Portada (institucionalmente establecida)

# Resumen

(máximo de 5 páginas, incluyendo al final Palabras clave y Códigos UNESCO)

# Contenido

[1 Resumen 2](#_Toc79516638)

[2 Contenido 3](#_Toc79516639)

[3 Introducción (donde se incluya los antecedentes y justificación) 5](#_Toc79516640)

[3.1 Antecedentes 5](#_Toc79516641)

[3.1.1 Herramientas de reconocimiento de ecuaciones 5](#_Toc79516642)

[3.1.2 Herramientas de solución simbólica de ecuaciones 5](#_Toc79516643)

[3.2 Justificación 5](#_Toc79516644)

[4 Objetivos 6](#_Toc79516645)

[5 Metodología 7](#_Toc79516646)

[5.1 Análisis del estado del arte 7](#_Toc79516647)

[5.2 Elección de la arquitectura 7](#_Toc79516648)

[5.3 Seguimiento de mejores prácticas 7](#_Toc79516649)

[5.4 Metodologías agile 7](#_Toc79516650)

[5.5 Elección de licencia de código abierto 7](#_Toc79516651)

[*5.6* Escalabilidad 7](#_Toc79516652)

[6 Resultados y discusión 8](#_Toc79516653)

[6.1 Arquitectura 8](#_Toc79516654)

[*6.1.1* *Front end* 8](#_Toc79516655)

[*6.1.2* *Back end* 8](#_Toc79516656)

[6.1.3 Segmentación 8](#_Toc79516657)

[6.1.4 Clasificación 8](#_Toc79516658)

[6.1.5 Reconstrucción 8](#_Toc79516659)

[6.1.6 Resolución 8](#_Toc79516660)

[6.1.7 Distribución de eventos 8](#_Toc79516661)

[6.1.8 Resultados de los tests 8](#_Toc79516662)

[7 Conclusiones 9](#_Toc79516663)

[8 Líneas futuras (opcional) 10](#_Toc79516664)

[9 Bibliografía (incluyendo normativa de aplicación al trabajo) 11](#_Toc79516665)

[10 Planificación temporal y presupuesto 12](#_Toc79516666)

[11 Índice de figuras (opcional) 13](#_Toc79516667)

[12 Índice de tablas (opcional) 14](#_Toc79516668)

[13 Abreviaturas 15](#_Toc79516669)

# Introducción (donde se incluya los antecedentes y justificación)

## Antecedentes

En la actualidad existen multitud de proyectos, herramientas y librerías que cubren tanto el campo del reconocimiento de ecuaciones mediante inteligencia artificial como la resolución de éstas mediante cálculo numérico y simbólico.

A continuación, se incluye explicación de los principales desarrollos que cubren ambos aspectos así como ejemplos de interés de ambos.

### Reconocimiento de ecuaciones

La investigación en reconocimiento de notación matemática tiene más de cinco décadas de recorrido [1]. Las soluciones desarrolladas se pueden clasificar según varios parámetros: naturaleza de las entradas, flujo de reconocimiento y salidas.

#### Clasificación según la naturaleza de los datos de entrada

En el campo de reconocimiento de ecuaciones se pueden distinguir dos enfoques distintos que difieren en el método de recogida de datos.

El primer método denominado *online* recoge datos de tabletas digitalizadoras y similares. El método de reconocimiento se basa en los trazos realizados con el bolígrafo digital y utilizan, entre otros datos de las direcciones de los trazados y el orden (marcas de tiempo) en el que éstos se realizan.

La gran ventaja de estos métodos es que cuentan con una gran cantidad de información añadida para realizar las predicciones. Por el contrario, generan la necesidad de tener dispositivos para la recogida de los datos y, por tanto, no se pueden utilizar en texto escrito en formato analógico.

Por otro lado, el método *offline* utiliza imágenes como fuente de datos lo que lo convierte en un método más flexible. Es relativamente sencillo convertir registros de *online* a *offline*. Sin embargo, el problema de inferir los trazos y marcas de tiempo para convertir registros *offline* a *online* constituiría un problema a resolver en sí mismo.

##### Reconocimiento *online*

El reconocimiento *online* ha sido ampliamente más investigado que el reconocimiento *offline*. Como referente cabe destacar el CROHME (Competición de reconocimiento *online* de expresiones matemáticas manuscritas, por sus siglas en inglés). Se han llevado a cabo 6 competiciones de CROHME en 2011 [1], 2012 [2], 2013 [3], 2014 [4], 2016 [5] y 2019 [6]. En todas ellas se han incluido categorías de reconocimiento de expresiones *online* mientras que solo la última introdujo categoría *offline.*

Actualmente se pueden ver varias implementaciones, tanto comerciales como de código abierto de sistemas que compitieron en las distintas ediciones del CROHME. Entre ellas WIRIS MathType [[1]](#footnote-1) y SESHAT [11] y MyScript Calculator

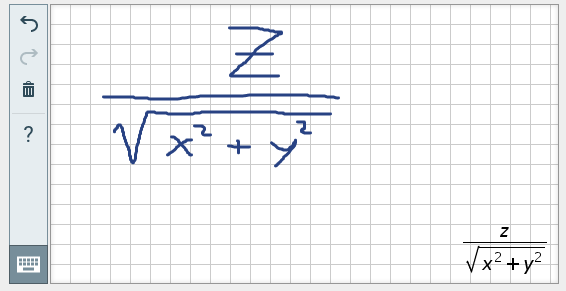


Figura 1: Reconocimiento «online» WIRIS MathType

La aplicación de WIRIS MathType dispone de versión de escritorio y es incrustable en diversos formatos y *frameworks* web.

SESHAT es parte del resultado de una tesis doctoral llevada a cabo en la Universidad Politécnica de Valencia. Es un desarrollo de código abierto bajo licencia GNU alojado en GitHub[[2]](#footnote-2). También hay una implementación disponible en formato web[[3]](#footnote-3).

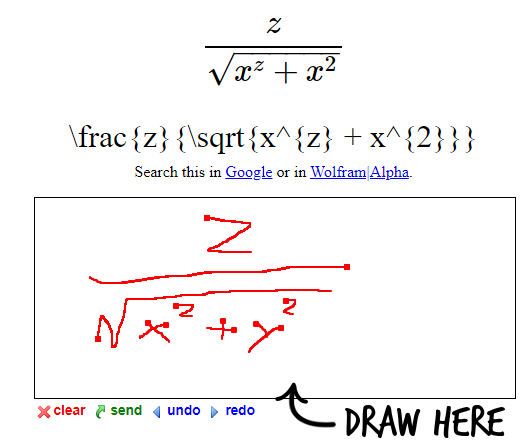


Figura 2: Reconocimiento «online» SESHAT

MyScript[[4]](#footnote-4) Calculator es una app móvil (de pago) disponible en iOS y Android. La aplicación es capaz de resolver ecuaciones sencillas.

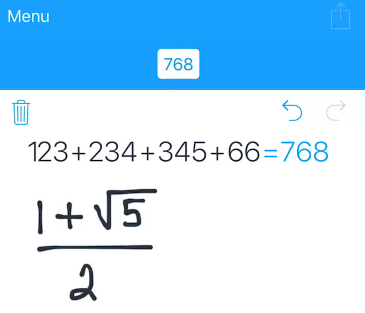


Figura 3 Reconocimiento «online» MyScript

##### Reconocimiento *offline*

La entrada de datos tipo *offline* si bien tiene también una larga trayectoria las publicaciones al respecto son más limitadas y sus resultados en la última competición del CROHME inferiores. Esto es debido principalmente a la ventaja que proporciona a nivel estructural conocer las marcas de tiempo de los trazos realizados.

Entre las implementaciones disponibles encontramos Mathpix Snip[[5]](#footnote-5): Mathpix Snip dispone de aplicaciones de escritorio para Windows, Linux y MacOS, así como apps para Android y iOS.

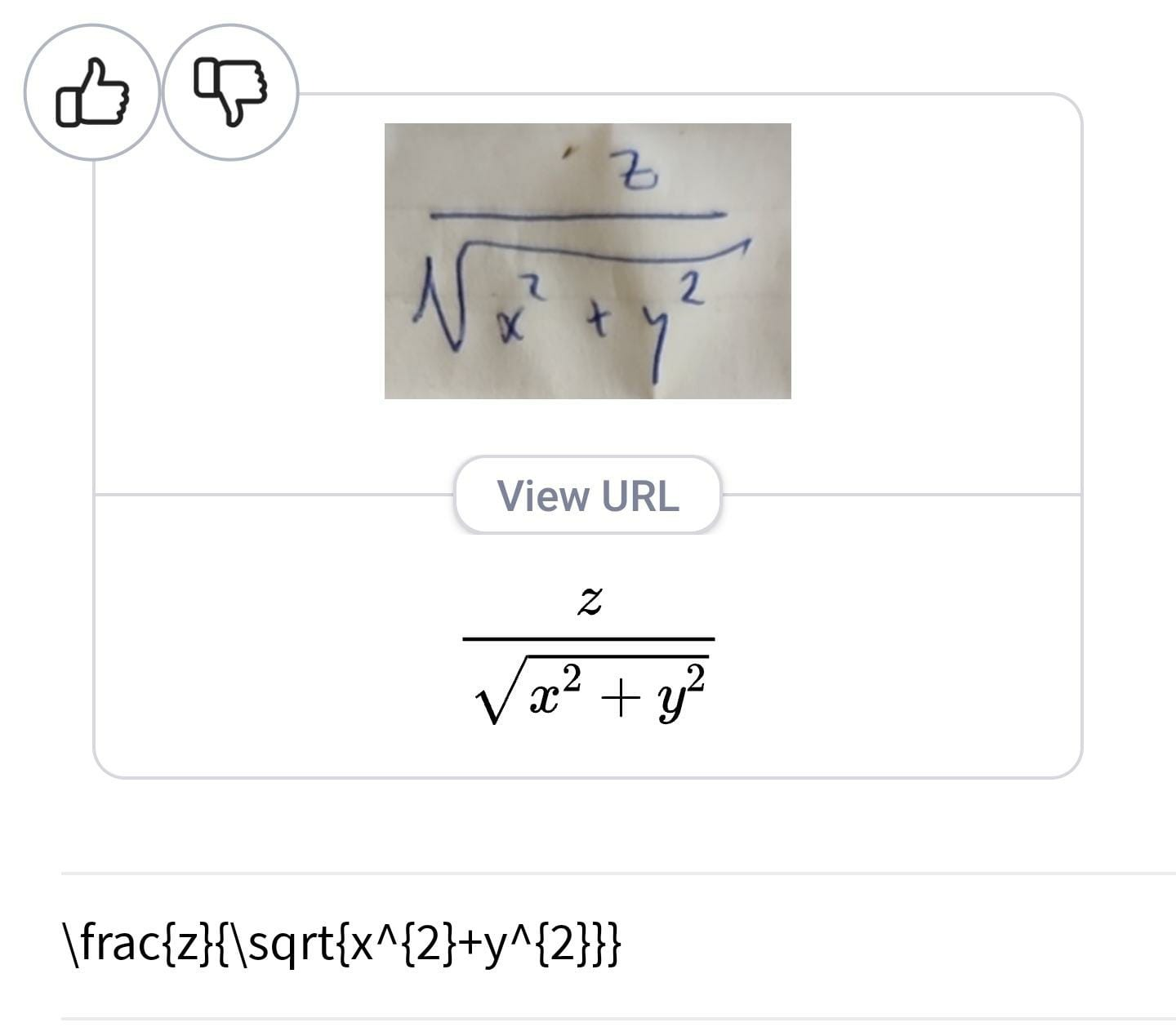


Figura 4: Reconocimiento «offline» Mathpix Snip

#### Clasificación según el flujo de resolución

El reconocimiento de expresiones matemáticas se puede dividir en tres procesos principales: segmentación, donde se divide la imagen en los distintos símbolos que la componen; clasificación, donde se identifica a qué clase pertenece cada símbolo; y reconocimiento de la estructura, donde se buscan las relaciones entre los distintos símbolos.

Las distintas soluciones propuestas se pueden clasificar según el orden en el que estas fases se ejecutan.

Así, es posible distinguir soluciones *top-down* [9] en las que se realiza primero el análisis estructural para después utilizar dicha información en la segmentación y clasificación de símbolos.

En las soluciones *bottom-up* se realiza primero la segmentación, seguida de la clasificación y por último el análisis estructural.

Por último, algunas soluciones realizan simultáneamente el análisis estructural y la clasificación de símbolos [10].

Dentro de cada una de estas fases existen enfoques distintos.

##### Fase de segmentación

En la fase de segmentación se encuentran soluciones basadas en la proyección recursiva de las imágenes (ya sean los datos de entrada o generadas a partir de los mismos) sobre los ejes vertical y horizontal para encontrar las divisiones [11]. Otros enfoques utilizan los cuadros delimitadores para separar los símbolos [12].

En algunos casos se obtienen varias posibles segmentaciones, a las que luego se aplican el resto de fases para seleccionar la segmentación que produzca una mayor probabilidad de acierto [13]. De hecho, en las competiciones de CROHME celebradas entre entre 2011 y 2014 la mayoría de los sistemas participantes utilizaba la información obtenida de la clasificación de símbolos para guiar el proceso de segmentación [14].

##### Fase de clasificación

En la fase de clasificación se pueden encontrar distintos tipos de clasificadores. Los más comunes suelen ser:

* Modelos de *machine learning* (aprendizaje automático), como KNN (k vecinos cercanos), arboles de decisión, SVM (Máquinas de vectores de soporte)…
* Modelos de redes neuronales como CNN (redes neuronales convolucionales) para datos *offline* y RNN (redes neuronales recurrentes) para el caso de datos *online*
* Modelos probabilísticos como los HMM (modelos ocultos de Márkov) [10].

En algunos casos la clasificación se realiza en conjunto con el análisis estructural, como se discute en la siguiente sección

##### Fase de análisis estructural

En los casos en los que la segmentación se ha hecho de forma independiente a la clasificación, la segmentación se suele utilizar para construir un árbol de relaciones con el que luego se construye la secuencia de salida.

Se pueden encontrar casos donde el método de análisis estructural se realiza con métodos de gramática libre de contexto probabilística. Estos métodos ampliamente utilizados en áreas similares como procesamiento de lenguaje natural (NLP) o traducción de textos, definen conjuntos de categorías y símbolos y unas reglas de relaciones entre unos y otros para formar combinaciones válidas según dichas reglas. Las reglas de relaciones llevan asignadas una probabilidad de ocurrencia. Al predecir una secuencia de símbolos se pueden utilizar algoritmos como el CYK (Cocke-Younger-Kasami) para comprobar si la secuencia es válida según las reglas de la gramática, así como para comprobar la probabilidad que tiene la secuencia de ser correcta.

El problema principal de dichos métodos es que son computacionalmente caros. El algoritmo CYK tiene una complejidad de O(n3), donde n es la longitud de la secuencia predicha. Aunque existen técnicas como la programación dinámica que elimina posibles caminos, reduciendo la complejidad, estos modelos siguen resultando lentos cuando se quieren hacer predicciones en «tiempo real».

Para los modelos que realizan todas las fases en conjunto se suelen utilizar modelos de codificador-descodificador con capas de atención. Estos mecanismos son ampliamente utilizados en conversiones de tipo secuencia a secuencia donde los tamaños de las secuencias de entrada y salida son variables.

La función del codificador es extraer las características principales de los datos de entrada codificándolas como un vector de longitud –generalmente– fija. A continuación, el decodificador transforma el vector codificado en una secuencia, de longitud variable.

En estos casos el codificador suele ser una red neuronal convolucional para datos *offline*. Existen multitud de redes utilizadas en estos modelos. Aunque varía según el set de datos utilizados para el entrenamiento, en la siguiente figura se pueden ver los nombres de los distintos tipos de redes convolucionales utilizados en las últimas décadas.

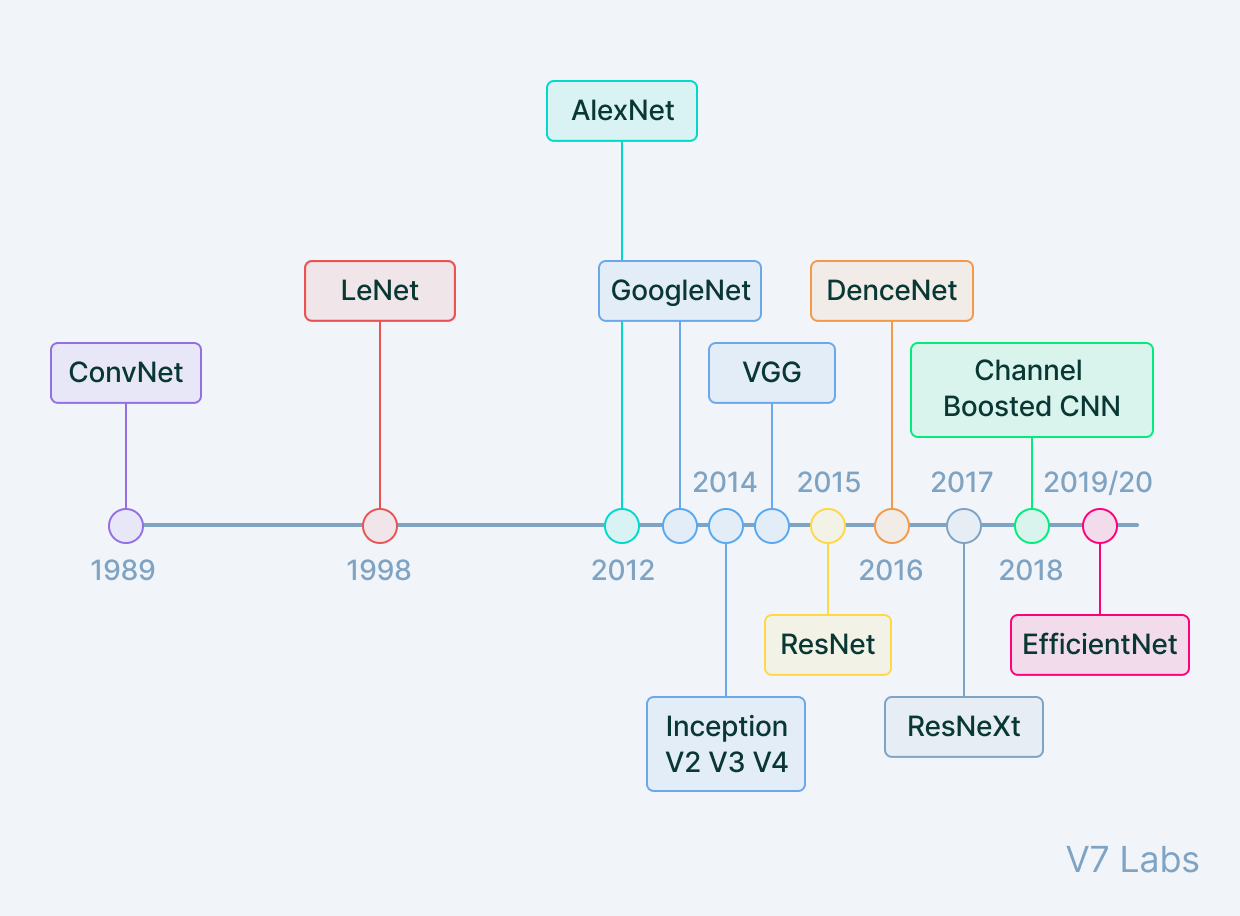


Figura 5: Redes neuronales convolucionales según año de invención[[6]](#footnote-6)

Para datos *online* se suelen utilizar redes neuronales recurrentes (RNN) ampliamente utilizadas en el reconocimiento de secuencias. Para predecir cada símbolo o relación espacial, las redes neuronales recurrentes tienen en cuenta las predicciones anteriores. Para sopesar las diferentes predicciones anteriores según importancia (en general ligadas con la cercanía) se utilizan estructuras llamadas células de memoria a corto y largo plazo (LSTM). Para tener en cuenta las predicciones de los símbolos anteriores y posteriores se utilizan estructuras bidireccionales, que replican las LSTM en ambos sentidos y después sopesan la importancia de la predicción obtenida en cada sentido. Estas estructuras se llaman memoria a corto y largo plazo bidireccionales (BLSTM). Otro tipo de redes con comportamiento similar a las LSTM son las unidades recurrentes cerradas (GRU). Existen soluciones que implementan ambos tipos.

También se aplican mecanismos de atención, que simulan el comportamiento de los humanos, que realizan las predicciones atendiendo a zonas concretas de los datos de entrada, permitiendo resolver secuencias grandes como suma de secuencias pequeñas, manteniendo la precisión y sin elevar excesivamente el coste computacional.

Por último, algunos sistemas usan mecanismos de cobertura para asegurar que no se interprete múltiples veces un mismo símbolo (por ejemplo, si apareciera en dos zonas de atención adyacentes) y no queden símbolos sin predecir (por no aparecer en ninguna zona de atención o aparecer divididos entre varias).

### Herramientas de solución simbólica de ecuaciones

Las herramientas de cálculo simbólico también tienen una historia que abarca varias décadas. La primera versión de Wolfram Mathematica fue publicada en 1988 y la primera versión de Maple en 1982. Desde entonces numerosos avances y alternativas has ido surgiendo.

La capacidad de resolución de ecuaciones diferenciales aportada por las versiones actuales de Maple (2021.1) y Mathematica (12.3.1) son superiores a las de otros sistemas de cálculo simbólico.

Por ejemplo, podemos ver los resultados obtenidos por ambas soluciones para las 1940 ecuaciones diferenciales del libro de Kamke [15] para ambos lenguajes [16] y del subset Kamke 88 (de ecuaciones diferenciales ordinarias de primer orden en forma normal) para la librería de Python SymPy [17] en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Sistema | Versión | Resultado (%) |
| Mathematica | 12.3.1 | 88,04 |
| Maple | 2021.1 | 88,06 |
| Sympy | >=1.6.2 | 39,77 |

Para Sympy se han tenido en cuenta solo los resultados que no fueron expresados en términos de potencias que aumentarían el ratio a 68,18%. Para la versión de Sympy se ha supuesto que tenía que ser menor o igual a la última versión publicada [18] antes de la publicación de los resultados.

A pesar de los mejores resultados de los dos sistemas estos poseen el problema de ser programas bajo licencia, lo cual aumenta los costes y reduce la facilidad de integración y personalización a la hora de ser integradas en sistemas mayores.

La selección de SymPy como comparación se debe a su gran popularidad como alternativa alcanzando las 8.500 estrellas (*stars*) y las 1.300 bifurcaciones (*forks*) en GitHub [19], con mucha diferencia con otras alternativas populares como SageMath u otros proyectos de código abierto. SymPy además cuenta con la ventaja de ser una librería de Python, por lo que para integrarla en el desarrollo de aplicaciones escritas en este lenguaje solo hace falta importarla.

Python además cuenta con grandes ventajas sobre otros lenguajes. En la siguiente tabla se pueden ver los resultados de la encuesta sobre lenguajes de StackOverflow (el foro más popular de programación) de 2020, en la que participaron más de 65.000 desarrolladores, con respecto a Python:

|  |  |  |
| --- | --- | --- |
| Pregunta | Porcentaje | Posición |
| Lo usan y quieren seguir | 66,7 | 3º |
| Quieren aprender | 30 | 1º |

También se puede observar el número de *pull requests* (peticiones de validación) en el segundo trimestre de 2021 [20]:

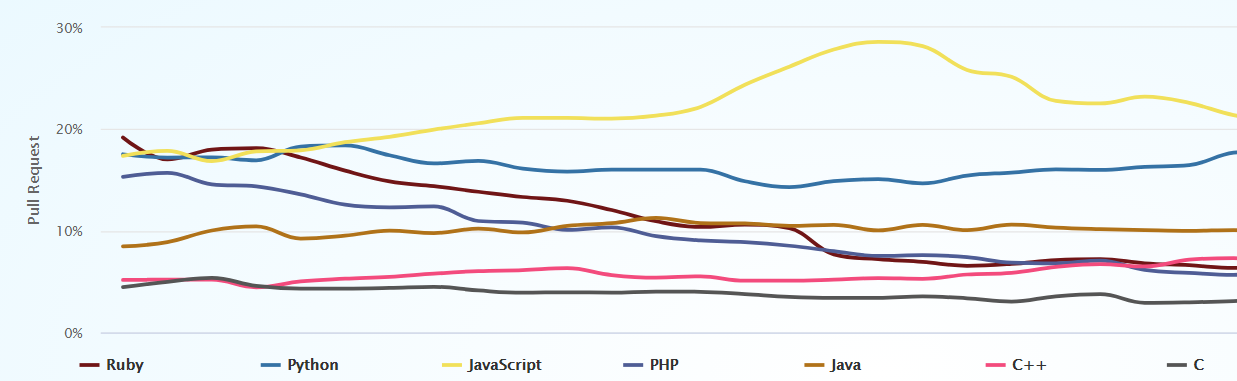


Figura : Peticiones de validación en GitHub por lenguaje en el segundo trimestre de 2021

Donde se puede ver que Python es el segundo lenguaje. O el número de búsquedas en Google en la última década [21]:

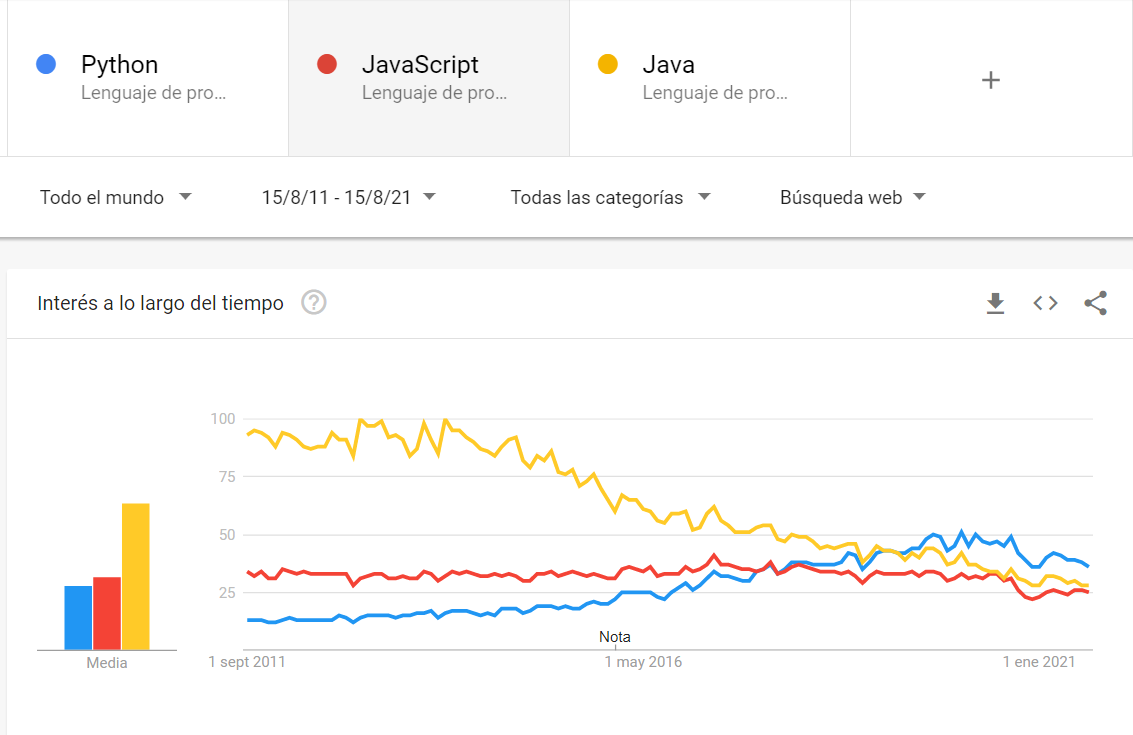


Figura : Búsquedas en Google de lenguajes de programación en el periodo 2011-2021

Donde se puede observar el crecimiento de Python hasta la primera posición.

## Justificación

# Objetivos

1. Crear una aplicación capaz de tomar fotografías de ecuaciones diferenciales y resolverlas.
2. Escribir código que siga las mejores prácticas de programación para asegurar la mantenibilidad del proyecto.
3. Asegurar la escalabilidad de la aplicación mediante el diseño de un proyecto modular que permita balancear la carga entre los distintos módulos.
4. Adquirir una visión global de la administración de sistemas.

# Metodología

## Análisis del estado del arte

## Elección de la arquitectura

## Seguimiento de mejores prácticas

## Metodologías agile

## Elección de licencia de código abierto

## Escalabilidad

# Resultados y discusión

(incluyendo la valoración de impactos y de aspectos de responsabilidad legal, ética y profesional relacionados con el trabajo)

## Arquitectura

### *Front end*

### *Back end*

### Segmentación

### Clasificación

### Reconstrucción

### Resolución

### Distribución de eventos

### Resultados de los tests

# Conclusiones

# Líneas futuras (opcional)

# Bibliografía

|  |  |
| --- | --- |
| [1] | R. H. Anderson, «Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics,» de *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, New York, Association for Computing Machinery, 1967, p. 436–459. |
| [2] | C. V.-G. D. H. K. J. H. K. a. U. G. H. Mouchere, «CROHME2011: Competition on Recognition of Online Handwritten Mathematical Expressions,» de *2011 International Conference on Document Analysis and Recognition*, 2011, pp. 1497-1500. |
| [3] | C. V.-G. D. H. K. J. H. K. U. G. Harold Mouchère, «ICFHR 2012 Competition on Recognition of On-Line Mathematical Expressions (CROHME 2012),» de *2012 International Conference on Frontiers in Handwriting Recognition*, 2012, pp. 811-816. |
| [4] | H. a. V.-G. C. a. Z. R. a. G. U. a. K. D. H. a. K. J. H. Mouchère, «ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions,» de *2013 12th International Conference on Document Analysis and Recognition*, 2013, pp. 1428-1432. |
| [5] | H. a. V.-G. C. a. Z. R. a. G. U. Mouchère, «ICFHR 2014 Competition on Recognition of On-Line Handwritten Mathematical Expressions (CROHME 2014),» de *2014 14th International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 791-796. |
| [6] | H. a. V.-G. C. a. Z. R. a. G. U. Mouchère, «ICFHR2016 CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions,» de *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 607-612. |
| [7] | M. a. Z. R. a. M. H. a. V.-G. C. a. G. U. Mahdavi, «ICDAR 2019 CROHME + TFD: Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection,» de *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 1533-1538. |
| [8] | «An integrated grammar-based approach for mathematical expression recognition,» *Pattern Recognition,* vol. 51, pp. 135-147, 2016. |
| [9] | A. a. H. J.-P. Belaid, «A Syntactic Approach for Handwritten Mathematical Formula Recognition,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vols. %1 de %2PAMI-6, nº 1, pp. 105-111, 1984. |
| [10] | A. a. R. G. Kosmala, «On-line handwritten formula recognition using statistical methods,» de *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, 1998, pp. 1306-1308 vol.2. |
| [11] | doi.org/10.1007/978-3-642-77281-8\_2, «An Experimental Implementation of a Document Recognition System for Papers Containing Mathematical Expressions,» de *Structured Document Image Analysis*, Springer Berlin Heidelberg, 1992, pp. 36-53. |
| [12] | J. a. H. R. a. P. I. Ha, «Understanding mathematical expressions from document images,» de *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, pp. 956-959. |
| [13] | S. a. N. K. a. A. J. Smithies, «A Handwriting-Based Equation Editor,» de *Proceedings of the 1999 Conference on Graphics Interface '99*, Morgan Kaufmann Publishers Inc., 1999, p. 84–91. |
| [14] | H. a. Z. R. a. G. U. a. V.-G. C. Mouchère, «Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011–2014,» *International Journal on Document Analysis and Recognition (IJDAR),* vol. 19, 2016. |
| [15] | F. a. H. N. S. a. V. C. a. M. H. a. M. S. JulcaAguilar, «Mathematical Symbol Hypothesis Recognition with Rejection Option,» de *2014 14th International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 500-505. |
| [16] | F. a. M. H. a. V.-G. C. a. H. N. S. T. Julca-Aguilar, «Top-Down Online Handwritten Mathematical Expression Parsing with Graph Grammar,» de *20th Iberoamerican Congress, CIARP 2015, Montevideo, Uruguay, November 9-12, 2015, Proceedings*, Montevideo, 2015, pp. 444-45. |
| [17] | J. Z. a. J. D. a. S. Z. a. D. L. a. Y. H. a. J. H. a. S. W. a. L. Dai, «Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition,» *Pattern Recognition,* pp. 196-206, 2017. |
| [18] | Y. F. Z. Y. Z. X. L. C. Wu JW., «Image-to-Markup Generation via Paired Adversarial Learning,» de *Machine Learning and Knowledge Discovery in Databases*, Cham, Springer International Publishing, 2019, pp. 18-34. |

# Planificación temporal y presupuesto

# Índice de figuras (opcional)

# Índice de tablas (opcional)

# Abreviaturas

CROHME: *Competition on Recognition of On-line Handwritten Mathematical Expressions* (Competición en reconocimiento *online* de expresiones matemáticas manuscritas).

LSTM: *Long-Short Term Memory* (Memoria de corto y largo plazo)

BLSTM: *Bidirectional Long-Short Term Memory* (Memoria de corto y largo plazo bidireccional)

GRU: *Gated Recurrent Unit* (Unidad recurrente cerrada)

RNN: *Recurrent Neural Network* (Red neuronal recurrente)

CNN: *Convolutional Neural Network* (Red neuronal convolucional)

SVM: *Support Vector Machine* (Máquina de vectores de soporte)

HMM: *Hidden Markov Models* (Modelos ocultos de Márkov)

KNN: *K-Nearest Neighbours* (K-vecinos cercanos)

1. https://demo.wiris.com/mathtype/en/index.php [↑](#footnote-ref-1)
2. https://github.com/falvaro/seshat [↑](#footnote-ref-2)
3. https://cat.prhlt.upv.es/mer [↑](#footnote-ref-3)
4. https://www.myscript.com/calculator [↑](#footnote-ref-4)
5. https://mathpix.com/ [↑](#footnote-ref-5)
6. https://www.v7labs.com/blog/convolutional-neural-networks-guide [↑](#footnote-ref-6)