



Universidad
Rey Juan Carlos

GRADO EN INGENIERÍA EN TELEMÁTICA

Curso Académico 2021/2022

Trabajo Fin de Grado

GAMIFICACIÓN DE PLATAFORMA UNIBOTICS

Autor : Daniel Hervás Rodao

Tutor : José María Cañas Plaza

Co-Tutor : David Roldán Álvarez

Trabajo Fin de Grado

Gamificación de la Plataforma Unibotics

Autor : Daniel Hervás Rodao

Tutor : José María Cañas Plaza **Co-Tutor :** David Roldán Álvarez

La defensa del presente Proyecto Fin de Carrera se realizó el día de
de 202X, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 202X

*Dedicado a
mi familia / mi abuelo / mi abuela*

Agradecimientos

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

1. Introducción	1
1.1. Robótica	1
1.2. Componentes robóticas	3
1.2.1. Middlewares robóticos	4
1.2.2. Simuladores robóticos	4
1.3. Robótica educativa	5
2. Objetivos	7
2.1. Objetivo general	7
2.2. Objetivos específicos	7
2.3. Planificación temporal	7
2.4. Control de versiones	7
3. Herramientas	9
3.1. Lenguaje JavaScript	9
3.2. Lenguaje HTML	10
3.3. Hojas de estilo CSS	11
4. Diseño e implementación	13
4.1. Arquitectura general	13
5. Experimentos y validación	15
6. Resultados	17

7. Conclusiones	19
7.1. Consecución de objetivos	19
7.2. Aplicación de lo aprendido	19
7.3. Lecciones aprendidas	20
7.4. Trabajos futuros	20
A. Manual de usuario	21
Bibliografía	23

Índice de figuras

1.1. Brazo robótico.	2
1.2. Coche autónomo.	2
1.3. Robot DaVinci.	3
1.4. Robot ATRIAS.	3
1.5. Robot militar.	3
1.6. Simulador Gazebo.	5
3.1. Documento básico de <i>HTML</i>	10
4.1. Estructura del parser básico.	14
4.2. Página con enlaces a hilos	14

Capítulo 1

Introducción

El TFG que será descrito a continuación se ha desarrollado en la plataforma *Unibotics* de la asociación *JdeRobot*¹, orientado al aprendizaje de robótica para estudiantes universitarios. El principal motivo de este proyecto es la introducción de técnicas de gamificación para los diferentes ejercicios contenidos la plataforma, así como añadir nuevos.

En este capítulo introductorio se introducirá el contexto en el que se desarrolla el trabajo, así como los motivos que han motivado a llevarlo a cabo.

El campo de la robótica es muy amplio, en concreto, este TFG se encuentra en el marco de la robótica educativa, destinada a la enzeñanza de la misma.

1.1. Robótica

Los avances en computación de las últimas décadas han sido el impulso que ha permitido la creación de máquinas muy cercanas al ideal de autonomía que se ha perseguido siempre. La robótica está muy relacionada no solo con la rama de la ingeniería, si no, que involucra conocimientos de matemáticas y física para el desarrollos de máquinas autónomas. Uno de los objetivos principales de la robótica es el de facilitar tareas de la vida diaria al ser humano, incluso, en algunas ocasiones, sustituir al ser humano.

La inteligencia artificial, también está muy ligada al campo de la robótica. Los avances en este campo permiten desarrollar sistemas capaces de tener una cierta memoria útil para realizar una serie de funciones.

¹<https://jderobot.github.io>

En 1950 la robótica experimenta un gran desarrollo. Esto se debe a los grandes avances en relación a la potencia y complejidad computacional. El acoplamiento mecánico empezó a sustituirse por sistemas eléctricos. Tal es el grado de desarrollo que se empiezan a generar sistemas de control automático consistentes en máquinas de estado secuencial.

Un claro ejemplo de este gran impluso es la implementación de robots en la industria automovilística, capaces de realizar tareas repetitivas, y que conllevan un gran riesgo para las personas (Figura 1.1).

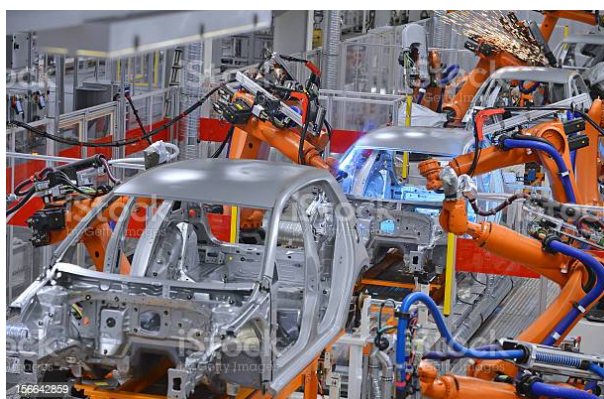


Figura 1.1: Brazo robótico.

En la actualidad, cada vez son más populares los coches autónomos. Es un campo muy amplio que cuenta con un gran número de posibilidades. Algunas de estas posibilidades serían los coches con conducción autónoma de *Tesla* (Figura 1.2) o los coches con aparcamiento autónomo que están desarrollando un gran número de compañías en la actualidad.

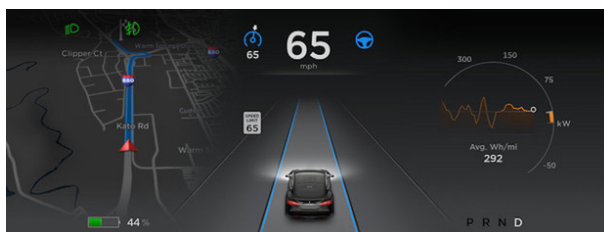


Figura 1.2: Coche autónomo.

Otro campo de la aplicación de la robótica es la medicina, donde existen robots capaces de filtrar las vibraciones naturales del humano para proporcionarle una gran precisión y seguridad, un claro ejemplo es el robot DaVinci (Figura 1.3). Adicionalmente, hay robots capaces de man-

tener una estabilidad la estabilidad necesaria para caminar sobre dos piernas robóticas, como es el robot ATRIAS (Figura 1.4).



Figura 1.3: Robot DaVinci.



Figura 1.4: Robot ATRIAS.

En el ámbito militar, existen robots capaces de sustituir a una persona en el a la hora de realizar tareas de gran peligro como la desactivación de bombas y la entrada en zonas contaminadas (Figura 1.5).



Figura 1.5: Robot militar.

1.2. Componentes robóticas

Todo robot está formado por dos componentes: el *software*, encargado de proporcionar la inteligencia al robot, el más importante, y, el *hardware* encargado de proporcionar la estructura física del robot.

Con al gran auge de la robótica han surgido numerosas plataformas que proporcionan herramientas que simplifican el desarrollo de software robótico, esto son los denominados *midd-*

middlewares robóticos.

Durante el desarrollo de software robótico es preciso realizar una serie de pruebas para comprobar el funcionamiento del código y depurar errores, por lo que se necesitan simuladores que nos proporcionen un entorno cercano a la realidad previa al ensamblado del robot.

1.2.1. Middlewares robóticos

Un *middleware* robótico es un *framework* que proporciona una serie de herramientas que facilitan el desarrollo de software para robots. Proporciona los servicios necesarios para soportar y simplificar aplicaciones complejas y distribuidas. Para el control de los sensores y actuadores de los robots, los *middlewares* proporcionan *drivers*, APIs, etc.

El *middleware* robótico más generalizado es ROS² (*Robotics Operating System*). *Robotics Operating System* fue desarrollado en 2007 por el Laboratorio de Inteligencia Artificial de Stanford para dar soporte a sus proyectos. A pesar de no ser un sistema operativo, ROS proporciona tales servicios como la abstracción *hardware*, mecanismos de comunicación entre procesos, el control de dispositivos de bajo nivel y el mantenimiento de paquetes. *Robotics Operating System* fue desarrollado para sistemas UNIX, aunque en la actualidad está siendo adaptado para su funcionamiento en sistemas operativos como Fedora, Mac OS X, Arch, Gentoo, OpenSUSE, Slackware, Debian o Microsoft Windows.

1.2.2. Simuladores robóticos

El surgimiento de estos *softwares* robóticos está condicionado por la necesidad de realizar pruebas durante el desarrollo del *software* para la detección y depuración de posibles errores antes de llevarlo a un robot real debido al gran coste que suponen.

El simulador más generalizado en la actualidad es *Gazebo*³. Su popularidad se debe a su robusto motor de físicas, sus gráficos de alta calidad y su amplio catálogo de robots y escenarios. Es una herramienta de código abierto integrada con ROS, por lo que permite ejecutar *software* robótico en un escenario simulado (Figura 1.6).

²<https://www.ros.org/>

³<http://gazebo-sim.org/>

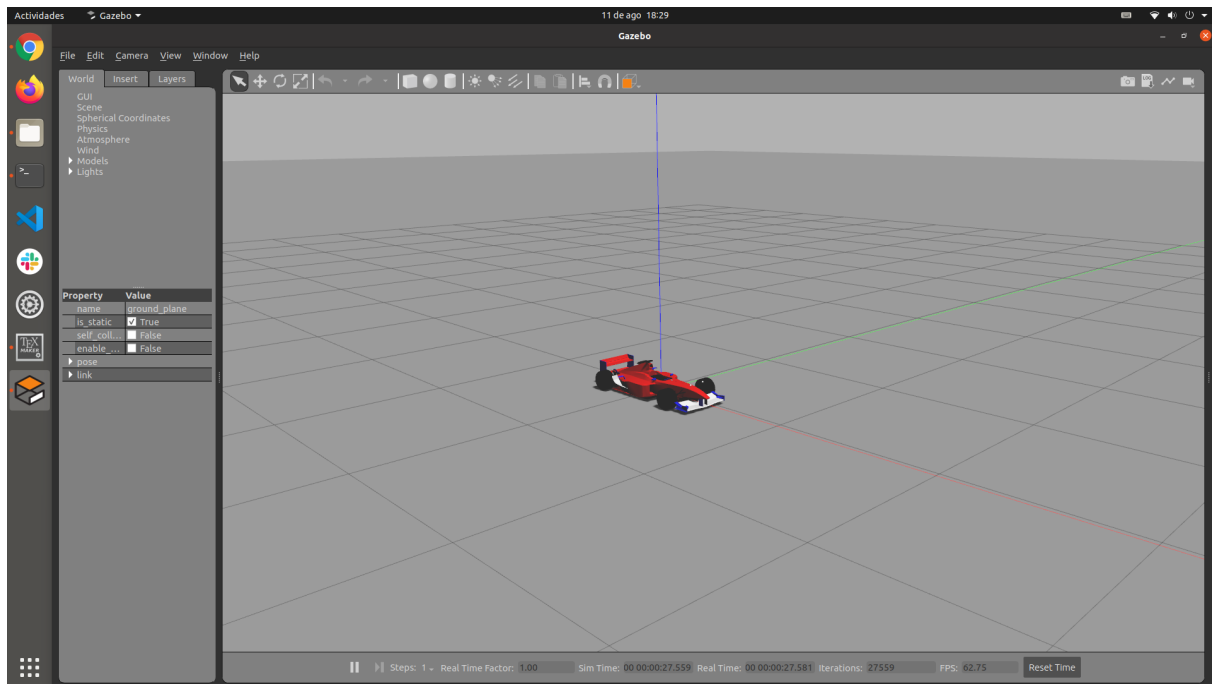


Figura 1.6: Simulador Gazebo.

1.3. Robótica educativa

La robótica educativa proporciona a los estudiantes la infraestructura para la construcción y programación de un robot, pero, además de la enseñanza robótica, estos entornos van más allá, ofreciendo la capacidad para el alumno de adquirir un pensamiento lógico. También, contribuye en la adquisición de una mentalidad resolutoria y al enriquecimiento de la cultura científica de los alumnos. Este método de educación con la robótica como objeto de enseñanza se denomina el método STEAM (Science, Technology, Engineering and Mathematics).

Como se ha comentado, para llevar a cabo este método de educación es necesaria una infraestructura, como puede ser LEGO Mindstorms, un kit de robótica que es capaz de aumentar la capacidad de pensamiento de los alumnos.

Dentro de la robótica educativa, es preciso enfatizar la plataforma *Unibotics* en la que se desarrolla el presente proyecto. Esta plataforma es un proyecto internacional que ofrece material para la enseñanza de robótica en las aulas. *Unibotics* proporciona una infraestructura software en conjunto a una colección de ejercicios, cada uno con el material teórico correspondiente para su resolución.

Capítulo 2

Objetivos

- 2.1. Objetivo general**
- 2.2. Objetivos específicos**
- 2.3. Planificación temporal**
- 2.4. Control de versiones**

Capítulo 3

Herramientas

En este capítulo se hará una breve presentación de todas las herramientas empleadas para el desarrollo del presente TFG. Estas tecnologías se pueden englobar en dos grupos, las tecnologías Front-End dedicadas a la presentación y a la interfaz de usuario (JavaScript, HTML y CSS), tecnologías Back-End dedicadas al servidor (Django), y, por último, tecnologías WebRTC que serán las encargadas de transmitir el vídeo en los juegos síncronos.

3.1. Lenguaje JavaScript

Se trata de un lenguaje de programación interpretado, su estándar es *ECMAScript*¹, basado en *Java* y *C*. Fue creado para aplicaciones web del lado del cliente. Es interpretado en el navegador web y permite mejoras en la interfaz de usuario, además de páginas web dinámicas. También puede usarse en el lado del servidor utilizando *Node.js*, un entorno de ejecución de JavaScript construido con el motor *JavaScript V8*.

En este proyecto se ha empleado JavaScript en el lado del cliente. La lógica de las plantillas que usan los ejercicios ha sido programada usando *ECMAScript-6*, así como los evaluadores automáticos de los mismos, y, adicionalmente, para los *WebSockets* encargados de comunicarse con el servidor para realizar tareas de señalización, o de envío de mensajes. Sus principales características son:

- Se trata de un lenguaje del lado del cliente, es decir, ejecuta en la máquina del propio

¹Especificación de lenguaje de programación que define un lenguaje de tipos dinámicos y soporta programación orientada a objetos basada en prototipos.

cliente a través de un navegador.

- Tipado débil, por lo que no es necesario especificar el tipo de dato al declarar una variable permitiendo que una misma variable pueda adquirir distintos tipos durante la ejecución.
- De alto nivel, con lo que significa que su sintaxis es fácilmente comprensible. Esta sintaxis se encuentra alejada del lenguaje máquina.
- Es un lenguaje interpretado puesto que utiliza un intérprete que traduce las líneas de código a lenguaje máquina en tiempo de ejecución.
- Es un lenguaje orientado a objetos, ya que utiliza clases y objetos como estructuras.

Adicionalmente, junto con *ES-6* se ha empleado una librería llamada *jQuery*² que permite agregar dinamismo a un sitio web permitiendo interactuar con los elementos HTML, manipular el DOM, manejar eventos, diseñar animaciones y agregar interacción con la técnica *AJAX*³ (*Asynchronous JavaScript and XML*).

3.2. Lenguaje HTML

HTML (*HyperText Markup Language*) es un lenguaje de marcado que permite indicar la estructura un documento utilizando etiquetas. Es utilizado para la creación de documentos electrónicos que se envían a través de la red. Los documentos pueden tener conexiones con otros a través de *hipervínculos*.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hola mundo!</title>
  </head>
  <body>
    <p>Hola mundo!</p>
  </body>
</html>
```

Figura 3.1: Documento básico de *HTML*.

²<https://jquery.com/>

³<https://developer.mozilla.org/es/docs/Web/Guide/AJAX>

Primeramente se debe declarar el tipo del documento *HTML* mediante la línea *DOCTYPE html* que indica que es un documento *HTML-5*. El elemento *html* engloba el documento *HTML*, dentro de este elemento se encuentran dos etiquetas:

- *HEAD* es la cabecera del documento, que contiene la información general (metadatos) acerca del documento, incluyendo el título y los enlaces a scripts y hojas de estilos.
- *BODY* es el cuerpo del documento *HTML* donde se encuentran las etiquetas que dan formato al mismo. Puede contener imágenes, enlaces, vídeos, menús, formularios, botones, además de animaciones, que se pueden crear dentro de un elemento *canvas*.

Con la incorporación de *HTML-5* se han introducido diversas novedades y mejoras que son de interés para este trabajo:

- *WebSockets*, es una tecnología que hace posible abrir una comunicación entre el navegador y el servidor.
- *WebRTC* (*Web Real-Time Communications*) es una tecnología que permite a las aplicaciones web capturar y transmitir audio y vídeo entre navegadores sin necesidad de un intermediario.
- Se añade un mejor soporte de contenido multimedia sin la necesidad de instalar plugins adicionales. Mediante el elemento *video* la página reproducirá de manera nativa el contenido.
- Proporciona un elemento, *canvas* que permite renderizar escenas gráficas en la web mediante el uso de *JavaScript*.

3.3. Hojas de estilo CSS

Capítulo 4

Diseño e implementación

Aquí viene todo lo que has hecho tú (tecnológicamente). Puedes entrar hasta el detalle. Es la parte más importante de la memoria, porque describe lo que has hecho tú. Eso sí, normalmente aconsejo no poner código, sino diagramas.

4.1. Arquitectura general

Si tu proyecto es un software, siempre es bueno poner la arquitectura (que es cómo se estructura tu programa a “vista de pájaro”).

Por ejemplo, puedes verlo en la figura 4.1. \LaTeX pone las figuras donde mejor cuadran. Y eso quiere decir que quizás no lo haga donde lo hemos puesto... Eso no es malo. A veces queda un poco raro, pero es la filosofía de \LaTeX : tú al contenido, que yo me encargo de la maquetación.

Recuerda que toda figura que añadas a tu memoria debe ser explicada. Sí, aunque te parezca evidente lo que se ve en la figura 4.1, la figura en sí solamente es un apoyo a tu texto. Así que explica lo que se ve en la figura, haciendo referencia a la misma tal y como ves aquí. Por ejemplo: En la figura 4.1 se puede ver que la estructura del *parser* básico, que consta de seis componentes diferentes: los datos se obtienen de la red, y según el tipo de dato, se pasará a un *parser* específico y bla, bla, bla. . .

Si utilizas una base de datos, no te olvides de incluir también un diagrama de entidad-relación.

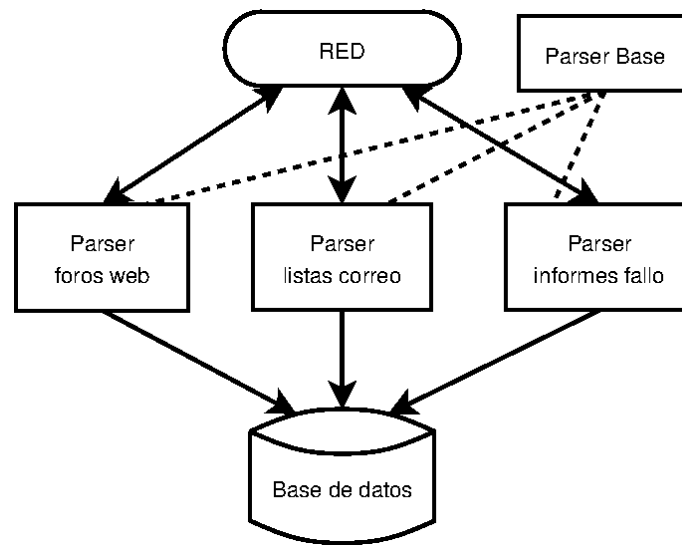


Figura 4.1: Estructura del parser básico.

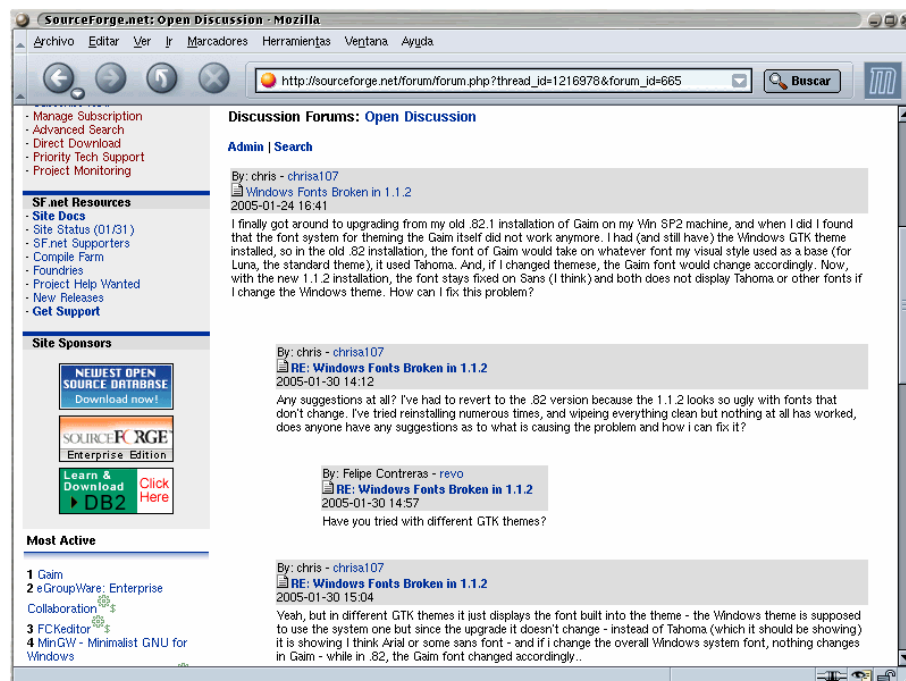


Figura 4.2: Página con enlaces a hilos

Capítulo 5

Experimentos y validación

Este capítulo se introdujo como requisito en 2019. Describe los experimentos y casos de test que tuviste que implementar para validar tus resultados. Incluye también los resultados de validación que permiten afirmar que tus resultados son correctos.

Capítulo 6

Resultados

En este capítulo se incluyen los resultados de tu trabajo fin de grado.

Si es una herramienta de análisis lo que has realizado, aquí puedes poner ejemplos de haberla utilizado para que se vea su utilidad.

Capítulo 7

Conclusiones

7.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

Y si has llegado hasta aquí, siempre es bueno pasarle el corrector ortográfico, que las erratas quedan fatal en la memoria final. Para eso, en Linux tenemos *aspell*, que se ejecuta de la siguiente manera desde la línea de *shell*:

```
aspell --lang=es_ES -c memoria.tex
```

7.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a

2. b

7.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. Aquí viene uno.
2. Aquí viene otro.

7.4. Trabajos futuros

Ningún proyecto ni software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFM.

Apéndice A

Manual de usuario

Esto es un apéndice. Si has creado una aplicación, siempre viene bien tener un manual de usuario. Pues ponlo aquí.

Bibliografía

- [1] HISTORIA DE LA ROBÓTICA. <https://scielo.isciii.es/pdf/aue/v31n3/v31n3a02.pdf>
- [2]
- [3]
- [4]
- [5]