tres, …).

Por estas razones me parece interesante poder recopilar una serie de normas por las que nos regimos a la hora de escribir un número bien en dígito o bien en su forma en palabras:

1. Escribir con palabras los números que comienzan una oración parece ser algo que está convenido. En el caso de no querer hacerlo, es normal también intentar darle otra forma a la frase para que el número no ocupe la primera posición.
    a. **Veintiséis (26)** personas esperaron en la sala de espera.
    b. **Vingt-six (26)** personnes patientaient dans la salle d'attente

2. Se escribe con palabras cualquier número usado como sustantivo en una oración.
    a. Con su único **cinco (5) de picas**, acababa de ganar la apuesta.
    b. Avec son seul **cinq (5) de pique**, il venait de rafler la mise.

3. Se escribe con palabras de la misma manera cualquier período histórico abreviado.
    a. Era un fanático de la música de los **años setenta (70)**.
    b. Il était fan de la musique des **années soixante-dix (70)**.

4. Cualquier número de cero (0) a veinte (20) será escrito en forma de palabras siempre y cuando el propósito del texto no sea la visualización de datos numéricos como tablas científicas y datos estadísticos. Esta regla también se aplica a números redondos como cincuenta (50), cien (100), mil (1000), …
    a. Invitó a **trece (13)** de sus amigos a su fiesta de cumpleaños.
    b. Il a invité **treize (13)** de ses amis à sa soirée d'anniversaire.

5. En cualquier caso, en la misma oración dos números tendrán que estar escritos en la misma forma para preservar la uniformidad del texto.
    a. A pesar de los **85** participantes, faltaban **15** personas para alcanzar la cuota.
    b. Malgré les **85** participants, il manquait **15** personnes pour atteindre le quota.
6. Se escribirá con palabras un número que exprese una duración. Si se habla en cuanto a distancias se escribe también en palabras el número si la unidad de medida no está abreviada o si no se expresa como un símbolo.
    a. El examen durará **una (1)** hora.
    b. L'examen durera **une (1)** heure.
    c. Con **30 ° C**, a una velocidad de **18 km / h**, realizó un recorrido de **25 km** para entregar su pedido de **3 kg** antes de las **16 h**.
    d. Par **30 °C**, à une vitesse de **18 km/h**, elle a pédalé **25 km** pour lui livrer sa commande de **3 kg** avant **16 h**.
7. En el caso de órdenes mayores al millón (millones, mil millones, …) se escribirán en palabras para evitar dificultades en su lectura.
    a. El país tiene casi **diez millones** de habitantes. Allí viven **1,8 millones** de hablantes de francés.
    b. Le pays compte près **dix millions** d'habitants. Y vivent **1,8 million** de francophones.
8. Los números decimales se escriben con cifras.
9. Los símbolos sólo se usan en el caso de tener un número escrito en cifras.
10. En el caso de que el número haga referencia a un número de serie, siempre se escribirá en su forma con cifras sin espacios ni comas.

    a. Según el **artículo 15** en la **página 353** de la **circular 4317** publicada el **2 de marzo de 2014**.
    b. Selon **l'article 15** en **page 353** de la **circulaire 4317** publiée le **2 mars 2014.**

11. Escribir con palabras números dentro de nombres propios.

2.3 Objetivos a cubrir en este proyecto

Hemos podido comprobar que si nos queremos mover bien dentro de cualquier idioma es necesario poder escribir **correctamente** números con palabras. Como es normal, cuando escribimos números pequeños (hasta, digamos, los millones) parece ser una tarea bastante sencilla que no requiere de ninguna complicación al respecto. El mayor problema viene cuando tenemos que escribir números que nos cuesta, incluso, tener que imaginárnoslos por la simple razón de que no son nada habituales entre las personas. Si, por encima de esto,

tenemos en cuenta que solo algunos números (todos menores que cien) son los que suelen ir escritos de forma escrita (o con palabras) es aún más complicado.

Por estas razones (y, por qué no, por simple y mera curiosidad) no estaría mal disponer de una herramienta que nos brindara las posibilidades de llevar a cabo estas operaciones con los números y nos diera un resultado válido con el que poder enriquecer nuestras cabezas.

Aparte de lo comentado anteriormente, se ha observado que un número puede ser escrito de diferentes formas: escala pequeña o grande, masculino o femenino, … Por norma general nos hemos acostumbrado a escribir los números de una sola forma, pero ¿Por qué no aprender a escribirlos de otra manera alternativa que nos pueda parecer mejor a nuestro parecer?

Con este servicio web se pretende lograr los siguientes objetivos:

- Dar a conocer las diferentes opciones a la hora de escribir un número con palabras.
- Gracias a las operaciones que se hacen, poder enseñar cómo se forman las palabras que forman un número independientemente de su tamaño.
- Servir de apoyo a otras herramientas informáticas que puedan hacer eso de este servicio.

3 Justificación de las competencias específicas cubiertas.

En la elaboración de este proyecto se han cubierto las siguientes áreas que se detallan a continuación:

3.1 Como futuro graduado en Ingeniería Informática

- **O2.** Consciente de la necesidad de actualizar su información permanentemente y dotado de habilidades de autoaprendizaje necesarias.

  El lenguaje en el que está escrito este servicio no es uno que se imparte en los estudios del grado y está, gracias a la plataforma en la que se programa, en constante evolución. Por lo tanto, creo conveniente citar este punto ya que se ha requerido de esa capacidad de autoaprendizaje que bien el lenguaje es parecido a otros, fue necesario aprender el funcionamiento y comprender los entresijos de este.

3.2 Común 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 la forma usada para llevar a cabo el proceso de conversión ha sido vital para obtener una aplicación eficiente y sostenible siguiendo las buenas prácticas de la programación. 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 superior a meramente nacional, ha sido necesaria la implementación de parte del código utilizando técnicas de programación paralela como el uso de los hilos (threading), que sin lugar a dudas otorgan al servicio un tiempo de respuesta menor al que tendría con una programación tradicional que se basa en las implementaciones secuenciales.

3.3 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.

3.4 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.
  El trabajo que se presenta ha sido realizado íntegramente por el autor de este bajo la supervisión de los tutores a cargo. Durante la creación del trabajo se han aplicado conceptos y algunas metodologías aprendidas y estudiadas a lo largo de la carrera.

4 Aportaciones

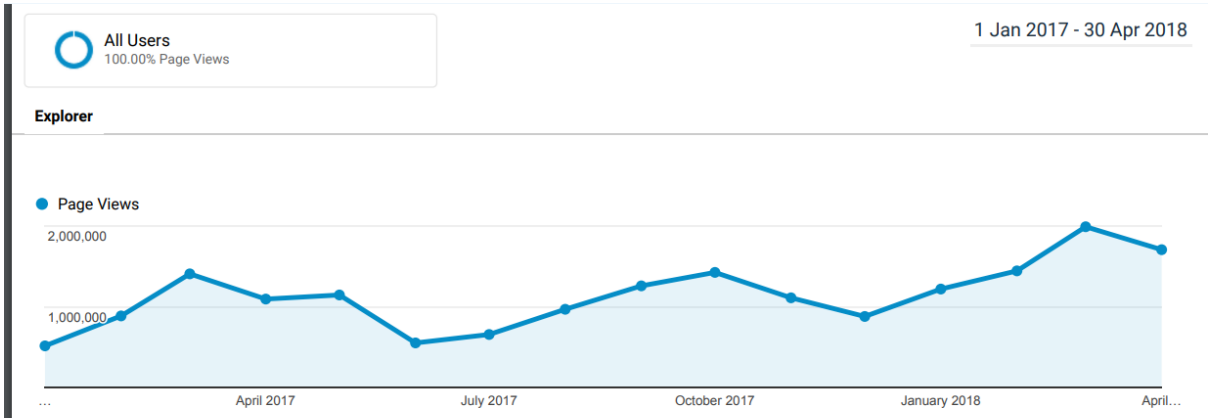
Con la realización de este trabajo se cree importante recalcar las aportaciones en los siguientes campos:

- **Educativo:** se puede decir que es en este campo donde se ha enfocado el objetivo principal del trabajo. Este servicio que se pone al alcance del usuario no tiene por qué servir solamente para personas que quieran aprender el idioma francés sino para los mismos francoparlantes que, por curiosidad, quieran saber cómo se pronuncian o escriben algunas cifras dado que les ahorra el trabajo de pensar en las reglas a aplicar a números de una envergadura considerable. Es más, los conceptos que se manejan en cifras extremadamente grandes son prácticamente desconocidos para cualquier persona que no pertenezca al campo de las matemáticas, dicho lo cual, podemos deducir la gran utilidad de la herramienta.
- **Económico:** es una herramienta bastante útil a la hora de tener que rellenar los campos de un cheque bancario ya que éstos necesitan de la escritura del número no solo en su forma de dígitos sino escrita con palabras.
- **Social:** la versatilidad de esta herramienta radica en la posibilidad de poder juntar este servicio con, por ejemplo, alguna aplicación de *text to speech* para dotar a una página o aplicación de un punto más dentro de la accesibilidad al sitio. Es una característica que, aunque no parezca muy útil, cada vez se ve más requerida (o más bien, valorada) de cara a las personas con discapacidades, sobre todo, visuales.

- **Justicia/Administración:** en el campo de la justicia o bien en la administración (de cualquier tipo) se suele acompañar una cifra de su equivalencia en palabras escritas debido a que se busca llevar a los menos fallos posibles.

No obstante, aunque un servicio de este estilo suene jugoso en cuanto a lo citado anteriormente, puede que haya algunas personas que se sientan un tanto escépticas en cuanto a su éxito. Por lo tanto, me gustaría poder mostrar a continuación una prueba fehaciente de que este tipo de servicios pueden llegar a cosechar bastante éxito.

Sin más, me gustaría añadir a continuación algunas estadísticas de la página del servicio conversor de números a español "Números TIP" desarrollada por el tutor a cargo de este proyecto, Francisco Javier Carreras Riudavets.



Page Title	Page Views	Unique Page Views	Avg. Time on Page	Entrances	Bounce Rate	% Exit	Page Value
	18,274,039 % of Total: 38.75% (47,163,679)	9,968,632 % of Total: 52.08% (19,139,506)	00:01:34 Avg for View: 00:01:19 (18.94%)	6,024,260 % of Total: 56.12% (10,734,740)	39.37% Avg for View: 41.68% (-5.54%)	32.91% Avg for View: 22.76% (44.58%)	€0.00 % of Total: 0.00% (€0.00)
1. Números TIP - Convertir número a texto o letras	**10,866,561 (59.46%)**	4,564,978 (45.79%)	00:01:43	1,848,650 (30.69%)	27.56%	26.27%	€0.00 (0.00%)
2. Números TIP - Numbers to words-text converter	**675,063 (3.69%)**	274,756 (2.76%)	00:01:27	122,563 (2.03%)	29.39%	27.57%	€0.00 (0.00%)

ILUSTRACIÓN 1 - ESTADÍSTICAS DE LA PÁGINA DE NUMEROS TIP

Aquí podemos ver la popularidad que goza el servicio alojado en http://tulengua.es/numeros-texto/ con cerca de dieciocho millones de vistas a lo largo de poco más de un año y el primer cuatrimestre del otro llegando a tener picos de casi dos millones de vistas en un corto espacio de tiempo como podemos observar en la **Ilustración 1**.

Aunque en la primera posición tenemos al servicio como tal en su idioma madre, el español, podemos observar cómo la herramienta que nos da el servicio en inglés implementada más tarde también tiene una cantidad más o menos consistente de vistas con más de medio millón de visitas.

ILUSTRACIÓN 2 - VOLUMEN DE VISITAS POR UBICACIÓN

En esta ilustración podemos observar también, de manera mucho más ilustrativa, el volumen de visitas que recibe este servicio por ubicación. Cabe destacar que la medida en la que se hace es usando un azul pastel desde los lugares con un volumen más modesto hasta un azul más intenso para aquellos lugares con la mayor concentración de visitas.

Country		Page Title	Acquisition			Behaviour		
			Users	New Users	Sessions	Bounce Rate	Pages/Session	Avg. Session Duration
			4,277,464 % of Total: 56.91% (7,516,062)	4,262,340 % of Total: 56.61% (7,529,739)	6,012,829 % of Total: 56.01% (10,735,852)	39.59% Avg for View: 41.68% (-5.03%)	3.04 Avg for View: 4.39 (-30.87%)	00:03:12 Avg for View: 00:04:29 (-28.58%)
1.	Mexico	Números TIP - Convertir número a texto o letras	**585,124** (8.08%)	192,098 (4.51%)	437,409 (7.27%)	27.57%	6.10	00:03:58
2.	Colombia	Números TIP - Convertir número a texto o letras	**414,874** (5.73%)	128,273 (3.01%)	393,971 (6.55%)	25.31%	5.72	00:05:32
3.	Argentina	Números TIP - Convertir número a texto o letras	**218,608** (3.02%)	53,685 (1.26%)	131,539 (2.19%)	33.07%	7.24	00:04:02
4.	Venezuela	Números TIP - Convertir número a texto o letras	**199,102** (2.75%)	76,285 (1.79%)	187,673 (3.12%)	24.39%	5.13	00:04:56
5.	Spain	Números TIP - Convertir número a texto o letras	**165,159** (2.28%)	54,373 (1.28%)	94,116 (1.57%)	25.36%	6.47	00:03:30

ILUSTRACIÓN 3 - TOP DE LOS 5 PAÍSES QUE MÁS HAN USADO EL SERVICIO

Como adición podemos ver que los 5 países que ocupan los puestos más altos en el uso de esta aplicación son hispanoparlantes. A partir de aquí podemos ver que el siguiente país que **no** es hispanohablante y que usa la aplicación es Estados Unidos y ocupa el noveno puesto (usando el servicio ofrecido en inglés) y otro sería, por ejemplo, Italia en su versión italiana ocupando el puesto 32.

Estos datos nos ofrecen una visión de lo útil que es un servicio como este y del éxito que puede tener gracias a las estadísticas ilustradas anteriormente.

Por lo tanto, es de esperar que un servicio que ofrezca soporte al idioma francés obtenga resultados mayores a los del italiano debido a que hay una mayor cantidad de personas francófonas a nivel mundial (cerca de 274 millones).

También a nivel europeo ocupa uno de los lugares más importantes a nivel de habla, justo después del inglés o el alemán.

También, por norma general en cualquier país europeo con lengua materna inglesa se divide a partes iguales aprender francés o español, por lo que podemos vaticinar un uso intensivo del servicio a medio-largo plazo.

5 Diseño

El proyecto basa su funcionamiento en la clase "Servicio" que a su vez llama a otras clases de las que se vale para realizar la conversión del número escrito a las diferentes formas descritas anteriormente.

5.1 Entrada – Salida del proceso

Como es de esperar, lo primero que realiza el proceso de traducción e interpretación es el análisis del formato del número (o más bien, de la entrada) que se ha introducido en la página. Una vez llevado a cabo el proceso de análisis, el proceso devuelve tras ciertos trabajos una serie de textos resultado de haber hecho pasar la entrada introducida por las diferentes clases del servicio.

En el caso de que haya habido algún error el proceso devolverá el error que se ha producido al intentar hacer la conversión.

5.1.1 Entrada de datos

La clase *Servicio* tiene un método llamado "getTabs" al que se le pasa por parámetro la entrada de datos que se ha introducido y que se quiere analizar.

El número que se introduce debe de cumplir alguna de las formas citadas a continuación para llevar a cabo su conversión. De lo contrario no se podrá realizar. El orden en el que se expresan las formas es importante.

- Número entero
    - [OPCIONAL] Signo + ó –
    - [OBLIGATORIO] De 1 a 120 dígitos numéricos

- Número decimal
    - [OPCIONAL] Signo + ó –
    - [OBLIGATORIO] De 1 a 120 dígitos numéricos
    - [OBLIGATORIO] Un separador decimal (puede ser un punto, "." o una coma ",")

- [OBLIGATORIO] De 1 a 120 dígitos numéricos

- Número fraccionario
  - [OPCIONAL] Signo + ó –
  - [OBLIGATORIO] De 1 a 120 dígitos numéricos
  - [OBLIGATORIO] Un signo "/" para poder separar las dos partes de la fracción, el numerador y el denominador.
  - [OBLIGATORIO] De 1 a 120 dígitos numéricos

- Número en notación científica
  - [OPCIONAL] Signo + ó –
  - [OBLIGATORIO] De 1 a 120 dígitos numéricos
  - [OPCIONAL] Un separador decimal (puede ser un punto, "." o una coma ",")
  - [OBLIGATORIO]Una letra "E" o "e" para delimitar la parte del exponente
  - [OPCIONAL] Signo + ó –
  - [OBLIGATORIO] De 1 a 3 dígitos numéricos

5.1.2 Salidas

Se comentó anteriormente que el método *getTabs* realiza todo el proceso de conversión. Para ello, este proceso se descompone en varios procesos de conversión dependiendo del análisis previo que se realiza para conocer las características clave del dígito en cuanto a sus posibles formas.

Estas tareas se llevan a cabo simultáneamente haciendo uso de *Threads* o hilos para el procesamiento de manera concurrente.

Todas estas tareas generan una serie de datos que, finalmente, confluyen en un conjunto de datos que forman el resultado devuelto por el proceso.

En cualquier caso, el tipo de dato que se devuelve se corresponde con la clase ArrayList por su sencilla manera de operar con los componentes que lo forman y la posibilidad de ir añadiendo resultados al mismo sin tener que poseer un tamaño fijado antes de empezar el proceso.

De todas formas, el ArrayList que se devuelve puede ser de dos tipos:

- ArrayList formado por las diferentes interpretaciones del número introducido.
- ArrayList formado por los posibles errores que se han podido dar en cuanto al procesamiento del número introducido en el servicio.

5.1.3 Tipos de salidas

Los resultados que devuelve el servicio varían en gran medida de la longitud del número escrito y del formato en el que se haya hecho (no es lo mismo convertir un número entero que uno en notación científica). En cualquier caso, se ha intentado llegar a la misma cantidad de números tanto en la escala pequeña como en la escala grande. El límite del servicio está en los 1000 vigintillones menos 1.

5.2 Vista del proceso de conversión

Existen diferentes maneras de obtener los resultados que dependen del formato del número, por lo que es conveniente poder explicarlas de antemano con la siguiente ilustración para mayor facilidad a la hora de comprenderlo.

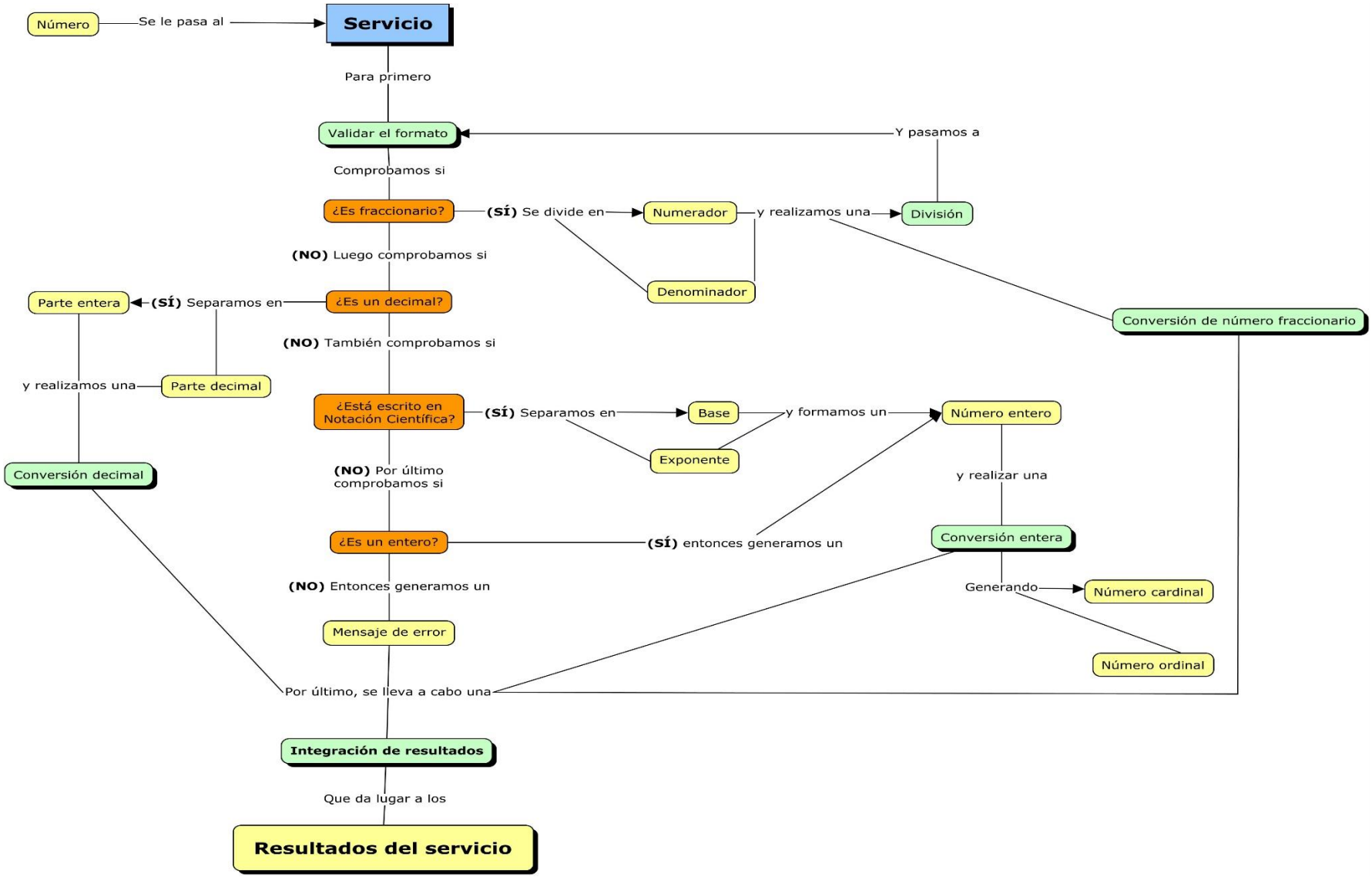


Ilustración 4 - Procesamiento que hace el servicio

En el caso de que no se haya explicado correctamente el diagrama presentado anteriormente se procede a una breve explicación del servicio.

1. Se crea una instancia de la clase “Servicio” y se llama al método que tiene
2. Una vez llamado el método, se procede a analizar el texto y formar sus resultados
    a. Se comprueba si el número que se ha introducido está en formato fraccionario.
        i. En el caso de que lo sea formamos dos resultados:
            1. Numerador
            2. Denominador
        ii. Llevamos a cabo la división entre los números y volvemos al paso 2 anterior.
        iii. Paralelamente llevamos a cabo la conversión de un número fraccionario y obtenemos los resultados
    b. Se comprueba si el número que se ha introducido está en formato decimal.
        i. En el caso de que lo sea formamos:
            1. Parte entera
            2. Parte decimal
        ii. Llevamos a cabo la conversión de un número decimal y obtenemos los resultados.
    c. Se comprueba si el número que se ha introducido está en formato exponencial o notación científica.
        i. En el caso de que lo sea obtenemos:
            1. Base
            2. Exponente
        ii. Se convierte a número entero.
        iii. Llevamos a cabo la conversión a número entero y obtenemos los resultados.
    d. Se comprueba si el número que se ha introducido está en formato entero
        i. En el caso de que lo sea obtenemos
            1. Número entero
        ii. Llevamos a cabo la conversión a número entero y obtenemos los resultados
    e. Si no se ha cumplido ningún punto anterior entonces generamos un mensaje de error
3. Se recopilan los resultados obtenidos
    a. Generamos el conjunto de resultados del servicio.

5.3 Estructura de la clase servicio

El servicio que vamos a dar viene dado por la clase autogenerada “Servicio”. En esta clase es donde llevamos a cabo el proceso de conversión de un número a sus diferentes posibles formas.

No obstante, para poder convertir el número que hemos introducido a las diversas formas que antes hemos descrito, ha hecho falta la implementación de diversas clases de las que hace uso el servicio para poder llevar a cabo su tarea.

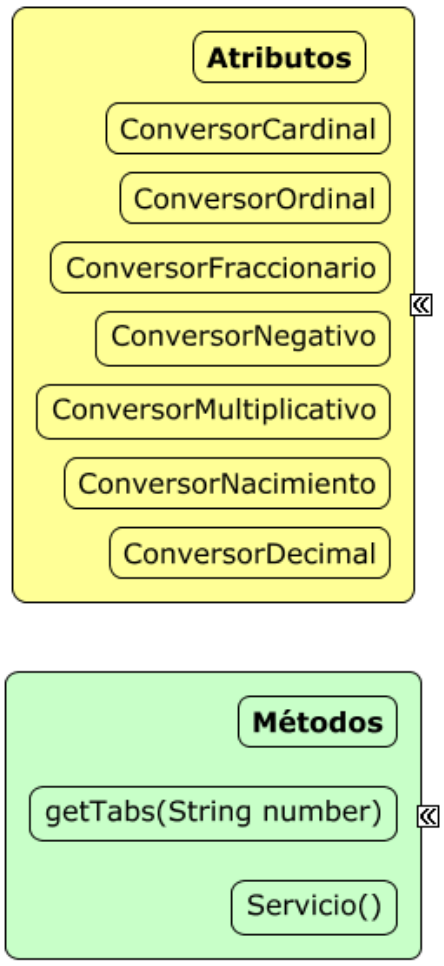


ILUSTRACIÓN 5 - ESTUCTURA DE LA CLASE SERVICIO

Como podemos observar en la ilustración, la clase Servicio tiene siete atributos, uno por cada una de las formas que soporta el servicio que se ha implementado. Por otra parte, también

tenemos que hablar de los métodos que esta clase implementa. En el caso de la clase Servicio encontramos solamente dos métodos: getTabs y Servicio.

- GetTabs es un método público al que llamamos a la hora de llevar a cabo la conversión de un número. Por parámetro se le envía el número que queremos convertir y nos devolverá una lista con las posibles conversiones del dígito en cuestión.

- Por último, Servicio es un simple constructor de la clase el cual no tiene demasiada importancia en este aspecto ahora mismo.

Cada uno de los atributos es la instanciación de una clase a parte que lleva a cabo su propia conversión de manera paralela (explicado más detalladamente en el apartado de Desarrollo más adelante).

5.3.1 Clase Cardinal

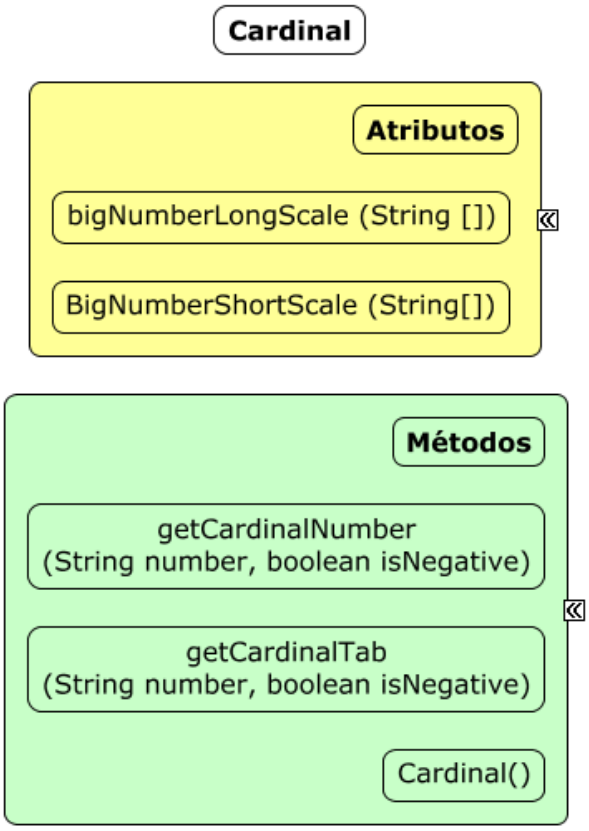


ILUSTRACIÓN 6 ESTRUCTURA DE LA CLASE CARDINAL

Aquí podemos observar la estructura de la clase de convertir un número en sus posibles equivalencias en formato cardinal.

Esta clase posee dos atributos esenciales que son las escalas de los números grandes ya comentadas en el punto 1. Cada uno de los atributos es un contenedor con las ristras de caracteres que tienen que ver con cada una de las escalas que vamos a tener en cuenta.

Por otra parte, esta es una clase que posee tres métodos.

- **Cardinal ()** es el constructor de la clase. La usamos a la hora de poder instanciarla e inicializar los contenedores con los números.
- **getCardinalNumber(…)** es un método público de la clase usado para, tras mandarle por parámetro el dígito del que queremos saber la conversión y si es negativo o no, obtener las posibles conversiones del número.
- **getCardinalTab(…)** es un método público que es usado por el servicio a la hora de poder organizar los resultados. Necesita de 2 parámetros: el número en cuestión y si es negativo o no. En este caso lo que obtenemos es la información de deberá salir en pantalla una vez se haya realizado la conversión.

5.3.2 Clase Ordinal

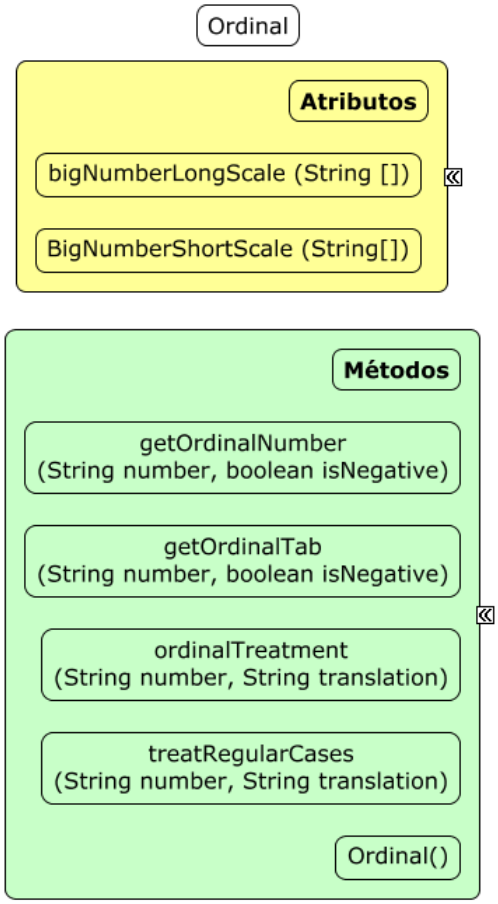


ILUSTRACIÓN 7 ESTRUCTURA DE LA CLASE ORDINAL

A través de esta ilustración se describe a grandes rasgos la estructura de la clase que lleva a cabo la conversión de un número a su equivalente (o equivalentes en el caso de que así fuera) al formato ordinal.

Esta clase posee dos atributos esenciales que son las escalas de los números grandes ya comentadas en el punto 1. Cada uno de los atributos es un contenedor con las ristras de caracteres que tienen que ver con cada una de las escalas que vamos a tener en cuenta.

Por otra parte, podemos destacar cinco métodos de esta clase:

- **getOrdinalNumber(…)** es un método usado para la misma función que su método parecido en la clase cardinal pero aplicado a números ordinales. Como entrada necesita del número que queremos convertir y de si es negativo o no.
- **getOrdinalTab(…)** es un método público que nos devuelve la información que vamos a mostrar a la hora de devolver los resultados de la conversión. Como entrada necesita del número que queremos convertir y de si es negativo o no.

- **ordinalTreatment(…)** es un método privado de la clase que lo que hace es aplicar las reglas que se tienen que tener en cuenta a la hora de formar un número ordinal en francés.
- **treatRegularCases(…)** es un método privado que, si bien podría formar parte del método anterior, se ha decidido encapsular en otro método la aplicación de las reglas generales de los ordinales en caso de que no se haya podido hacer el tratamiento antes.
- **Ordinal ()** simplemente es el constructor de la clase usado para poder instanciarla e inicializar los contenedores con las escalas numéricas.

5.3.3 Clase Fraccionario

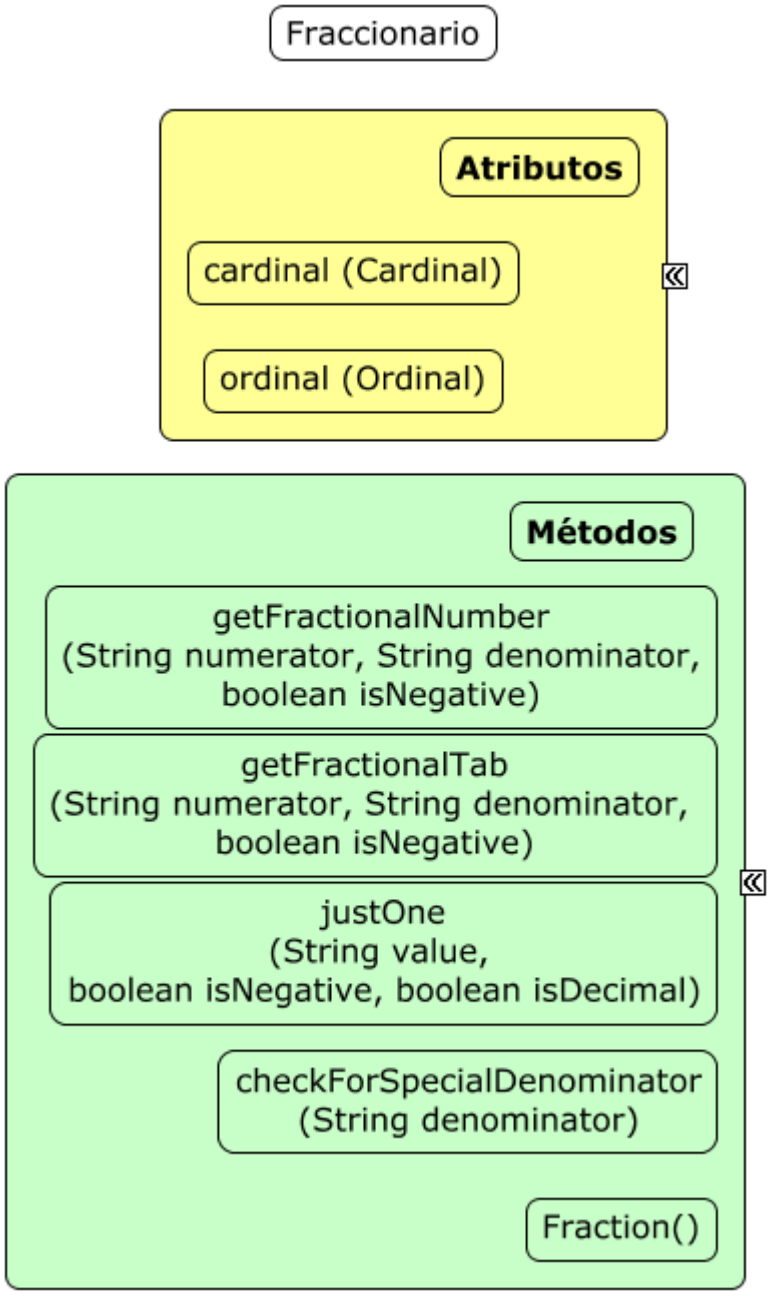


ILUSTRACIÓN 8 ESTRUCTURA DE LA CLASE FRACCION

Con esta ilustración se describe de manera bastante gráfica la estructura que tiene la clase que convierte un número a su formato (o varios) fraccional.

Como podemos observar en la ilustración superior, posee dos atributos principales:

- Un conversor a cardinal (usado a la hora de convertir el numerador de la fracción)

- Un conversor a ordinal (usado, por otra parte, para convertir el denominador a su forma correcta en fraccional).

Por otra parte, tenemos que destacar ciertos métodos implementados en esta clase, estos son:

- **getFractionalNumber(…)** es un método que es utilizado para obtener las diferentes formas de poder nombrar el número que hemos introducido en su formato fraccional. Desde aquí se accede a las funciones *justOne* o *checkForSpecialDenominator* más abajo explicadas.
- **getFractionalTab(…)** es el método principal de la clase, ya que es el que se encarga de recoger la información relativa a la conversión a formato fraccional del número para mostrarla al usuario.
- **justOne(…)** es un método privado al que se llama siempre que falte alguno de los componentes de la fracción (ya sea el numerador, que se trataría con el formato **x/1** o bien el denominador, que se trataría con el formato **1/x**).
- **checkForSpecialDenominator(…)** es un método privado que comprueba si, como vimos en el primer punto del documento, el denominador que hemos introducido es alguno que contiene ciertas excepciones.
- **Fraction ()** es el constructor de la clase usado para instanciarla e inicializar los conversores cardinal y ordinal que contiene esta clase.

5.3.4 Clase Multiplicativo

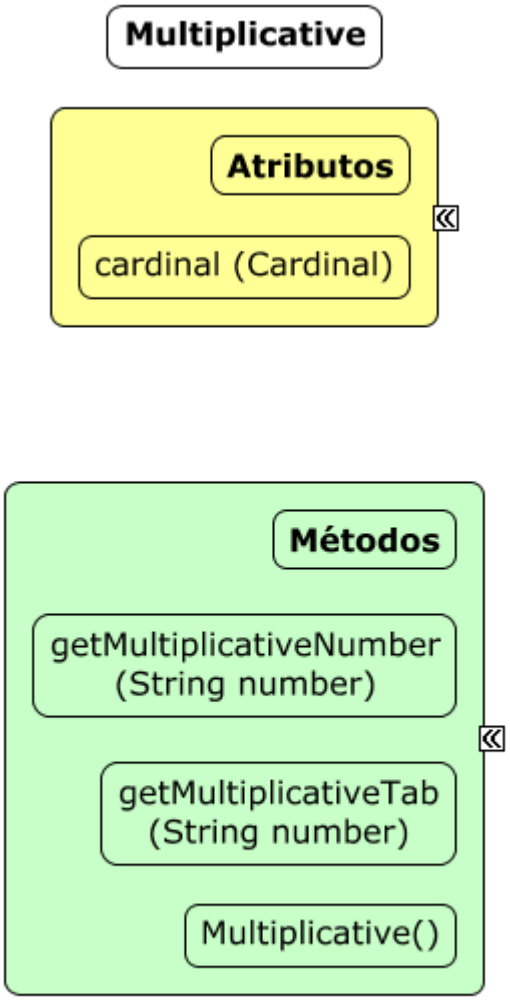


ILUSTRACIÓN 9 ESTRUCTURA DE LA CLASE MULTIPLICATIVE

Gracias a esta ilustración podemos observar, de la manera más gráfica posible, la estructura de la clase que lleva a cabo la conversión de un número mandado por parámetro a su forma multiplicativa.

Como es de esperar, esta clase es algo más simple que las anteriores y posee solamente un atributo:

- Un conversor a cardinal (usado para poder llevar a cabo la conversión de un número a su forma multiplicativa a partir de su forma cardinal)

Y, por otra parte, cabría hablar de los métodos implementados en esta clase:

- **getMultiplicativeNumber(…)** es el método usado para poder obtener las posibles formas multiplicativas de un número.

- **getMultiplicativeTab(…)** es el método principal de la clase, usado para poder obtener la información que va a ser presentada al usuario en el caso de poder tener un número multiplicativo a partir de la conversión de lo que introdujo por teclado.
- **multiplicative()** es el constructor de la clase. Simplemente se usa para poder instanciar la clase y poder inicializar el conversor a cardinal que está alojado aquí.

5.3.5 Clase Decimal

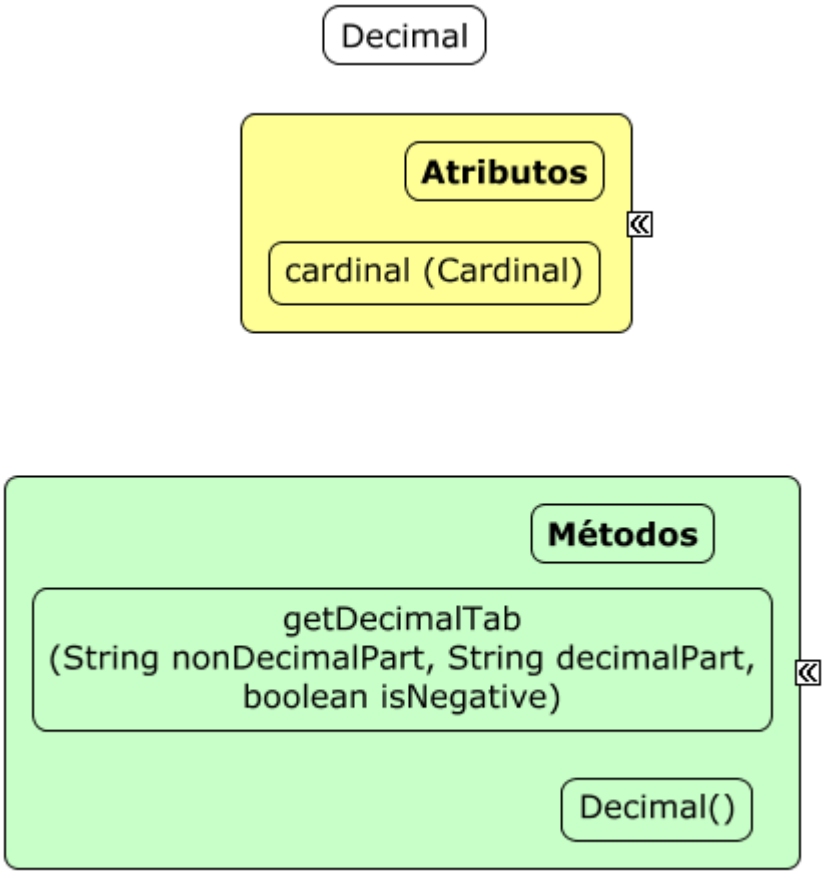


ILUSTRACIÓN 10 ESTRUCTURA DE LA CLASE DECIMAL

Como se observa en esta foto, la clase decimal posee un atributo y dos funciones.

Como atributos podemos observar:

- Conversor a cardinal para llevar a cabo la conversión de un número a su formato decimal. Este conversor se usa para convertir la parte entera como la decimal al formato correcto.

Como funciones, tenemos que destacar:

- **getDecimalTab (…)** es el método principal de la clase, usado para poder recopilar las formas en las que se ha convertido el número introducido en el servicio a su formato decimal si lo requiere.

- **decimal ()** es el constructor de la clase, se usa para poder instanciarla y así inicializar el conversor a cardinal que la clase utiliza.

5.3.6 Clase Negativo

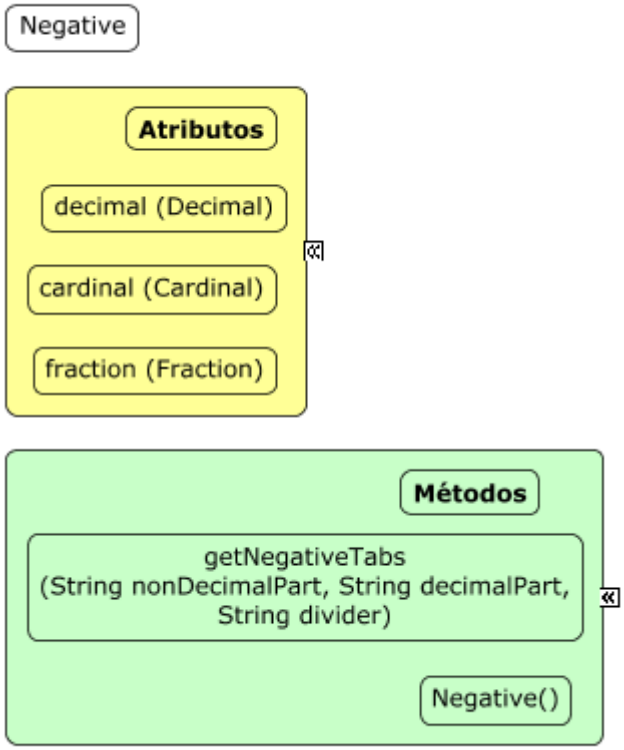


ILUSTRACIÓN 11 ESTRUCTURA DE LA CLASE NEGATIVO

Tomando como referencia la ilustración de arriba, podemos describir de la manera más concreta posible los rasgos principales de la clase Negativo. Como se puede observar, podemos ver que posee tres atributos y dos métodos que vale la pena destacar.

Por parte de los atributos hay que mencionar:

- Conversor a cardinal para poder llevar a cabo la conversión de un número en el caso de que sea entero a su forma cardinal negativa. Este formato se usa solo en caso de que el número posea **solamente** parte entera. Es decir, que ambos, divisor y parte decimal sean cadenas vacías.

- Conversor a fraccionario para poder llevar a cabo la conversión de un número en el caso de que tenga formato fraccionario para poder obtener sus formas negativas. En caso necesario se llevará a cabo la división de los números para obtener, por otra parte, su forma negativa decimal del mismo modo.
- Conversor a decimal para poder llevar a cabo la conversión de un número en el caso de que tenga formato de número decimal para poder obtener sus formas negativas. Este conversor es utilizado en el caso de que tenga parte decimal o después de comprobar que la división (en el caso de que haya fracción) de un resto diferente a 0.

Una vez mencionados los atributos, cabe hablar de dos funciones:

- **getDecimalTabs(…)** es la función principal de la clase encargada de recopilar las pestañas que se le mostrarán al usuario en el caso de que se haya introducido un número con formato negativo.
- **Negative()** es la función constructora de la clase, se utiliza para poder instanciarla y así inicializar los diversos conversores de dentro.

5.3.7 Clase Nacimientos

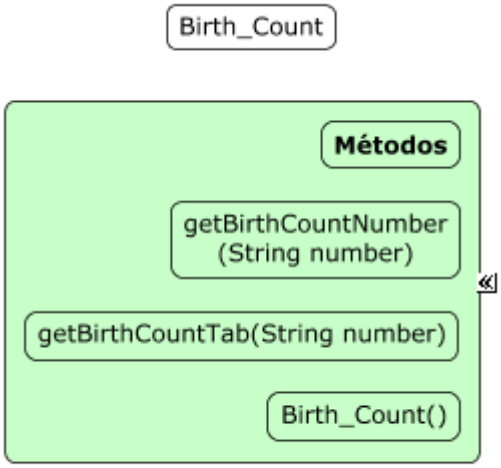


ILUSTRACIÓN 12ESTRUCTURA DE LA CLASE DE NACIMIENTOS

Gracias a esta ilustración podemos hacernos una idea de la simple estructura que tiene la clase Nacimientos. Esta no tiene atributos y tiene dos funciones para tener en cuenta:

- **getBirthCountNumber(…)** es la función usada por la de más abajo y que trata de obtener la conversión del número a formato de número de nacimientos en un mismo parto.

- **getBirthCountTab(…)** es el método principal de la clase, se usa para recopilar la información que será presentada al usuario con las formas en las que se puede escribir el número que ha introducido.
- **BirthCount()** es la función constructora de la clase, se usa solamente para poder instanciarla y así llevar a cabo la conversión del número a su formato correspondiente.

5.4 Formato de salida

Como vimos en el apartado de *salidas* la clase servicio nos devuelve un contenedor con los resultados que se van a mostrar al usuario con la conversión del número que ha introducido. Por otra parte, conviene echar un vistazo a la manera en la que están representados los datos para comprender cómo se han estructurado.

- En primera instancia, la primera aparición de una almohadilla (“#”) implica un título nuevo dentro del cliente. Como sería, por ejemplo, los resultados de la parte de los números cardinales.
- Tras el título viene un contenedor con los siguientes parámetros:
    - Si hay una almohadilla (“#”) se trata de un subtítulo.
    - Si hay dos ampersand (“&&”) se trata de un título en otro color para designar otras maneras en las que el número se ha convertido.
    - Si no tiene nada se trata de un resultado realizado por la conversión

Tras esta breve explicación se muestra la estructura de, por ejemplo, 3 números escogidos al azar obtenido por el método *getTabs* del servicio.

1. 530 765 255 405

[0]	Count = 7
[0]	"#Cardinal"
[1]	"Los números cardinales expresan cantidad en relación con la serie de los números naturales."
[2]	"#Número traducido a texto cardinal. (French)"
[3]	"cinq-cent-trente-milliards-sept-cent-soixante-cinq-millions-deux-cent-cinquante-cinq-mille-quatre-cent-cinq"
[4]	"@cinq cent trente milliards sept cent soixante cinq millions deux cent cinquante cinq mille quatre cent cinq"
[5]	"&&Otras formas de decirlo:"
[6]	"cinq-cent-trente-billions-quatre-cent-cinq"
Vista sin formato	
[1]	Count = 5
[0]	"#Ordinal"
[1]	"Los números ordinales expresan orden o sucesión e indican el lugar que ocupa el elemento en una serie ordenada."
[2]	"#Número traducido a texto ordinal.(French)"
[3]	"cinq-cent-trente-milliards-sept-cent-soixante-cinq-millions-deux-cent-cinquante-cinq-mille-quatre-cent-cinquième"
[4]	"@cinq cent trente milliards sept cent soixante cinq millions deux cent cinquante cinq mille quatre cent cinquième"
Vista sin formato	
[2]	Count = 5
[0]	"#Multiplicativo"
[1]	"Los números multiplicativos expresan que el sustantivo al que se refieren se compone de tantas unidades o implica tantas repeticiones como el numeral indica."
[2]	"#Número traducido a texto multiplicativo. (French)"
[3]	"cinq-cent-trente-milliards-sept-cent-soixante-cinq-millions-deux-cent-cinquante-cinq-mille-quatre-cent-cinq fois"
[4]	"@cinq cent trente milliards sept cent soixante cinq millions deux cent cinquante cinq mille quatre cent cinq fois"
Vista sin formato	
[3]	Count = 7
[0]	"#Fraccionario"
[1]	"Los números fraccionarios expresan división de un todo en partes y designan las fracciones iguales en que se ha dividido la unidad."
[2]	"#Número traducido a texto fraccionario o partitivo. (French)"
[3]	"cinq-cent-trente-milliards-sept-cent-soixante-cinq-millions-deux-cent-cinquante-cinq-mille-quatre-cent-cinquième"
[4]	"@cinq cent trente milliards sept cent soixante cinq millions deux cent cinquante cinq mille quatre cent cinquième"
[5]	"#Valor numérico"
[6]	"1.88407208236898E-12"

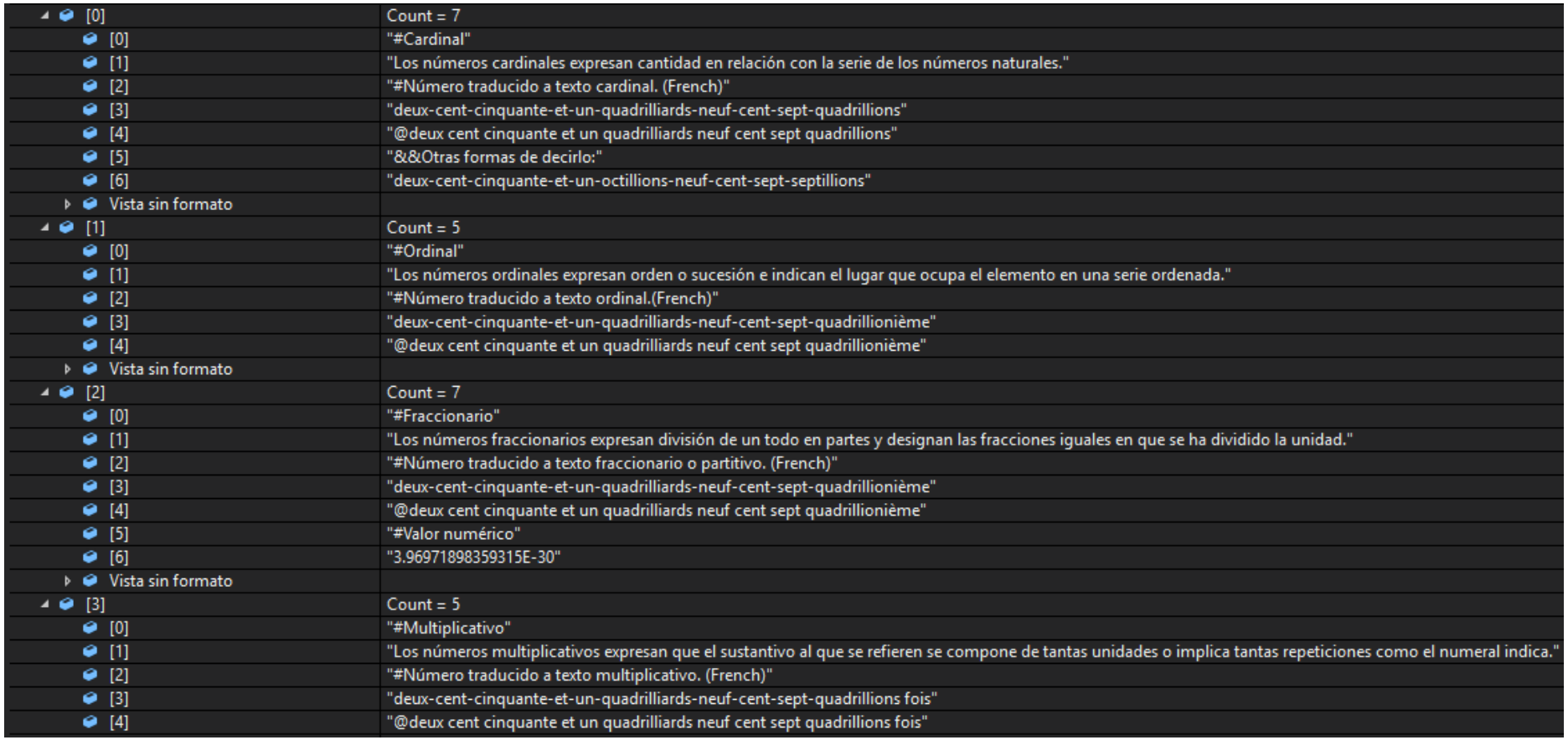
ILUSTRACIÓN 13 CONVERSIÓN DEL NÚMERO 530 765 255 405

2. -325/621

[0]	Count = 5
[0]	"#Decimal"
[1]	"Los números decimales expresan una cantidad en relación con la serie de los números naturales más una fracción de una unidad separada por una coma o un punto."
[2]	"#Número traducido a texto decimal.(French)"
[3]	"zéro-virgule-cinq-cent-vingt-trois-billions-trois-cent-quarante-neuf-milliards-quatre-cent-trente-six-millions-trois-cent-quatre-vingt-douze-mille-neuf-cent-quinze"
[4]	"@zéro virgule cinq cent vingt trois billions trois cent quarante neuf milliards quatre cent trente six millions trois cent quatre vingt douze mille neuf cent quinze"
Vista sin formato	
[1]	Count = 7
[0]	"#Fraccionario"
[1]	"Los números fraccionarios expresan división de un todo en partes y designan las fracciones iguales en que se ha dividido la unidad."
[2]	"#Número traducido a texto fraccionario o partitivo. (French)"
[3]	"moins trois-cent-vingt-cinq six-cent-vingt-et-unièmes"
[4]	"@moins trois cent vingt cinq six cent vingt et unièmes"
[5]	"#Valor numérico"
[6]	"-0.523349436392915"

ILUSTRACIÓN 14 CONVERSIÓN DEL NÚMERO -325/621

3. 251907e24



[0]	Count = 7
[0]	"#Cardinal"
[1]	"Los números cardinales expresan cantidad en relación con la serie de los números naturales."
[2]	"#Número traducido a texto cardinal. (French)"
[3]	"deux-cent-cinquante-et-un-quadrilliards-neuf-cent-sept-quadrillions"
[4]	"@deux cent cinquante et un quadrilliards neuf cent sept quadrillions"
[5]	"&&Otras formas de decirlo:"
[6]	"deux-cent-cinquante-et-un-octillions-neuf-cent-sept-septillions"
Vista sin formato	
[1]	Count = 5
[0]	"#Ordinal"
[1]	"Los números ordinales expresan orden o sucesión e indican el lugar que ocupa el elemento en una serie ordenada."
[2]	"#Número traducido a texto ordinal.(French)"
[3]	"deux-cent-cinquante-et-un-quadrilliards-neuf-cent-sept-quadrillionième"
[4]	"@deux cent cinquante et un quadrilliards neuf cent sept quadrillionième"
Vista sin formato	
[2]	Count = 7
[0]	"#Fraccionario"
[1]	"Los números fraccionarios expresan división de un todo en partes y designan las fracciones iguales en que se ha dividido la unidad."
[2]	"#Número traducido a texto fraccionario o partitivo. (French)"
[3]	"deux-cent-cinquante-et-un-quadrilliards-neuf-cent-sept-quadrillionième"
[4]	"@deux cent cinquante et un quadrilliards neuf cent sept quadrillionième"
[5]	"#Valor numérico"
[6]	"3.96971898359315E-30"
Vista sin formato	
[3]	Count = 5
[0]	"#Multiplicativo"
[1]	"Los números multiplicativos expresan que el sustantivo al que se refieren se compone de tantas unidades o implica tantas repeticiones como el numeral indica."
[2]	"#Número traducido a texto multiplicativo. (French)"
[3]	"deux-cent-cinquante-et-un-quadrilliards-neuf-cent-sept-quadrillions fois"
[4]	"@deux cent cinquante et un quadrilliards neuf cent sept quadrillions fois"

ILUSTRACIÓN 15 CONVERSIÓN DEL NÚMERO 2.51907E24

6 Desarrollo

El servicio que se corresponde con esta memoria está escrito en c# mediante el entorno de desarrollo de Microsoft Visual Studio en su versión Enterprise 2017. Entre otras cosas, el código que se ha desarrollado tiene las siguientes características, entre otras, a destacar:

- **Programación orientada a objetos** mediante la implementación de las diversas clases que componen la aplicación que se ha desarrollado.
- **Expresiones regulares** a la hora de analizar la entrada de datos que nos ha proporcionado el usuario.
- **Concurrencia** a la hora de obtener los diversos resultados (refiriéndose a los diversos formatos aplicables) del número introducido.
- **Fraccionamiento y concatenación** del número introducido en grupos de 3 cifras para una mayor rapidez a la hora del análisis de datos y su conversión para poder formar resultados mayores.
- **Integración de resultados** por parte del servicio para agrupar todos los *outputs* que se hayan producido en la aplicación.
- **Implementación del cliente** para poder experimentar de la manera más cómoda posible y con una mejor experiencia de usuario.
- **Cliente responsive** para adaptarse también a las tecnologías móviles.
- **Cliente independiente de los datos** del servicio para no tener que depender de los resultados del servicio que estamos dando.
- **Escucha de los resultados** en el caso de que nuestra curiosidad cruce límites insospechados en algunos casos y queramos saber cómo pronunciar el número convertido a esa lengua.

6.1 Programación orientada a objetos

Tras haber echado un vistazo al punto 5, se espera que haya quedado bastante claro el hecho de la programación de esta aplicación tiene base en la Programación Orientada a Objetos (POO), donde podemos encontrar métodos, propiedades, atributos …

6.2 Expresiones regulares

Antes de poder convertir un número a los diferentes formatos que hemos hablado a lo largo del documento, es necesario (si no imperativo) poder conocer si el número es correcto. La forma en la que se ha decidido hacer es mediante el uso de 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.

Se ha decidido esta forma por diversas razones:

- Es una manera óptima de poder aprovechar los métodos de los paquetes que nos proporciona el entorno de desarrollo y que ayudan mucho a la hora de ahorrar trabajo codificando una gran cantidad de código en forma de condicionales.
- Las expresiones regulares pueden agruparse en diversos campos que pueden tratarse por separado y así tener una forma sencilla de llevar a cabo todo el proceso de análisis.

Las expresiones regulares tienen una larga lista de campos, sintaxis y demás puntos que, creo, no es el objetivo de esta memoria, por lo que creo conveniente pasar a la explicación de cómo se han hecho estas expresiones para detectar:

- Números enteros
    - Formato exponencial
    - Formato entero
- Números decimales
- Números fraccionarios

Esta validación se hace mediante la clase *TratamientoInicialRegExp* que tiene solamente una función con el mismo nombre que la clase. A continuación, se detallan las expresiones regulares empleadas:

- **Números enteros (Formato exponencial)**

    ((\d+)([,]?(\d+)))[eE]([+-]?(\d+))

Caracteres	Opcional	Descripción
(\d+)	**No**	**Uno o más dígitos**
([,]?(\d+)))	**Sí**	**Un separador en forma de coma “,” y uno o más dígitos**
[eE]	**No**	**Una e o una E para marcar el comienzo de la parte exponencial**

([+-]?(\d+))	**No**	**Un signo "+" ó un signo "-" acompañado de uno o más dígitos**

TABLA 14 EXPRESIÓN DE LOS NÚMEROS EN FORMATO EXPONENCIAL

- **Números enteros (Formato entero)**

^(\d{1,120})$

Caracteres	Opcional	Descripción
^		**Ristra que empiece por lo que contiene**
\d{1,120}	**No**	**De 1 a 120 dígitos**
$		**Fin de la cadena**

TABLA 15 EXPRESIÓN DE LOS NÚMEROS EN FORMATO ENTERO

- Números fraccionarios

(\d{1,120})/([-+])?(\d{1,120})

Caracteres	Opcional	Descripción
\d{1,120}	**No**	**De 1 a 120 dígitos**
/	**No**	**Una barra de separación para las divisiones "/"**
[+-]	**Sí**	**Un signo (y solo uno) de + ó –**

TABLA 16 EXPRESIÓN DE LOS NÚMEROS EN FORMATO FRACCIONARIO

- Números decimales

(\d{1,120}[.,])(\d{1,120})

Caracteres	Opcional	Descripción
\d{1,120}	**No**	**De 1 a 120 dígitos**
[.,]	**No**	**Un separador como un punto o una coma**

TABLA 17 EXPRESIÓN DE LOS NÚMEROS EN FORMATO DECIMAL

En el caso de que el número cumpla alguno de estos patrones se descompone en las partes en las que fuera necesario como, por ejemplo:

- Numerador y denominador para los fraccionarios

- Parte entera y decimal para los decimales
- …

6.3 Concurrencia

Como se ha comentado a lo largo y ancho de este documento, lo que se ha implementado es un servicio web, es algo que está al alcance de todos y en cualquier lugar a 3 golpes de click. Esto puede conllevar a una gran demanda por parte de los clientes y que el servidor pueda verse muy usado. Por ello, se intenta que los procesos de conversión sean todo lo rápido y eficiente posibles.

En el caso de la conversión de un número, al tener diferentes formatos podríamos decir que cada formato al que queremos convertir un número es una tarea independiente del resto, por lo que sería conveniente pensar en paralelizarla junto con el resto de sus formatos. Es decir, dividir la carga de un solo procesamiento secuencial de formato 1 por 1 a un procesamiento en el que todos los formatos que sean necesarios se ejecuten a la vez de manera concurrente en el servidor. Esto nos haría ganar un tiempo de respuesta precioso en cuanto a la carga que podríamos evitar.

Para poder implementar la concurrencia usando los recursos que nos brinda el framework de desarrollo .NET es posible hacerlo mediante 2 puntos de vista:

- **Tareas**

  El framework de .NET nos brinda la posibilidad de hacer un proceso asíncrono con el concepto en mente de una tarea y ejecutar tareas de forma simultánea para que la carga en el servidor sea menor. La biblioteca TPL (Task Parallel Library) nos presenta las siguientes ventajas:

  - Uso más eficaz y 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 que admiten el uso de esperas, cancelaciones, continuaciones, control robusto de excepciones, estado detallado, programación personalizada, y más.

- Hilos

  El framework de trabajo .NET también nos da la posibilidad de llevar a cabo procesos en paralelo llevando a cabo la creación de hilos de procesamiento físicos en el servidor. Estos hilos, como las tareas, se ejecutan de manera asíncrona y reciben señales para esperar unos por otros y demás.

Dado que .NET nos ofrece estas dos alternativas, se llevó a cabo una pequeña investigación para saber cuál de las dos era mejor que la otra y, gracias a un análisis de la página Artesanos.de se llegó a la conclusión de que las tareas son mucho menos invasivas que el uso de los hilos de cara al servidor.

Por estas razones y las ventajas habladas del primer elemento, se ha empleado el uso de tareas a la hora de implementar la concurrencia en el servicio.

De cara a las tareas que tenemos en el servicio, podemos destacar las siguientes:

Nombre de la tarea	Descripción
Decimal	Lleva a cabo la conversión de un número a sus formatos decimales
Cardinal	Lleva a cabo la conversión de un número a sus formatos cardinales
Ordinal	Lleva a cabo la conversión de un número a sus formatos ordinales
Multiplicativo	Lleva a cabo la conversión de un número a su formato multiplicativo
Fraccionario	Lleva a cabo la conversión de un número a su formato fraccionario
Nacimiento	Lleva a cabo la conversión de un número a su formato de nacimientos

TABLA 18 LISTA DE TAREAS DEL SERVICIO

Como es de esperar, no se llaman a todas las tareas de manera simultánea, sino que tras analizar el número se llaman a las tareas que se crean pertinentes.

6.4 Fraccionamiento y concatenación

El servicio recibe los resultados de las tareas lanzadas de manera paralela en una variable con distintos resultados como la conversión a cardinal, la conversión a ordinal, etc. No obstante, dentro de la conversión a cardinal, como vimos en las primeras páginas del documento, tenemos tanto la forma usando la escala grande o la escala pequeña y lo mismo para la forma cardinal.

No obstante, la obtención de los resultados no se obtiene de manera directa en la gran mayoría de las ocasiones. De hecho, cuando el número supera los 3 dígitos de longitud, se lleva a cabo un proceso de fragmentación en grupos de hasta 3 dígitos que se convierten por separado secuencialmente.

Antes de llevar a cabo el proceso de conversión se inicializan dos cadenas de texto para poder guardar ahí los resultados, una cadena para la escala grande y otra para la pequeña. La salida de cada una de las conversiones parciales de 3 dígitos es almacenada junto con la información que previamente estaba en la cadena y así ir formando el número poco a poco mediante sus grupos de tres. En el caso de ser necesario (cuando hablamos, por ejemplo, de números con escala pequeña y grande) se almacena también el orden al que pertenece el número (millón, billón, …) y se amplían los resultados también en la otra cadena de texto con los resultados pertinentes en la otra escala.

6.5 Integración de resultados

Como vimos anteriormente, en el servicio se lanzan una serie de tareas asíncronas para conseguir un menor tiempo de respuesta y mejor experiencia de usuario por parte del cliente.

El problema de este tipo de implementación es que, si buscamos algo de orden en los resultados como, por ejemplo, que nuestro primer resultado sea el cardinal, el segundo el ordinal, etc. Tenemos que, primero, esperar a todos los demás resultados que se hayan lanzado.

Este servicio, una vez recibido los resultados de las tareas, lleva a cabo un proceso de ordenación mandar al cliente los datos definitivos en un formato que, creemos, de decreciente importancia:

- Cardinal
- Ordinal
- Multiplicativo
- Fraccionario
- Decimal
- Nacimientos

6.6 Implementación del cliente

Como se puede esperar de un servicio web, su uso se hará mediante un cliente en una página web. Para llevar a cabo el testeo de los resultados del servicio y para dar uso a algunas de las tecnologías emergentes se ha desarrollado, de manera provisional, un cliente de pruebas para poder llevar a cabo las pruebas pertinentes y comprobar el buen funcionamiento de lo que se quiere enseñar.

Para la creación de este cliente se ha hecho uso de un framework de CSS y JavaScript llamado Bootstrap, que nos ofrece una serie de componentes atractivos a la vista y que ya está prácticamente estandarizado en el mercado de las páginas web.

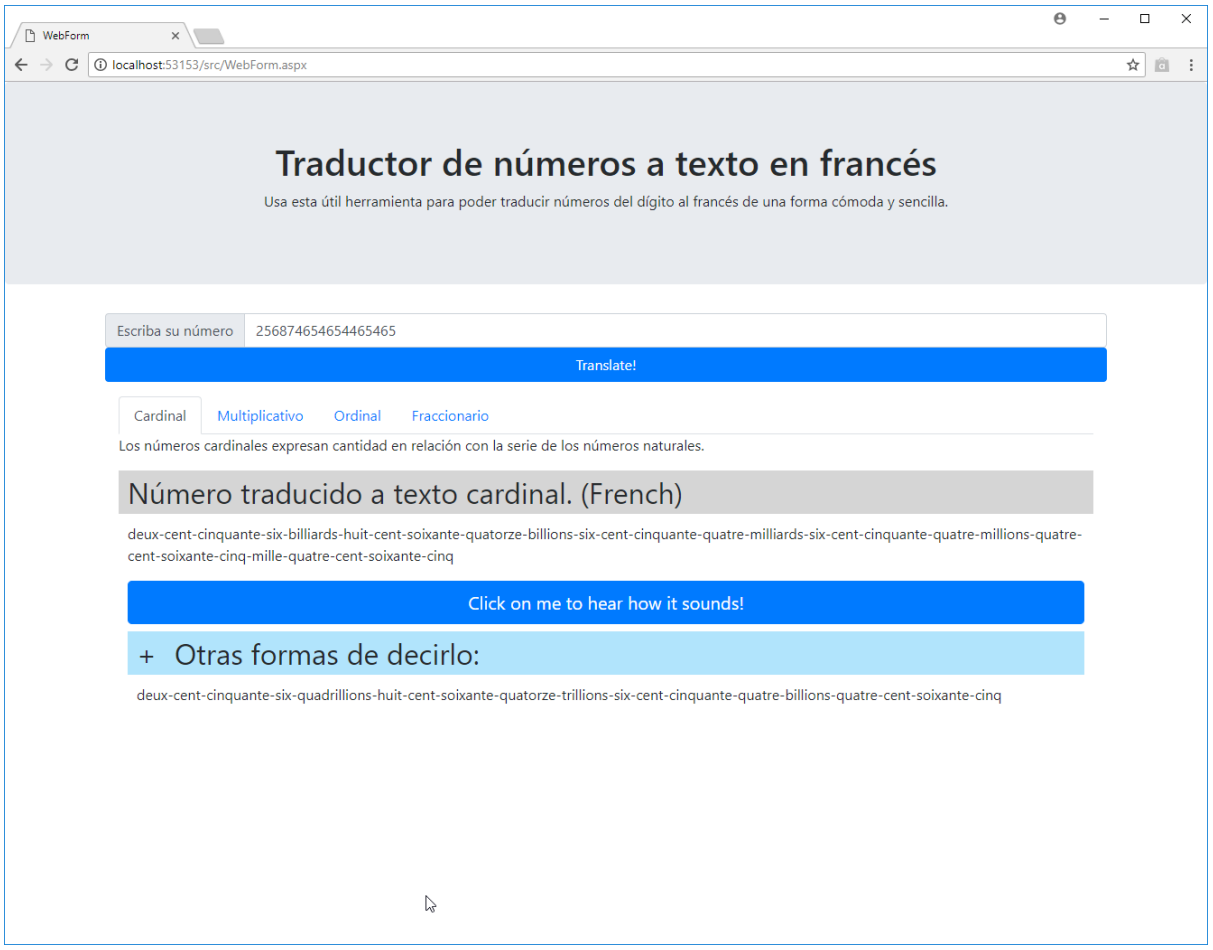


ILUSTRACIÓN 16 ESQUELETO DE LA IMPLEMENTACIÓN DEL CLIENTE.

6.7 Cliente responsive

Por otra parte, dado que cada vez se hace más uso de dispositivos móviles que de aparatos electrónicos como, por ejemplo, los ordenadores, sería conveniente que el cliente se redimensionara correctamente a la hora de mostrar los resultados en un móvil.

Bootstrap es un framework que también apoya el hecho de que las webs sean responsive por lo que ayuda a que su creación de haga en vista de estos dispositivos también. En el caso de que la pantalla sea más pequeña de lo estándar, ciertos breakpoints harán que la página adopte una forma u otra.

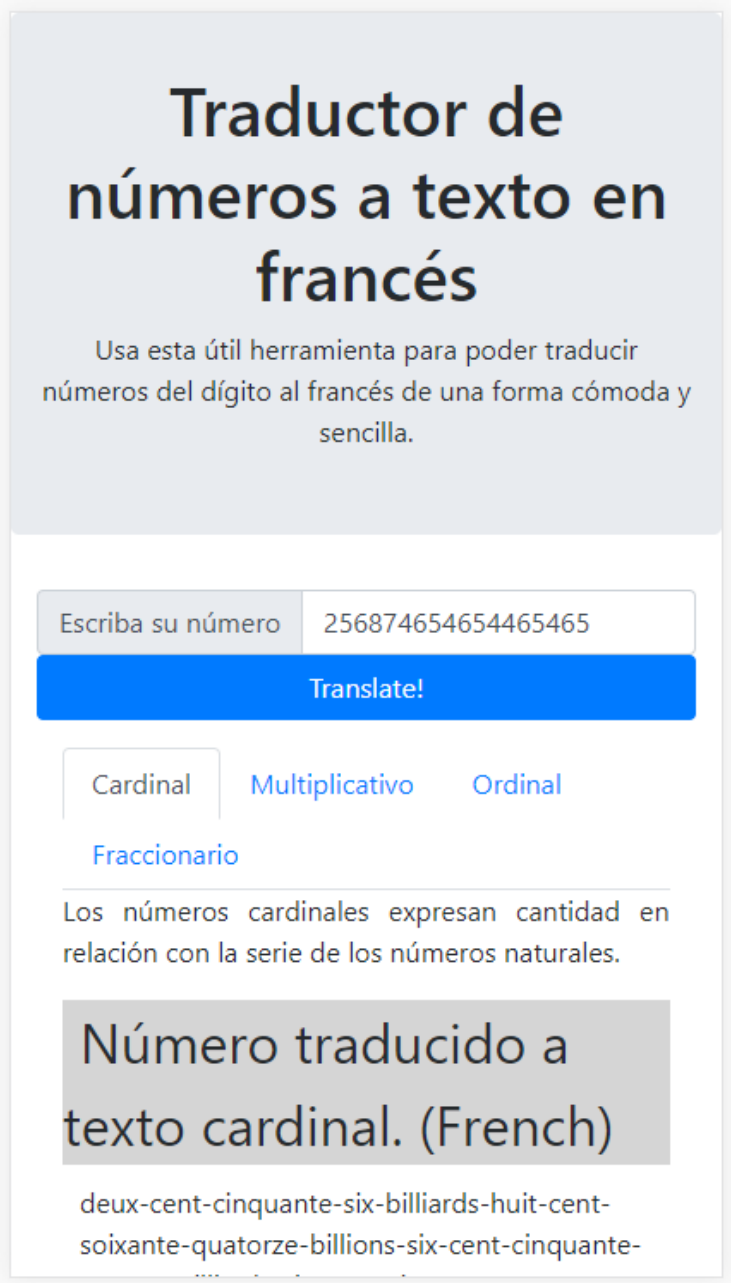


ILUSTRACIÓN 17 VISTA DEL CLIENTE DESDE EL PUNTO DE VISTA DE UN DISPOSITIVO MÓVIL

6.8 Cliente independiente de los datos

Como podemos observar en las ilustraciones anteriores, hay veces en las que un número tiene ciertas pestañas (Cardinal, ordinal, fraccionario, …) u otras (Decimal, …).

El problema al que nos enfrentamos a la hora de convertir un número es que solamente queremos mostrarle al usuario las pestañas pertinentes de las que se ha obtenido algún resultado a tener en cuenta.

Por lo tanto, el cliente se ha programado de tal manera que, una vez recibidos los datos por parte del servidor, éste sea capaz de generar de manera dinámica los elementos HTML pertinentes a la hora de crear estos contenedores.

Cuando hablamos de la creación dinámica de los objetos nos referimos a que cada una de las pestañas se crea porque ha llegado la información de ésta mediante algún código que previamente se ha fijado en la comunicación Cliente-Servidor (se puede ver en el apartado de la salida de los datos citada anteriormente). Es decir, que no hay ninguna pestaña ni contenedor creado más que lo mostrado en la siguiente ilustración:

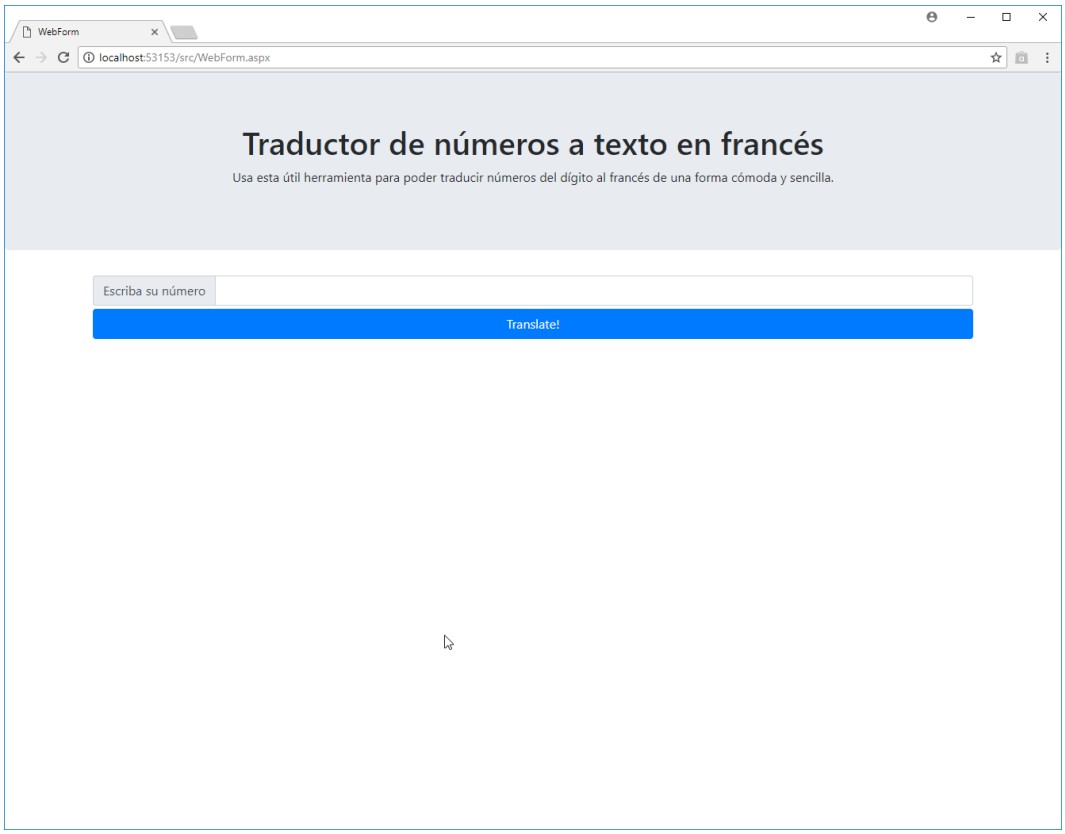


ILUSTRACIÓN 18 ESQUELETO DE LA PÁGINA DEL SERVICIO

Cuando los datos llegan al cliente, se realiza un proceso de análisis de los datos para clasificarlos y generar el contenido pertinente a lo que se necesita visualizar.

6.9 Otros: escucha de los datos

La verdad es que el saber cómo se escribe un número en diversos formatos es bastante gratificante, ya que puede responder a una duda en un momento puntual que puede ser importante en la persona. Pero, claro, a veces no solo queremos saber la escritura del número en cuestión sino también su pronunciación por cualquier otra razón.

La biblioteca de JavaScript ResponsiveVoice es una librería *Text-To-Speech* basada en HTML5 diseñada exclusivamente para añadir este distintivo de voz a páginas web y aplicaciones como smartphones, tablets y ordenadores de sobremesa. Como complemento, soporta 168 voces diferentes, no requiere dependencias y es muy ligero.

La implementación de este curioso añadido se hace mediante un código nuevo en el servidor: la arroba ("@") seguido del texto que hay que pronunciar. En las ilustraciones 13, 14 y 15 se podemos observar como uno de los valores del contenedor es una "@" seguido del número en cuestión.

Dentro del cliente tenemos un caso especial que crea un botón que, una vez pulsado, ejecuta una función de esta biblioteca para pronunciar nuestro contenido.

```
case '@':
    HtmlGenericControl button = new HtmlGenericControl("button");
    button.Attributes["class"] += "btn btn-primary btn-lg btn-block my-2";
    button.Attributes["onclick"] = "responsiveVoice.speak(\'" + text.Substring(1) + "\', \'French Female\');";
    button.InnerText = "Click on me to hear how it sounds!";
    currentContainer.Controls.Add(button);
    break;
```

ILUSTRACIÓN 19 CREACIÓN DEL CONTENIDO DEL BOTÓN DE RESPONSIVE VOICE EN EL CLIENTE

6.10 Otros: internacionalización de la interfaz

El problema de un servicio web como el que queremos ofrecer a los diversos usuarios potenciales es que no todos tienen por qué tener la misma lengua materna o dominar el mismo idioma en el que se ha presentado la web, en este caso, el español.

Por lo tanto, resulta conveniente comentar que se ha desarrollado este servicio usando para ello parámetros de internacionalización para adaptar el contenido de la página que mostramos a tres idiomas en los que el servicio va a ser usado en mayor medida: el francés, el español y el inglés.

También cabe destacar que el idioma por defecto del servicio es el inglés, cosa que es detectado mediante la configuración del navegador. En el caso de que el usuario utilice algún idioma no soportado, se le mostrará el inglés por defecto al ser un lenguaje internacional.

7 Conclusiones

Teniendo en cuenta todo lo que se ha comentado a lo largo de este documento, podemos sacar en claro los siguientes puntos:

1. Aunque, como en prácticamente todas las lenguas, hay excepciones, la escritura en francés de los números se basa en un proceso bastante sistemático.
2. Podemos hablar de dos maneras de poder escribir los números:
    a. Escala grande
    b. Escala pequeña
3. Aunque los términos que se han utilizado para el nombramiento de números sean finitos nos dan la capacidad de referirnos a cantidad tan ingentemente grandes que son prácticamente inimaginables para una persona normal aun estando en el ámbito matemático.

Por último, como este servicio tiene una función similar a la del traductor de números a español, podemos afirmar que se podría replicar para el soporte de otros idiomas que tienen cierta importancia a nivel europeo o mundial.

8 Bibliografía

Escritura de números a francés

Principal conversor de números a texto en francés (0 – 999 999 999 999 999): https://leconjugueur.lefigaro.fr/frnombre.php

Conversor auxiliar de números: http://www.lexisrex.com/French-Numbers/

Conversor auxiliar de números hasta el *sextillion*: http://www.heartandcoeur.com/convert/convert_chiffre_lettre.php

Datos sobre algunos números grandes en francés:

- http://www.btb.termiumplus.gc.ca/tpv2guides/guides/clefsfp/index-fra.html?lang=fra&lettr=indx_catlog_m&page=9-nI6-pQZOTM.html
- https://www.frenchtoday.com/blog/learn-french-larger-numbers-mille-million-audio
- http://www.miakinen.net/vrac/zillions

Escalas grande y pequeña: https://fr.wikipedia.org/wiki/%C3%89chelles_longue_et_courte

Normas para escribir números cardinales y ordinales:

- http://helpmelearnfrench.com/numbers/
- https://study.com/academy/lesson/ordinal-numbers-in-french.html

Normas para escribir fracciones: https://www.lawlessfrench.com/vocabulary/fractions/

Normas para escribir la forma multiplicativa: https://www.lawlessfrench.com/vocabulary/multiplicative-numbers/

Normas para escribir un número en letras o en dígitos: http://www.aproposdecriture.com/ecrire-les-nombres-en-chiffres-ou-en-lettres

Página sobre la reforma de la escritura de 1990: http://sweet.ua.pt/fmart/aparo.htm

Paralelismo y tareas

Documentación sobre las tareas en .NET: https://msdn.microsoft.com/es-es/library/dd537609(v=vs.100).aspx

Documentación sobre los hilos en .NET: https://docs.microsoft.com/en-us/dotnet/standard/threading/using-threads-and-threading

Ejemplos para el trabajo con los hilos:

- https://www.tutorialspoint.com/csharp/csharp_multithreading

- http://www.albahari.com/threading/

Análisis de tareas e hilos en c#: http://artesanos.de/software/2011/10/04/tareas-vs-hilos-en-c-3/

Otros añadidos

Documentación de Bootstrap: https://getbootstrap.com/docs/4.1/getting-started/introduction/

Documentación de ResponsiveVoice: https://responsivevoice.org/text-to-speech-sdk/text-to-speech-play-button/

Introducción a la globalización y la localización en .NET: https://docs.microsoft.com/es-es/aspnet/core/fundamentals/localization?view=aspnetcore-2.0

Ejemplo de uso de localización en c#: https://stackoverflow.com/questions/1142802/how-to-use-localization-in-c-sharp