



ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA INFORMÁTICA

**Asistente para el diseño de la interfaz para control,
seguimiento y sharing de robots en tiempo real**

Manuel López Urbina
Director: Arturo Morgado Estévez

Cádiz, 6 de mayo de 2017



ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA INFORMÁTICA

RobotUI:
Interfaz de Usuario de propósito general

- Departamento: Ingeniería en Automática, Electrónica, Arquitectura y Redes de Computadores
- Director del proyecto: Arturo Morgado Estévez
- Autor del proyecto: Manuel López Urbina

Cádiz, 6 de mayo de 2017

Fdo: Manuel López Urbina

0.1. Agradecimientos

Este proyecto significa la culminación de mi carrera, por lo que me gustaría dedicárselo a todas las personas que me han ayudado a conseguir acabarla.

En primer lugar me gustaría agradecerle, de igual modo que en el PFC de la Inegniería Técnica, a mi familia el apoyarme y ayudarme durante estos años, y el esfuerzo que han hecho para que yo haya podido culminar mi ingeniería.

Mención especial para Natalia Luciano, mi pareja, la cual ha estado siempre apoyándome en mis objetivos y tanta paciencia y comprensión me ha mostrado tras tantos días, meses e incluso años de estudio y dedicación.

Agradecimientos a D. Arturo Morgado por su ayuda y dedicación durante la realización y dirección de este proyecto, así como su carácter amable y servicial que han hecho más ameno el trabajo realizado.

También me gustaría agradecérselo a mis compañeros, con los que tantos ratos inolvidables he pasado, y que tanto me han ayudado.

Por último quiero dedicarle este proyecto a todos los estudiantes de informática, en especial a todos los amantes del fascinante mundo de la robótica, a los que espero que mi trabajo les sea de utilidad.

0.2. Licencia

Este documento ha sido liberado bajo Licencia GFDL 1.3 (GNU Free Documentation License). Se incluyen los términos de la licencia en inglés al final del mismo.

Copyright © 2017 Manuel López Urbina.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Resumen

RobotUI es un proyecto web para el control y seguimiento de dispositivos robóticos y compartirlos con otros usuarios.

Palabras clave: Internet, aplicación web, robótica, robots, interfaz de usuario, streaming de vídeo, control remoto, robot sharing, Sails.js, Node.js, Socket.io, Websockets, compartir, asistente, tiempo real.

Índice general

0.1. Agradecimientos	I
0.2. Licencia	III
Índice general	I
Índice de figuras	v
1. Introducción	1
1.1. Introducción y antecedentes	1
1.2. Objetivos	4
1.3. Acerca de este documento	5
2. Conceptos básicos	7
2.1. Transmisión y comunicación	7
2.2. Socket	7
2.3. WebSocket	8
2.4. Streaming	9
2.5. Framework	9
3. Herramientas utilizadas	11
3.1. Tecnologías software utilizadas	12
3.1.1. L ^A T _E X	12
3.1.2. WebStorm	12
3.1.3. Github	12
3.1.4. Git	13
3.1.5. DigitalOcean	13
3.1.6. Node js	14
3.1.7. Sails js	14
3.1.8. Npm	15
3.1.9. SocketIO	15
3.1.10. FFmpeg	16
3.1.11. Bootstrap	17
3.1.12. JQuery	17
3.1.13. Mongo DB	17
3.1.14. Pm2	18
3.1.15. Robomongo	18
3.2. Tecnologías hardware y materiales utilizados	19
3.2.1. Raspberry Pi Model B	19

3.2.2. Controladora de motores L298N	20
3.2.3. Batería LiPo	20
3.2.4. Tarjeta de expansión con batería de Litio para Raspberry Pi	21
3.2.5. Cámara USB de alta definición	21
4. Desarrollo software	23
4.1. Metodología de desarrollo	23
4.2. Recolección de requisitos	24
4.2.1. Requisitos funcionales	24
4.2.2. Requisitos no funcionales	25
4.3. Diagramas de casos de uso	26
4.4. Especificación de los casos de uso	28
5. Comunicaciones	37
5.0.1. Fundamentos	37
5.0.2. Comunicaciones en RobotUI	38
6. Robot de pruebas	45
6.1. Análisis	45
6.1.1. Análisis de requerimientos hardware	46
6.2. Análisis de requerimientos de Software	46
6.3. Montaje	47
6.3.1. Interconexión de elementos	47
6.4. Software de control	48
7. Organización temporal	55
7.1. Planificación temporal de tareas	58
7.1.1. Hito 1: Planificación y análisis	58
7.1.2. Hito 2: Definición de requisitos	58
7.1.3. Hito 3: Comienzo de desarrollo de la aplicación	59
7.1.4. Hito 4: Desarrollo de la aplicación, módulo componentes	59
7.1.5. Hito 5: Desarrollo de la aplicación, módulo interfaz	59
7.1.6. Hito 6: Desarrollo del módulo de comunicaciones	60
7.1.7. Hito 7: Construcción del vehículo de pruebas	60
7.1.8. Hito 8: Programación del vehículo de pruebas	60
7.1.9. Hito 9: Documentación	60
7.2. Diagrama de Gantt	60
8. Guía de Usuario	63
8.0.1. Objetivo de esta guía	63
8.0.2. Dirigido a	64
8.0.3. Obtener RobotUI	64
8.1. Uso de RobotUI	64
8.1.1. Registro de usuario	65
8.1.2. Creación de un Robot	65
8.1.3. Configuración de la interfaz	66
8.2. Programa tu robot	68

9. Comentarios finales	69
9.1. Presupuesto	69
9.2. Conclusiones	70
9.3. Mejoras futuras	70
Anexos	73
.1. Instalación de Node.js	73
.2. Instalación Sails.js	73
.2.1. Prerrequisitos	73
.2.2. Instalación	74
.3. Instalación de MongoDB	75
.3.1. Prerrequisitos	75
.3.2. Instalación	75
.4. Instalación del control de versiones Git	77
.4.1. Configuración	77
.5. Despliegue de una aplicación Sails	77
Bibliografía	81
GNU Documentation Free License	83

Índice de figuras

1.1. Logo RobotUI ¹	3
1.2. Página principal de de RobotUI.	3
1.3. Imagen del vehículo de pruebas desarrollado. ²	4
2.1. Representación de los sockets como una interfaz de la capa de transporte del protocolo TCP/IP.	8
3.1. Droplet desplegado en DigitalOcean	14
3.2. Imagen de una Raspberry Pi 3 Model B	19
3.3. Imagen de la controladora de motores L298n utilizada	20
3.4. Imagen de la batería LiPo utilizada.	21
3.5. Imagen de la tarjeta de expansión con batería de Litio utilizado.	21
3.6. Imagen de la cámara USB utilizada.	22
4.1. Subsistemas existentes en el proyecto junto con el modelo de ciclo de vida utilizado para su desarrollo.	24
4.2. Diagrama de casos de uso para la interacción con la aplicación.	27
4.3. Diagrama de casos de uso para la interacción con el robot de pruebas.	28
5.1. Página de gestión de usuarios actualizable en tiempo real.	38
5.2. Esquema representativo del flujo de conexiones de RobotUI.	38
6.1. Imagen del robot de pruebas.	45
6.2. Esquema GPIO de una Raspberry Pi Model B+.	47
6.3. Esquema de conexiones del robot de pruebas.	48
7.1. Panel de actividades - Trello	55
7.2. Descomposición de las tareas implicadas en el desarrollo del proyecto (Primera Parte).	57
7.3. Descomposición de las tareas implicadas en el desarrollo del proyecto (Segunda parte).	58
7.4. Diagrama de Gantt 1. Desarrollo del proyecto.	61
7.5. Diagrama de Gantt 2. Desarrollo del proyecto.	62
8.1. Página principal RobotUI.	64
8.2. Página principal RobotUI.	64
8.3. Página principal RobotUI.	65
8.4. Formulario de creación de un robot.	66

8.5. Panel de configuración de la interfaz.	67
8.6. Panel de herramientas y panel de acciones para la configuración de la interfaz.	67
1. Iniciando Sails ³	74
2. Sails en funcionamiento ⁴	75
3. Utilización del gestor de procesos Pm2 en el entorno de producción.	79

Capítulo 1

Introducción

*Lo mejor que podemos hacer por otro
no es sólo compartir con él nuestras riquezas,
sino mostrarle las suyas*
Benjamin Disraeli

1.1. Introducción y antecedentes

La robótica es una rama de la ingeniería, la cual se ocupa del diseño, construcción, operación y uso de robots¹, así como sistemas informáticos para su control, retroalimentación sensorial y procesamiento de información. Entre las diversas disciplinas aplicadas a la robótica podemos encontrar: la mecánica, la electrónica, la informática, la inteligencia artificial, la ingeniería de control y la física, entre otras muchas, de lo cual podemos considerar la robótica como una ciencia multidisciplinaria.

En la actualidad, los robots comerciales e industriales son ampliamente utilizados y cada día realizan tareas de forma más exacta o más barata que los humanos. También se les utiliza en trabajos demasiado sucios, peligrosos o tediosos. Los robots son muy utilizados en plantas de fabricación, montaje y embalaje, en transporte, en exploraciones en la Tierra y en el espacio, cirugía, armamento, investigación en laboratorios y en la producción en masa de bienes industriales o de consumo. Otras aplicaciones incluyen la limpieza de residuos tóxicos, minería, búsqueda y rescate de personas y localización de minas terrestres. En definitiva, la robótica está presente en prácticamente cualquier ámbito que podamos imaginar en la actualidad.

Por otra parte, ninguno de los sistemas robóticos actuales podrían ser funcionales sin un software adecuado para su manejo y control, en ocasiones siendo éste tremadamente complejo y específico para garantizar una correcta sincronización entre los diferentes elementos hardware y software implicados con la finalidad de garantizar un correcto

¹Robot: Máquina automática programable capaz de realizar determinadas operaciones de manera autónoma y sustituir a los seres humanos en algunas tareas, en especial las pesadas, repetitivas o peligrosas; puede estar dotada de sensores, que le permiten adaptarse a nuevas situaciones.

funcionamiento del conjunto robótico.

Por estas razones cada vez son más las escuelas que hacen uso de la robótica para que los estudiantes se interesen en la tecnología ya que pueden encontrar un entorno divertido donde aprender, y que ofrece multitud de ventajas:

1. Los niños lo encuentran divertido Hay varios concursos orientados a distintos grupos de edad que pueden canalizar la competencia de una manera positiva. Por ejemplo, se le puede pedir a los niños que construyan un robot y luego hacer competiciones.
2. Es una manera eficaz de enseñarles programación a los estudiantes La programación puede ser muy abstracta. Al tener que controlar un robot físico y ver lo que sale mal, los estudiantes aprenden lo que los robots pueden y no pueden hacer. También aprenden la necesidad de dar instrucciones precisas.
3. Desarrolla habilidades útiles Capacidad de resolución de problemas, trabajo en equipo, capacidad de análisis, y un largo etcétera.

De lo anterior se extrae la necesidad de elaborar un sistema que, además de acercar la robótica a los estudiantes, permita compartir las creaciones con otros usuarios en internet. Todos hemos visto alguna vez vídeos en las redes sociales donde los usuarios nos muestran sus dispositivos en funcionamiento donde, en ocasiones, nos gustaría poder tomar control sobre ellos o visualizar su manejo en tiempo real.

Por tanto el sistema resultante debe cubrir dos necesidades principales, la primera, dotar al usuario de las herramientas necesarias para permitir la configuración de una interfaz de control de sus dispositivos sin necesidad de amplios conocimientos de programación, y la segunda, cubrir la necesidad paralela en la que los usuarios, orgullosos de sus creaciones, dispongan de una manera de compartir sus robots con el resto del mundo de una manera más dinámica. Es decir, en la que otros usuarios, a modo de espectadores, puedan visualizar el control de los dispositivos por parte de su creador, como si de una sesión de vídeo en streaming se tratara. También se dotará de la posibilidad de permitir el control por otros usuarios externos. En la actualidad no existe ningún medio eficaz donde hacer una difusión de la robótica de una manera similar.

Dada la problemática actual presentada, junto con que la programación web y la robótica son temas que causan en mi un especial interés, hicieron que me lanzara a la elaboración de este proyecto que unifica ambos campos anteriormente citados.

Así surgió *RobotUI* y con él un nuevo concepto llamado *RobotSharing*.

Figura 1.1: Logo RobotUI ².

RobotUI (nombre del sistema resultante) será una combinación de un elemento software (aplicación web) y hardware (vehículo de pruebas y demostración) surgido como muestra de la solución obtenida a los citados problemas.

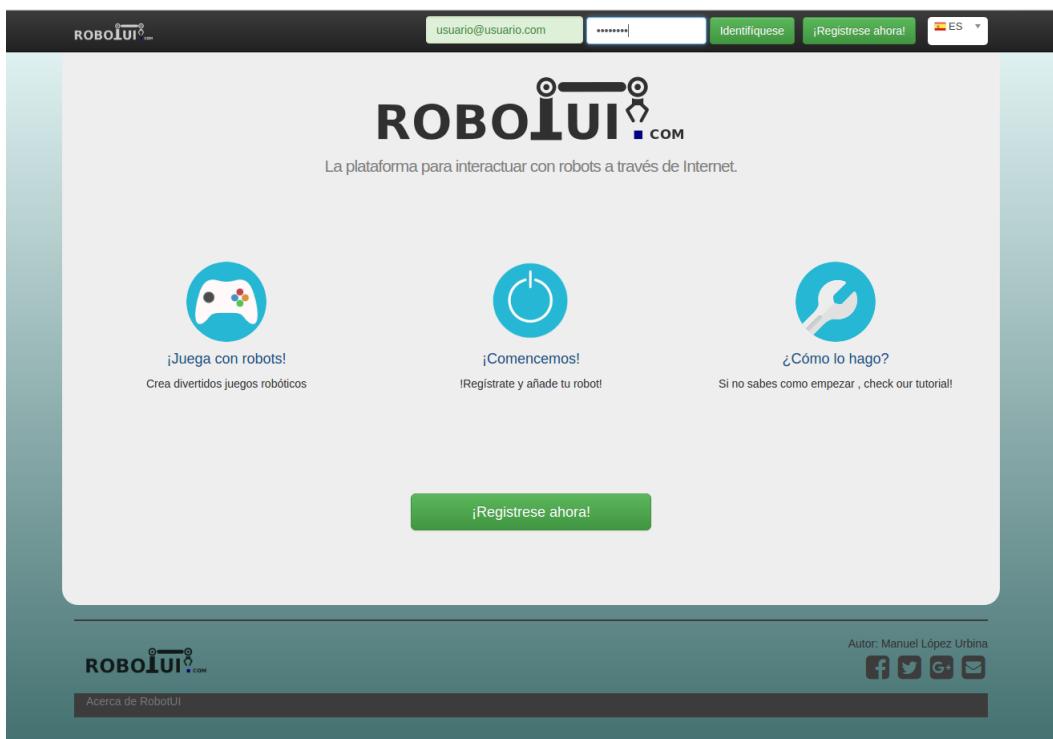


Figura 1.2: Página principal de RobotUI.

El elemento hardware de este proyecto se compone de un vehículo controlado vía WiFi el cual responde a una serie de señales, *comandos*³, a los que responde realizando determinadas acciones.

La interfaz web se configurará de tal manera que permita el control del susodicho vehículo a modo demostrativo. Este procedimiento servirá de guía para que cualquier usuario pueda proceder a dar de alta sus dispositivos robóticos para su control y difusión con otros usuarios.

²Logotipo RobotUI.

³ Comando: instrucción u orden que el usuario proporciona a un sistema informático, desde la línea de comandos (como una shell) o desde una llamada de programación.

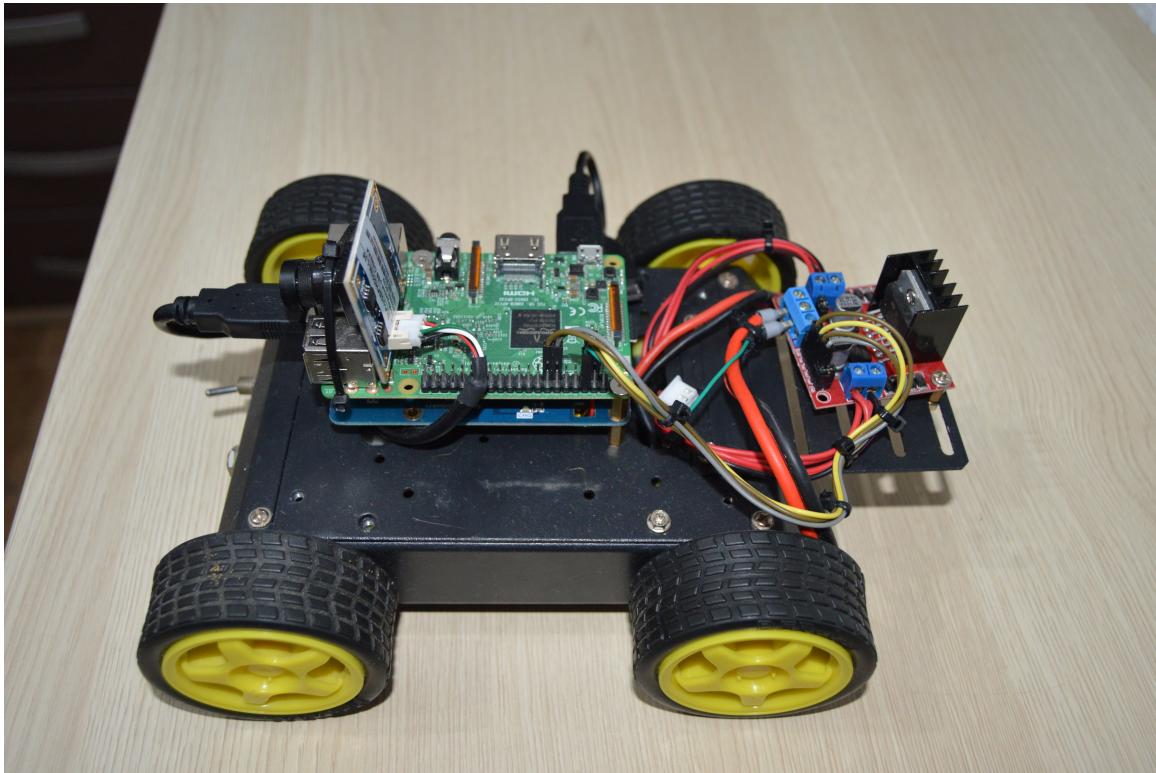


Figura 1.3: Imagen del vehículo de pruebas desarrollado.⁴

1.2. Objetivos

Como hemos visto, se requiere de multitud de conocimientos a la hora de afrontar un proyecto robótico con ciertas garantías. Este proyecto trata, al menos, de reducir, o facilitar, el área relacionada con la informática, más concretamente con la programación donde multitud de personas ven un impedimento a la hora de comenzar a desarrollar sus ideas.

Por otro lado, existe la imperiosa necesidad de que los usuarios quieran mostrar sus creaciones al resto del mundo, compartir experiencias, problemas, opiniones, etc, de una forma directa y no mediante la grabación de vídeos del funcionamiento de los proyectos robóticos en cuestión, ya que no disponen de una herramienta adecuada para ello.

De lo anterior deducimos que existe la necesidad de que otros usuarios puedan participar de manera más activa, ya sea visualizando el control del robot por parte de su creador o permitir que otros usuarios tomen el control de esos proyectos en tiempo real.

⁴Vehículo desarrollado para probar el funcionamiento de RobotUI. Su desarrollo y características quedan descritas en el capítulo 6.

El sistema a desarrollar dispondrá de dos modos de funcionamiento, el primero de ellos proporciona las herramientas para la configuración de una interfaz a gusto del usuario, en la cual, una vez configurada, el usuario podrá controlar a su antojo el dispositivo. En el segundo modo de funcionamiento, la aplicación permitirá que otros usuarios puedan visitar la interfaz anteriormente configurada y actuar como espectadores en el control del robot por el usuario propietario del mismo. En definitiva, se proporcionará un sistema de control y difusión en uno solo.

En definitiva, este proyecto busca facilitar la árdua labor de programación de los proyectos robóticos junto con la posibilidad de compartir las creaciones realizadas con otros usuarios.

1.3. Acerca de este documento

El documento se ha sido elaborado en un lenguaje claro y sencillo para permitir que un estudiante universitario de Ingeniería Informática pueda comprender los contenidos sin apenas dificultad añadida.

Este documento se organiza en los siguientes capítulos:

- En el capítulo 1, Introducción, se comentan las razones que han motivado la creación de este proyecto, así como el propósito del mismo.
- En el capítulo 2, Conceptos básicos, se incluyen definiciones de aquellos conceptos considerados de interés para la correcta comprensión del contenido de la presente memoria.
- En el capítulo 3, Estado del arte y herramientas utilizadas, se realiza una descripción de las diferentes elementos hardware y software empleados durante el desarrollo del proyecto y necesarios para la utilización del mismo. Así como una breve descripción del conocimiento acumulado y tecnologías existentes hasta la fecha.
- En el capítulo ??, Especificación y análisis de requisitos, se realiza una descripción de las diferentes elementos hardware y software empleados durante el desarrollo del proyecto y necesarios para la utilización del mismo. Así como una breve descripción del conocimiento acumulado y tecnologías existentes hasta la fecha.
- En el capítulo 7, Organización temporal, se recoge todo lo que concierne a la distribución y duración de cada una de las tareas llevadas a cabo durante el desarrollo del proyecto que el presente documento describe.
- En el capítulo ??, Configuración y montaje de los dispositivos hardware, se explica el proceso seguido para la correcta integración de los dispositivos hardware empleados describiendo la interconexión entre ellos así como su configuración.

- En el capítulo 4, Desarrollo software, se realiza un análisis sobre la metodología empleada para el desarrollo software, describiendo los modelos de ciclo de vida utilizados, la descripción de los requisitos funcionales junto con el diagrama de casos de uso.
- En el capítulo ??, Software de reconocimiento, se hace una descripción explicando los diferentes aspectos y elementos de cada uno de los prototipos desarrollados junto con los problemas encontrados y soluciones adoptadas.
- En el capítulo ??, Software de control, se describe cómo se ha llevado a cabo la comunicación ordenador-vehículo a nivel software.
- En el capítulo ??, Interfaz gráfica, se recogen aquellos aspectos técnicos de interés referentes a la elaboración de la interfaz gráfica.
- En el capítulo 8, Guía de usuario, se describen los diferentes aspectos necesarios para la correcta utilización del conjunto software y hardware de los que se compone el presente proyecto.
- En el capítulo 9, Conclusiones, se hace mención de las conclusiones obtenidas tras la realización del proyecto además de las posibles mejoras aplicables.
- En el capítulo Anexos 9.3, aparecen los manuales de instalación del software que ha sido necesario para la realización del proyecto.

Capítulo 2

Conceptos básicos

En el presente capítulo se recogen aquellos conceptos, definiciones y protocolos que resultan de especial interés y que ayudarán a la comprensión de los diferentes puntos tratados en el resto de la memoria sin profundizar demasiado en detalles técnicos. Todos estos conceptos se encuentran estrechamente ligados con tecnologías de la comunicación y transmisión de información, más concretamente en el ámbito de la programación web.

2.1. Transmisión y comunicación

Se denomina *transmisión* como el proceso de transporte de una señal de un lugar a otro y *comunicación* como el intercambio entre dos entes mediante una transmisión, los cuales son capaces de interpretar la información circundante entre ellos y en el cual existen un conjunto de reglas definidas, los protocolos¹, que rigen el proceso.

2.2. Socket

Socket designa un concepto abstracto mediante el cual dos programas, generalmente situados en computadoras distintas, pueden intercambiar cualquier flujo de datos de manera fiable y ordenada.

El término *socket* es también usado como el nombre de una interfaz de programación de aplicaciones (API) para la familia de protocolos de red TCP/IP², provista usualmente por el sistema operativo.

Los sockets constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números

¹Protocolo: reglamento o una serie de instrucciones que se fijan por tradición o por convenio.

²TCP/IP es un conjunto de protocolos que permiten la comunicación entre los ordenadores pertenecientes a una red. La sigla TCP/IP significa Protocolo de control de transmisión/Protocolo de Internet. Proviene de los nombres de dos protocolos importantes incluidos en el conjunto TCP/IP, es decir, del protocolo TCP y del protocolo IP.

de puerto local y remoto.

Cuando se habla de dirección y puerto local/remoto, se sobreentiende que nos referimos a dos procesos (cliente/servidor o nodo/nodo) ya que ambas direcciones IP y puerto pueden coincidir para el intercambio de información entre procesos dentro de una misma máquina y; además, la comunicación puede ser perfectamente bidireccional, asumiendo que el par que la inicia es el cliente y su contrapartida un servidor pero pudiendo ejercer de forma ambivalente ambas partes.

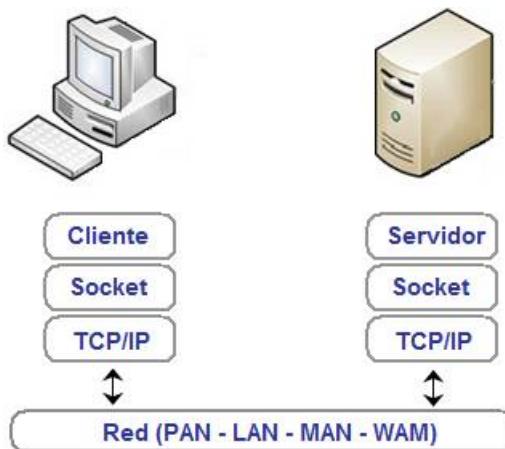


Figura 2.1: Representación de los sockets como una interfaz de la capa de transporte del protocolo TCP/IP.

2.3. WebSocket

Comprendido previamente el concepto de *socket* descrito en el punto 2.2, definimos *websocket* como una tecnología que proporciona un canal de comunicación bidireccional y full-duplex³ utilizada por cualquier aplicación cliente/servidor.

La API de WebSocket está siendo normalizada por el W3C, mientras que el protocolo WebSocket ya fue normalizado por la IETF⁴ como el RFC 6455.

Debido a que las conexiones TCP comunes sobre puertos diferentes al 80 son habitualmente bloqueadas por los administradores de redes, el uso de esta tecnología proporcionaría una solución a este tipo de limitaciones proveyendo una funcionalidad similar a la apertura de varias conexiones en distintos puertos, pero multiplexando diferentes

³Full Duplex: definido como la capacidad de transmisión y recepción en ambas direcciones al mismo tiempo.

⁴Internet Engineering Task Force (IETF) (en español, Grupo de Trabajo de Ingeniería de Internet) es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad. Se creó en los Estados Unidos, en 1986. Es mundialmente conocido porque se trata de la entidad que regula las propuestas y los estándares de Internet, conocidos como RFC.

servicios WebSocket sobre un único puerto TCP a costa de una pequeña sobrecarga del protocolo.

2.4. Streaming

La retransmisión (en inglés streaming, también denominado transmisión) es la distribución digital de contenido multimedia a través de una red de computadoras, de manera que el usuario utiliza el producto a la vez que se es descargado. La palabra retransmisión se refiere a una corriente continua que fluye sin interrupción, habitualmente audio o vídeo, aplicándose la difusión de vídeo en el presente proyecto.

Este tipo de tecnología funciona mediante un búfer de datos que va almacenando el flujo de descarga en la estación del usuario para inmediatamente mostrarle el material descargado. Esto se contrapone al mecanismo de descarga de archivos, que requiere que el usuario descargue los archivos por completo para poder acceder al contenido.

La retransmisión requiere de una conexión por lo menos de igual ancho de banda que la tasa de transmisión del servicio. La retransmisión de vídeo por Internet se popularizó a fines de la década de 2000, cuando la contratación del suficiente ancho de banda para utilizar estos servicios en el hogar se hizo lo suficientemente barato.

2.5. Framework

Un framework, también denominado entorno de trabajo o marco de trabajo, es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Aplicado a informática, concretamente al desarrollo de software, un entorno de trabajo o framework es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

En general, con el término framework, nos estamos refiriendo a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un framework son:

- Acelerar el proceso de desarrollo

- Reutilización de código.
- Promover buenas prácticas de desarrollo como el uso de patrones.

Capítulo 3

Tecnologías utilizadas

3.1. Tecnologías software utilizadas

A continuación se detallan las diferentes tecnologías/bibliotecas/lenguajes que se han empleado para la elaboración del proyecto y por qué se han escogido por encima de otras posibles soluciones.

3.1.1. L^AT_EX

Web: <https://www.latex-project.org/>

L^AT_EX es un lenguaje de marcado que sirve para la redacción de documentos científicos o técnicos. Con esta herramienta o lenguaje se ha desarrollado la memoria actual del proyecto de final de carrera.

3.1.2. WebStorm



Web: <https://www.jetbrains.com/webstorm/>

WebStorm es un IDE de JavaScript ligero pero potente, perfectamente equipado para el desarrollo del lado del cliente y el desarrollo del servidor con Node.js. Permite la integración con frameworks de desarrollo como Sails.js. Para el desarrollo de la aplicación se optó por este IDE.

3.1.3. Github



Web: <https://about.github.com/>

Repositorio: <https://github.com/lopi87/SAILS-RobotUI>

GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el framework Ruby on Rails por GitHub, Inc. (anteriormente conocida como Logical Awesome). Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

3.1.4. Git



Web: <https://git-scm.com/>

Git es un sistema open-source de control de versiones diseñado para manejar integralmente las fases de desarrollo de proyectos, simples y complejos, con velocidad y eficiencia.

3.1.5. DigitalOcean



Web: <https://www.digitalocean.com/>

Servidor web para alojar proyectos en la nube. La ventaja de este servicio de VPS¹ es que te permite desplegar máquinas de cualquier tipo (siempre que sean software libre) de una manera muy fácil y rápida. Además tiene un punto fuerte y es que la información se almacena en discos SSD, con lo que el procesamiento se ve muy mejorado a la hora de computar (en este caso trabajo con websockets y transmisión de datos).

¹ VPS: Servidor Virtual Privado, del inglés Virtual Private Server, es un método de particionar un servidor físico en varios servidores de tal forma que todo funcione como si se estuviese ejecutando en una única máquina. Cada servidor virtual es capaz de funcionar bajo su propio sistema operativo y además cada servidor puede ser reiniciado de forma independiente.

The screenshot shows the DigitalOcean control panel. At the top, there are navigation links: 'Droplets', 'Images', 'Networking', 'API', and 'Support'. On the right, there's a green 'Create Droplet' button and a user profile icon. Below the header, the word 'Droplets' is prominently displayed next to a search bar with the placeholder 'Search by Droplet name'. Underneath, there are two tabs: 'Droplets' (which is selected) and 'Volumes'. A table lists the currently deployed droplet:

Name	IP Address	Created	Tags
robotui 512 MB / 20 GB Disk / FRA1 - Ubuntu 16.04.1 x64	46.101.102.33	19 days ago	More

Figura 3.1: Droplet desplegado en DigitalOcean

3.1.6. Node js



Web: <https://nodejs.org/es/>

Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. Incorpora un sistema de gestión de paquetes llamado, npm, es el ecosistema mas grande de librerías de código abierto en el mundo.

Node.js tiene una arquitectura basada en eventos capaz de E/S asíncronos. Estas opciones de diseño apuntan a optimizar el rendimiento y la escalabilidad en aplicaciones Web con muchas operaciones de entrada/salida, así como para aplicaciones Web en tiempo real (por ejemplo, programas de comunicación en tiempo real y juegos de navegador), lo que lo hacen ideal para este proyecto.

3.1.7. Sails js



Web: <http://sailsjs.com/>

Sails.js es un framework web que facilita la creación de aplicaciones personalizadas Node.js de nivel empresarial. Está diseñado para parecerse a la arquitectura MVC de frameworks como Ruby on Rails, pero con soporte para el estilo de desarrollo de aplicaciones web más moderno y orientado a datos.

Utiliza Express para funciones como la gestión de peticiones HTTP y websockets. Su envoltorio intuitivo de websockets lo hace especialmente bueno para construir características en tiempo real como por ejemplo un Chat, por estas razones se ha considerado como el framework más adecuado hasta la fecha para la elaboración de este proyecto.

3.1.8. Npm



Web: <https://www.npmjs.com/>

npm es el manejador de paquetes por defecto para Node.js, un entorno de ejecución para JavaScript. Utilizado para la descarga de las librerías incorporadas al proyecto.

3.1.9. SocketIO



Web: <https://socket.io/>

Socket.io es una librería que nos permite manejar eventos en tiempo real mediante una conexión TCP y todo ello en JavaScript desde un cliente. Es realmente potente y podemos hacer todo tipo de aplicaciones en tiempo real.

Socket.IO es una biblioteca de JavaScript para aplicaciones web en tiempo real. Permite la comunicación bidireccional en tiempo real entre clientes web y servidores. Consta de dos partes: una biblioteca del lado del cliente que se ejecuta en el navegador y una biblioteca del lado del servidor para Node.js. Ambos componentes tienen una API casi idéntica. Al igual que Node.js, es impulsado por eventos.

Socket.IO puede usarse simplemente como un wrapper para WebSocket aunque proporciona muchas más funciones, incluyendo la transmisión a múltiples sockets, almacenamiento de datos asociados a cada cliente y E/S asíncronas.

3.1.10. FFmpeg



Web: <https://ffmpeg.org/>

FFmpeg es una colección de software libre que puede grabar, convertir (transcodificar) y hacer streaming de audio y vídeo. Incluye libavcodec, una biblioteca de códecs. FFmpeg está desarrollado en GNU/Linux, pero puede ser compilado en la mayoría de los sistemas operativos, incluyendo Windows.

FFmpeg es un programa bastante sencillo y muy fácil de usar, orientado tanto a personas con conocimientos avanzados como usuarios inexpertos.

El proyecto FFmpeg está compuesto por:

- ffmpeg: es una herramienta de línea de comandos para convertir audio o video de un formato a otro. También puede capturar y codificar en tiempo real desde DirectShow, una tarjeta de televisión u otro dispositivo compatible.
- ffserver: es un servidor de streaming multimedia de emisiones en directo que soporta HTTP (la compatibilidad con RTSP está en desarrollo). Todavía no está en fase estable, y de momento no está disponible para Windows.
- ffplay: es un reproductor multimedia basado en SDL y las bibliotecas FFmpeg.
- libavcodec: es una biblioteca que contiene todos los códecs de FFmpeg. Muchos de ellos fueron desarrollados desde cero para asegurar una mayor eficiencia y un código altamente reutilizable.
- libavformat: es una biblioteca que contiene los multiplexadores/demultiplexadores para los archivos contenedores multimedia.
- libavutil: es una biblioteca de apoyo que contiene todas las rutinas comunes en las diferentes partes de FFmpeg.
- libpostproc: es una biblioteca de funciones de postproceso de vídeo.
- libswscale: es la biblioteca de escalado de vídeo.

Para el desarrollo de RobotUI, concretamente para la transmisión de vídeo desde el robot de pruebas desarrollado hacia el cliente, el módulo utilizado ha sido el de la herramienta de línea de comandos.

3.1.11. Bootstrap



Web: <http://getbootstrap.com/>

Bootstrap es un framework o conjunto de herramientas de Código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales. Se ha utilizado en el presente proyecto para la maquetación de la aplicación.

3.1.12. JQuery



Web: <https://jquery.com/>

JQuery es una biblioteca de JavaScript rápida, pequeña y característica. Hace que las cosas como HTML documento transversal y manipulación, manejo de eventos, animación, y Ajax mucho más simple con una fácil de usar API que funciona a través de una multitud de navegadores. Con una combinación de versatilidad y extensibilidad, JQuery ha cambiado la forma en que millones de personas escriben JavaScript.

3.1.13. Mongo DB



Web: <https://www.mongodb.com/es>

MongoDB (de la palabra en inglés “humongous” que significa enorme) es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En lugar de guardar los datos en tablas como se hace en las base de datos relacionales,

MongoDB guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

3.1.14. Pm2



Web: <http://pm2.keymetrics.io/>

PM2 es un gestor de procesos de producción para aplicaciones Node.js con un equilibrador de carga incorporado. Le permite mantener las aplicaciones vivas para siempre, recargarlas sin tiempo de inactividad y facilitar las tareas comunes del administrador del sistema.

3.1.15. Robomongo



3.2. Tecnologías hardware y materiales utilizados

3.2.1. Raspberry Pi Model B



Figura 3.2: Imagen de una Raspberry Pi 3 Model B

Web: <https://www.raspberrypi.org>

La Raspberry Pi 3 es la tercera generación de Raspberry Pi. Sus especificaciones son las siguientes:

- Una CPU ARMv8 quad-core de 64 bits de 64 bits y 1.2 GHz
- LAN inalámbrica 802.11n
- Bluetooth 4.1
- Bluetooth baja energía (BLE)
- 1 GB de RAM
- 4 puertos USB
- 40 conexiones GPIO
- Puerto HDMI
- Puerto Ethernet
- Conector de audio combinado de 3,5 mm y vídeo compuesto
- Interfaz de la cámara (CSI)
- Interfaz de pantalla (DSI)
- Ranura para tarjeta Micro SD
- VideoCore IV núcleo de gráficos 3D

3.2.2. Controladora de motores L298N

El módulo controlador de motores L298N H-bridge nos permite controlar la velocidad y la dirección de dos motores de corriente continua o un motor paso a paso de una forma muy sencilla, gracias a los 2 los dos H-bridge que dispone.

básicamente un puente-H o H-bridge es un componente formado por 4 transistores que nos permite invertir el sentido de la corriente, y de esta forma podemos invertir el sentido de giro del motor.

El rango de tensiones en el que trabaja este módulo va desde 3V hasta 35V, y una intensidad de hasta 2A. A la hora de alimentarlo hay que tener en cuenta que la electrónica del módulo consume unos 3V, así que los motores reciben 3V menos que la tensión con la que alimentemos el módulo.

Además el L298N incluye un regulador de tensión que nos permite obtener del módulo una tensión de 5V, perfecta para alimentar nuestro Arduino. Eso sí, este regulador sólo funciona si alimentamos el módulo con una tensión máxima de 12V.

Es un módulo que se utiliza mucho en proyectos de robótica, por su facilidad de uso y su reducido precio.

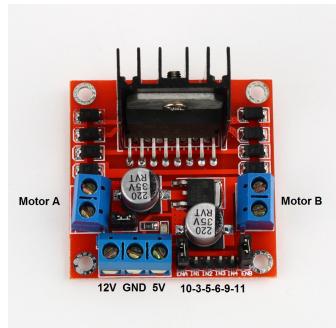


Figura 3.3: Imagen de la controladora de motores L298n utilizada

3.2.3. Batería LiPo

Para alimentar el conjunto se ha empleado una batería LiPo de 1000mAh a 3,7V. Las baterías de polímero de iones de litio, son pilas recargables (células de secundaria), compuestas generalmente de varias células secundarias idénticas en paralelo para aumentar la capacidad de la corriente de descarga. Siendo ideales para este tipo de usos.



Figura 3.4: Imagen de la batería LiPo utilizada.

3.2.4. Tarjeta de expansión con batería de Litio para Raspberry Pi

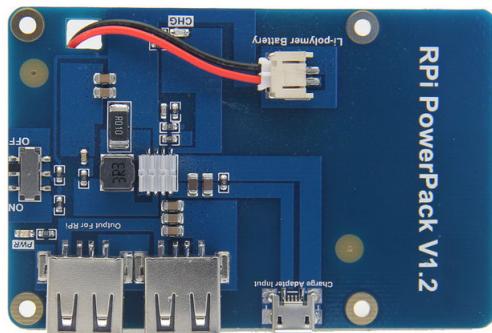


Figura 3.5: Imagen de la tarjeta de expansión con batería de Litio utilizado.

3.2.5. Cámara USB de alta definición

Cámara USB para su conexión en la Raspberry para la emisión de imágenes. La cámara seleccionada dispone de las siguientes características:

- 2 megapíxeles de resolución.
- Ángulo de visión de 170 grados.

- Interfaz USB 2.0 de alta velocidad, refresco de 60 fps en resolución 1280X720, 30 fps en resolución 1920X1080
- Tamaño reducido y perfil delgado ideal para aplicaciones embebidas.



Figura 3.6: Imagen de la cámara USB utilizada.

Capítulo 4

Desarrollo software

4.1. Metodología de desarrollo

Este proyecto ha sido obtenido empleando una metodología de desarrollo basada en el modelo de desarrollo incremental para la parte software referente a todos los subsistemas web y una metodología de desarrollo en cascada para el desarrollo de la parte software del robot de pruebas.

El modelo de desarrollo incremental proporciona una serie de características que lo hacen idóneo para este proyecto. Dicho modelo se basa en la filosofía de construir e ir incrementando las funcionalidades del sistema mediante el desarrollo de los diferentes módulos. Esto permite ir aumentando gradualmente las capacidades del software.

Dicha metodología de desarrollo resulta especialmente útil en las siguientes situaciones:

- Facilita el desarrollo permitiendo a cada miembro del equipo desarrollar un módulo particular. En el caso del presente proyecto me ha permitido desarrollar un módulo tras otro de una manera secuencial.
- Es similar al ciclo de vida en cascada aplicándose un ciclo en cada nueva funcionalidad del programa.
- A final de cada ciclo se entrega el software al cliente. En el caso que compete a este proyecto se mantenía una reunión con el director del proyecto para su aprobación.

Por otro lado, el modelo de desarrollo en cascada resulta adecuado en situaciones en las que:

- Se dispone de unos requisitos claros y precisos.
- El sistema a desarrollar es de pequeña envergadura.

- Las tecnologías utilizadas son conocidas por los desarrolladores.

Centrándonos nuevamente en el desarrollo del proyecto, los motivos que llevaron a cabo la elección de un modelo de desarrollo incremental viene dada por la necesidad de simplificar e ir desarrollando de una forma gradual y modularizada debido a la extensión del proyecto. Más si cabe que el equipo de desarrollo solo consta de una persona.

Por otro lado, para el desarrollo del vehículo de pruebas y por su simplicidad, se ha optado por un desarrollo en cascada además de que en otras ocasiones ya había realizado trabajos similares.

Por tanto el proyecto queda distribuido en los siguientes subsistemas:

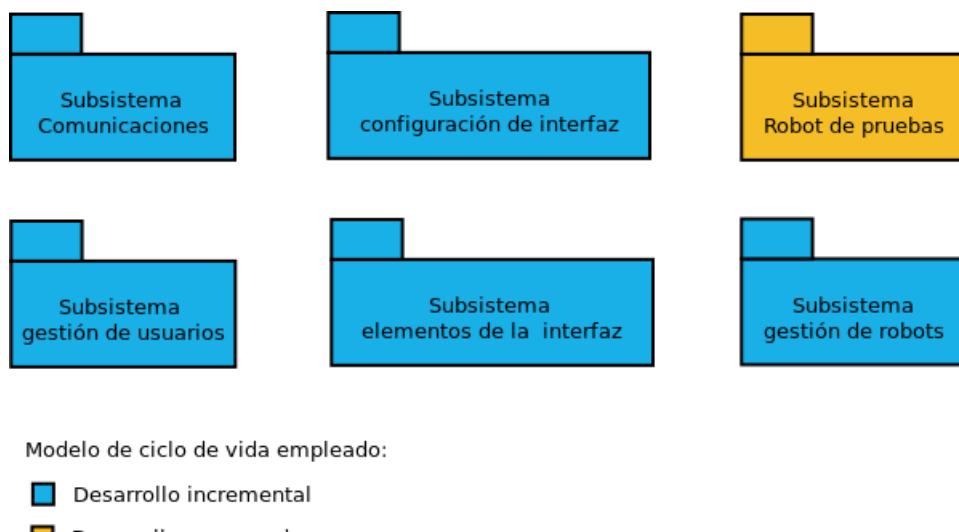


Figura 4.1: Subsistemas existentes en el proyecto junto con el modelo de ciclo de vida utilizado para su desarrollo.

4.2. Recolección de requisitos

4.2.1. Requisitos funcionales

Para la elaboración de este proyecto se ha utilizado la metodología Métrica V3 así como el estándar ISO/IEC 12207. Métrica es una metodología de planificación, desarrollo y mantenimiento de los sistemas de información desarrollada por el Ministerio de Administraciones Públicas del Gobierno de España. Tiene como objetivo proporcionar una guía para la sistematización de actividades del ciclo de vida de los proyectos software en el ámbito de las administraciones públicas.

Métrica V3 [?] está basada en el modelo de procesos del ciclo de vida de desarrollo ISO/IEC 12207 (Information Technology - Software Life Cycle Processes) así como en

la norma ISO/IEC 15504 SPICE (Software Process Improvement And Assurance Standards Capability Determination).

En la página oficial de Métrica V3, sus desarrolladores indican que puede ser utilizada libremente con la única restricción de citar la fuente de su propiedad intelectual, la del Ministerio de Administraciones Públicas.

En esta etapa del modelado de requisitos se captura el propósito general del sistema:

- Se analiza qué debe hacer el sistema.
- Se obtiene una versión contextualizada del sistema.
- Identifica y delimita el sistema.
- Se determinan las características, cualidades y restricciones que debe satisfacer el sistema.

Por tanto, Los requisitos funcionales que se han obtenido después del proceso de obtención de requisitos son:

- Definir los pasos para dar de alta un dispositivo robótico en el sistema.
- Una vez configurado el dispositivo, configurar la interfaz de control con las acciones de control específicas.
- Realizar un sistema de monitorización y visualización para los usuarios y los dispositivos robóticos en tiempo real.
- Sistema de gestión de base de datos en donde se encuentren los datos de la aplicación recogidos.
- Disponer de un panel de administración donde visualizar la información de los usuarios conectados y dispositivos en uso en tiempo real.
- Proporcionar un sistema de streaming de vídeo para la difusión de imágenes a los usuarios espectadores procedentes de los robots dispongan de cámara.
- Deberá ser una herramienta multiplataforma.

4.2.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos que describen cualidades o restricciones del sistema que no se relacionan de forma directa con el comportamiento funcional del mismo. A continuación se especifican los más importantes del sistema:

- No requiere un conocimiento específico del sistema una vez puesto en funcionamiento.

- La aplicación tendrá manual de uso.
- La base de datos estará implementada en un lenguaje objeto no relacional como MongoDB.
- La aplicación estará realizada en el lenguaje de programación Python.
- La interfaz debe reflejar claramente la distinción entre las distintas partes del sistema.
- El sistema se desplegará sobre una versión GNU Linux Debian 8 Jessie.
- El código fuente de la aplicación seguirá un estilo uniforme y normalizado para todos los módulos del mismo.

4.3. Diagramas de casos de uso

Un caso de uso es una descripción de los pasos o actividades que deberán realizarse para llevar a cabo algún proceso. Los objetivos de los casos de uso son los siguientes:

- Obtener los requisitos funcionales del sistema y expresarlos desde un punto de vista más cercano al usuario.
- Proporcionar una guía de todo el proceso de desarrollo del sistema de información. Los casos de uso proporcionan, por tanto, un modo claro y preciso de comunicación entre lo que desea el cliente y el desarrollador proporcionando desde el punto de vista del cliente una visión de “caja negra” del sistema eliminando los detalles de su construcción o desarrollo. Para los desarrolladores es utilizado como punto de partida y el eje sobre el que se apoya todo el desarrollo del sistema en los procesos de análisis y diseño.

Los diagramas de caso de uso muestra una representación del comportamiento ofrecido por el sistema de información desde el punto de vista del usuario. El comportamiento del sistema es representado como un conjunto de transacciones ejecutadas entre el sistema y los actores junto con la descripción sobre las de las relaciones de comunicación existentes entre un actor y el sistema.

En la figura 4.3 podemos observar los diagramas de casos de uso resultantes tras la recolección de requisitos:

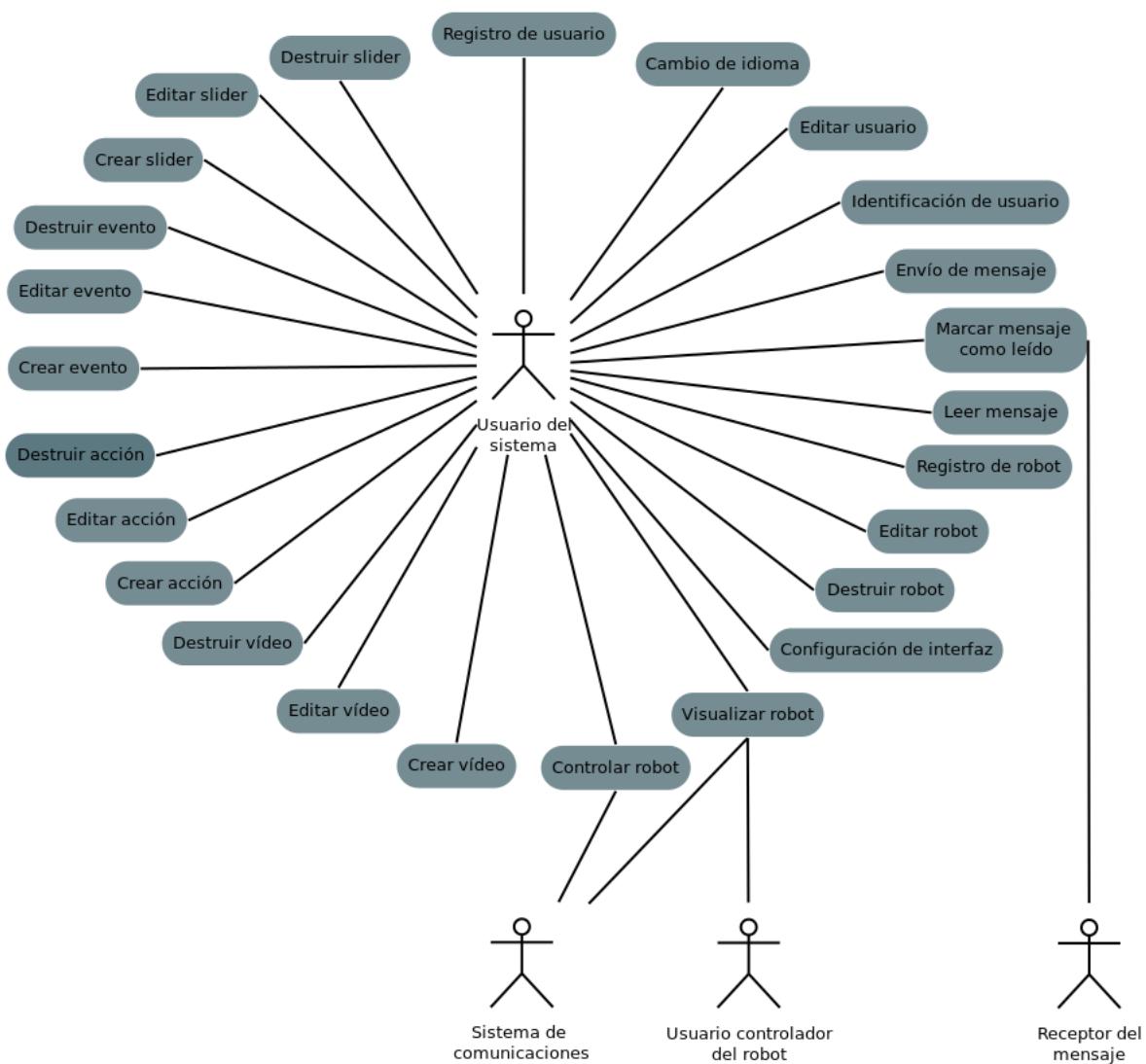


Figura 4.2: Diagrama de casos de uso para la interacción con la aplicación.

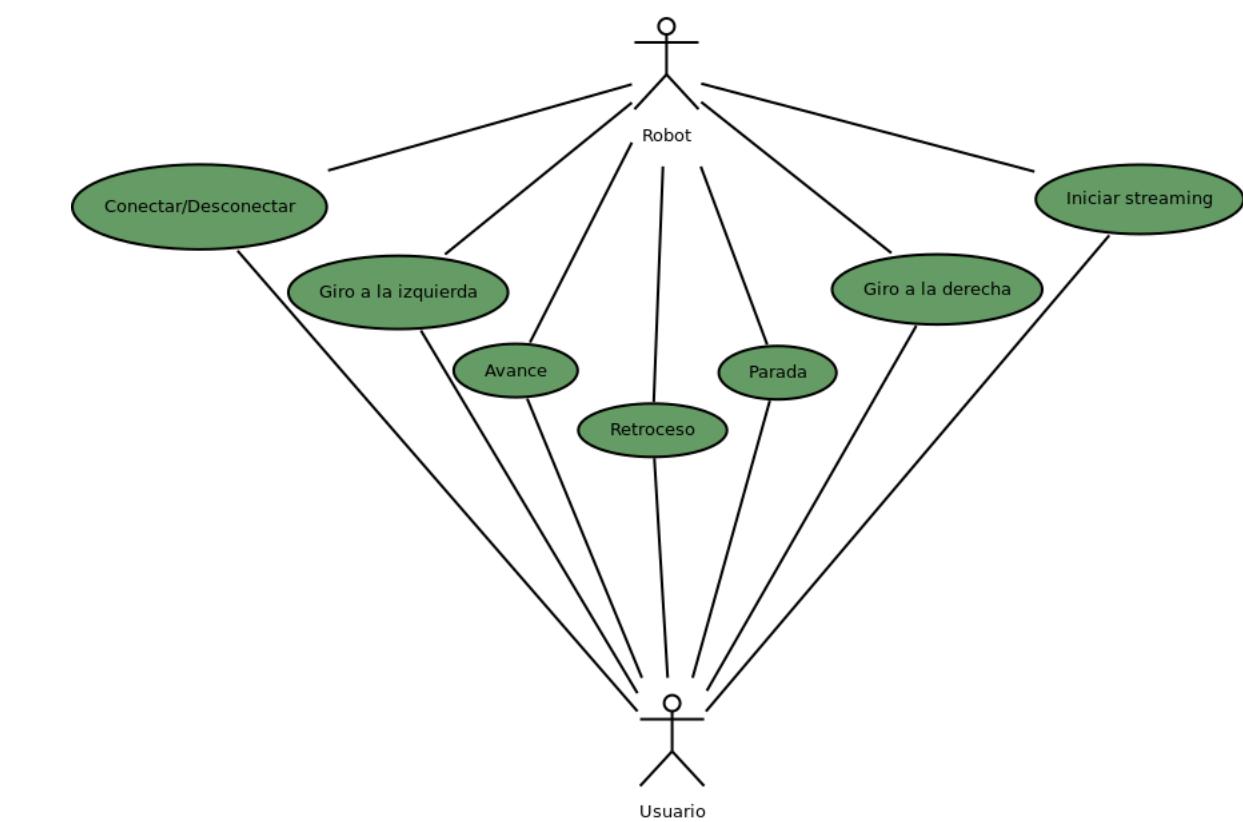


Figura 4.3: Diagrama de casos de uso para la interacción con el robot de pruebas.

4.4. Especificación de los casos de uso

A continuación se proporciona la especificaciones de cada uno de los casos de uso de la figura 4.3:

CAPÍTULO 4. DESARROLLO SOFTWARE. ESPECIFICACIÓN DE LOS CASOS DE USO

Caso de uso:	Registro de usuario.
Descripción:	El sistema deberá actuar como describe en este caso de uso cuando el usuario solicita el registro en el sistema.
Actor principal:	Usuario del sistema.
Actor secundario:	-
Precondiciones:	-
Flujo principal:	<ol style="list-style-type: none"> 1. El actor usuario del sistema solicita el registro en el sistema. 2. El sistema solicita los datos correo electrónico, nombre de usuario, avatar, contraseña, confirmación de contraseña e idioma. 3. El usuario proporciona al sistema al menos los siguientes datos, correo electrónico, nombre de usuario, contraseña y confirmación de contraseña. 4. El sistema realiza el registro de un nuevo usuario e informa de que el registro se ha realizado con éxito.
Postcondición	Se realiza el registro de un usuario en el sistema.
Excepciones:	<ol style="list-style-type: none"> 1. Si existe un usuario con el correo electrónico introducido: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar el registro. • Se cancela el caso de uso. 2. Si la contraseña y la confirmación no coinciden: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar el registro. • Se cancela el caso de uso.

Tabla 4.1: Descripción del caso de uso: Registro de usuario.

Caso de uso:	Identificación de usuario.
Descripción:	El sistema deberá actuar como describe en este caso de uso cuando el usuario solicita la Identificación en el sistema.
Actor principal:	Usuario del sistema.
Actor secundario:	-
Precondiciones:	El usuario debe estar registrado en el sistema
Flujo principal:	<ol style="list-style-type: none"> 1. El actor usuario del sistema solicita la Identificación en el sistema. 2. El sistema solicita los datos correo electrónico y contraseña. 3. El usuario proporciona al sistema el correo electrónico y contraseña. 4. El sistema realiza el logueo del usuario y crea una sesión.
Postcondición	Se realiza la identificación del usuario en el sistema, se crea una sesión y se carga el idioma establecido por el usuario.
Excepciones:	<ol style="list-style-type: none"> 1. Si no existe un usuario con el correo electrónico introducido: <ul style="list-style-type: none"> • El sistema informa de que no existe ningún usuario con ese correo electrónico. • Se cancela el caso de uso. 2. Si la contraseña es incorrecta: <ul style="list-style-type: none"> • El sistema informa de que la contraseña es incorrecta. • Se cancela el caso de uso.

Tabla 4.2: Descripción del caso de uso: Identificación de usuario.

CAPÍTULO 4. DESARROLLO SOFTWARE. ESPECIFICACIÓN DE LOS CASOS DE USO

Caso de uso:	Cambio de idioma.
Descripción:	El sistema deberá actuar como describe en este caso de uso cuando el usuario solicita cambiar de idioma la página.
Actor principal:	Usuario del sistema.
Actor secundario:	-
Precondiciones:	-
Flujo principal:	<ol style="list-style-type: none">1. El actor usuario del sistema selecciona un idioma disponible.2. El sistema establece el idioma seleccionado.
Postcondición	Se realiza el cambio de idioma en la aplicación.
Excepciones:	-

Tabla 4.3: Descripción del caso de uso: Cambio de idioma.

Caso de uso:	Editar usuario.
Descripción:	El sistema deberá actuar como describe en este caso de uso cuando el usuario solicita editar usuario.
Actor principal:	Usuario del sistema.
Actor secundario:	-
Precondiciones:	El usuario debe estar registrado en el sistema
Flujo principal:	<ol style="list-style-type: none"> 1. El actor usuario del sistema selecciona editar su perfil. 2. El sistema solicita los nuevos datos de usuario (correo electrónico, nombre de usuario, avatar, contraseña, confirmación de contraseña e idioma). 3. El usuario modifica al menos uno de los datos anteriores. 4. El sistema realiza la modificación de los datos al usuario.
Postcondición	Se realiza la modificación de los datos al usuario.
Excepciones:	<ol style="list-style-type: none"> 1. Si existe un usuario con el correo electrónico introducido: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar el registro. • Se cancela el caso de uso. 2. Si la contraseña y la confirmación no coinciden: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar el registro. • Se cancela el caso de uso.

Tabla 4.4: Descripción del caso de uso: Edición de usuario.

CAPÍTULO 4. DESARROLLO SOFTWARE. ESPECIFICACIÓN DE LOS CASOS DE USO

Caso de uso:	Envío de mensaje usuario.
Descripción:	El sistema deberá actuar como describe en este caso de uso cuando el usuario solicita enviar un mensaje.
Actor principal:	Usuario del sistema.
Actor secundario:	Receptor del mensaje
Precondiciones:	El usuario debe estar registrado e identificado en el sistema
Flujo principal:	<ol style="list-style-type: none"> 1. El actor usuario del sistema solicita comenzar el proceso de envío de un mensaje. 2. El sistema solicita los nuevos datos del mensaje (título, destinatario y contenido del mensaje). 3. El usuario proporciona los datos anteriores y ordena mandar el mensaje. 4. El sistema realiza el envío del mensaje al destinatario e informa al receptor de que dispone un nuevo mensaje en su bandeja de entrada.
Postcondición	Se realiza el envío de un mensaje a un usuario.
Excepciones:	<ol style="list-style-type: none"> 1. Si el usuario no se encuentra logueado en el sistema: <ul style="list-style-type: none"> • El sistema informa de que es necesario identificarse. • Se cancela el caso de uso. 2. Si el usuario no ha introducido título, contenido o destinatario: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar el envío. Faltan datos obligatorios. • Se regresa al punto 2.

Tabla 4.5: Descripción del caso de uso: Envío de mensaje.

Caso de uso:	Leer mensaje.
Descripción:	El sistema deberá actuar como describe en este caso de uso cuando el usuario solicita leer un mensaje.
Actor principal:	Usuario del sistema.
Actor secundario:	-
Precondiciones:	El usuario debe estar registrado, logueado y disponer de al menos un mensaje en su bandeja de entrada.
Flujo principal:	<ol style="list-style-type: none"> 1. El actor usuario del sistema solicita leer un mensaje de su bandeja de entrada. 2. El sistema muestra contenido del mensaje y lo establece como leído.
Postcondición	Se muestra el contenido del mensaje.
Excepciones:	<ol style="list-style-type: none"> 1. Si el usuario no se encuentra logueado en el sistema: <ul style="list-style-type: none"> • El sistema informa de que es necesario identificarse. • Se cancela el caso de uso.

Tabla 4.6: Descripción del caso de uso: Leer mensaje.

CAPÍTULO 4. DESARROLLO SOFTWARE. ESPECIFICACIÓN DE LOS CASOS DE USO

Caso de uso:	Registro de un robot.
Descripción:	El sistema deberá actuar como describe en este caso de uso cuando el usuario solicita el registro de un nuevo robot en el sistema.
Actor principal:	Usuario del sistema.
Actor secundario:	-
Precondiciones:	Usuario registrado e identificado en el sistema.
Flujo principal:	<ol style="list-style-type: none"> 1. El actor usuario del sistema solicita el registro de un robot en el sistema. 2. El sistema solicita los datos del robot, nombre, dirección IP, puerto, descripción, imagen, usuarios espectadores, usuarios controladores y si es de control abierto o visualización abierta . 3. El usuario proporciona al sistema al menos los siguientes datos, nombre, dirección Ip, puerto, y permisos. 4. El sistema realiza el registro de un nuevo robot e informa de que la creación se ha realizado con éxito.
Postcondición	Se realiza el registro de un robot en el sistema ligado al usuario que lo ha creado.
Excepciones:	<ol style="list-style-type: none"> 1. Si el usuario no se encuentra logueado en el sistema: <ul style="list-style-type: none"> • El sistema informa de que es necesario identificarse. • Se cancela el caso de uso. 2. Si faltan datos obligatorios: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar el registro. • Vuelve al paso 2. 3. Si la dirección IP es inválida: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar el registro. • Vuelve al paso 2. 4. Si el puerto no es un valor numérico: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar el registro. • Vuelve al paso 2.

4.4. ESPECIFICACIÓN DE LOS CASOS DE USO

ÍNDICE

Caso de uso:	Editar robot.
Descripción:	El sistema deberá actuar como describe en este caso de uso cuando el usuario solicita la edición de un robot en el sistema.
Actor principal:	Usuario del sistema.
Actor secundario:	-
Precondiciones:	Usuario registrado e identificado en el sistema y propietario del robot.
Flujo principal:	<ol style="list-style-type: none"> 1. El actor usuario del sistema solicita la edición de un robot en el sistema. 2. El sistema solicita los datos del robot, nombre, dirección IP, puerto, descripción, imagen, usuarios espectadores, usuarios controladores y si es de control abierto o visualización abierta . 3. El usuario proporciona al sistema al menos los siguientes datos, nombre, dirección Ip, puerto, y permisos. 4. El sistema realiza la modificación de los datos del robot e informa de que la edición se ha realizado con éxito.
Postcondición	Se realiza la edición de un robot en el sistema.
Excepciones:	<ol style="list-style-type: none"> 1. Si el usuario no se encuentra logueado en el sistema: <ul style="list-style-type: none"> • El sistema informa de que es necesario identificarse. • Se cancela el caso de uso. 2. Si el robot no pertenece al usuario: <ul style="list-style-type: none"> • El sistema informa de que no tiene permisos. • Se cancela el caso de uso. 3. Si faltan datos obligatorios: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar la edición. • Vuelve al paso 2. 4. Si la dirección IP es inválida: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar el registro. • Vuelve al paso 2. 5. Si el robot ya no existe: <ul style="list-style-type: none"> • El sistema informa de que no se puede realizar la edición. • Vuelve al paso 2.

Capítulo 5

Comunicaciones

En el presente capítulo comenzaremos con una introducción del funcionamiento y los fundamentos teóricos de cómo se gestionan las diferentes conexiones y eventos del framework para, posteriormente, centrarnos en el ámbito específico de este proyecto con la finalidad de comprender mejor su funcionamiento.

5.0.1. Fundamentos

En esta sección se describe cómo Sails trabaja con websocket y socket.io. Para ello se explicará con un ejemplo que facilite su comprensión.

Bien, pero ¿qué diablos son los eventos en tiempo real del modelo? Al igual que creamos manejadores de eventos para eventos DOM usando jquery, voy a mostrar un patrón para configurar los controladores de eventos para los cambios en los modelos que se emiten a cualquier socket que esté suscrito al evento.

REVISAR INTRODUCCION <http://irlnathan.github.io/sailscasts/blog/2013/10/10/building-a-sails-application-ep21-integrating-socket-dot-io-and-sails-with-custom-controller-actions-using-real-time-model-events/>

Centrémonos en el modelo usuario de RobotUI. Uno de sus atributos de este modelo es un booleano *online*, el cual representa si un usuario se encuentra logueado en el sistema. Uno de los objetivos es saber cuando el usuario cambia el estado para poder proporcionar una actualización en tiempo real de la página de gestión de usuarios siempre y cuando un usuario inicie sesión o salga de la aplicación alternando entre las imágenes online:  y offline:  sin necesidad de refrescar la página manualmente.

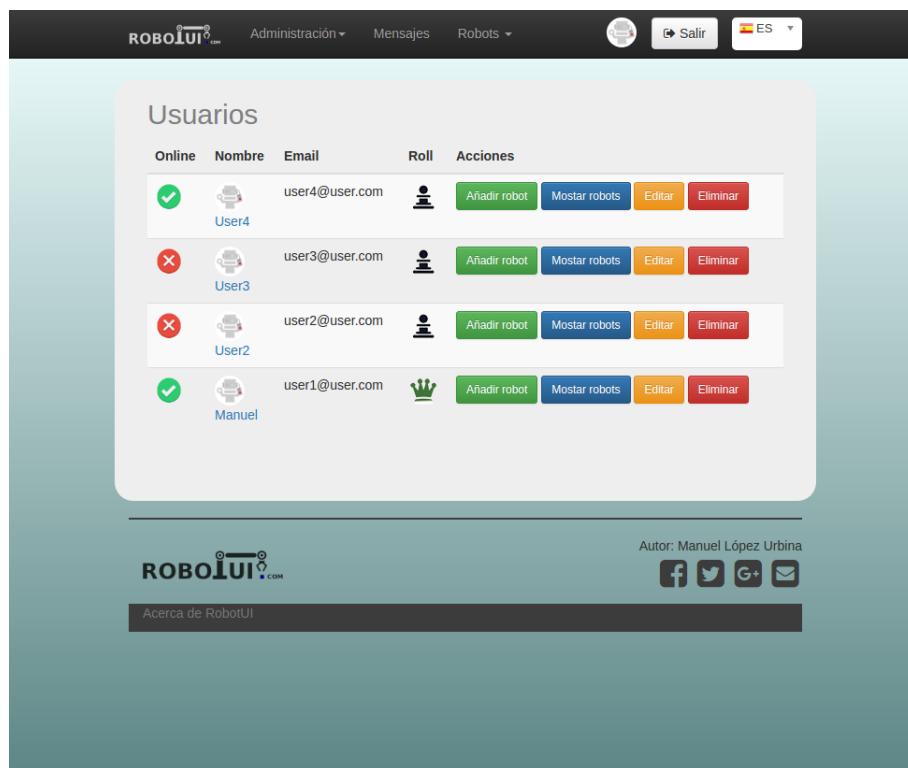


Figura 5.1: Página de gestión de usuarios actualizable en tiempo real.

5.0.2. Comunicaciones en RobotUI

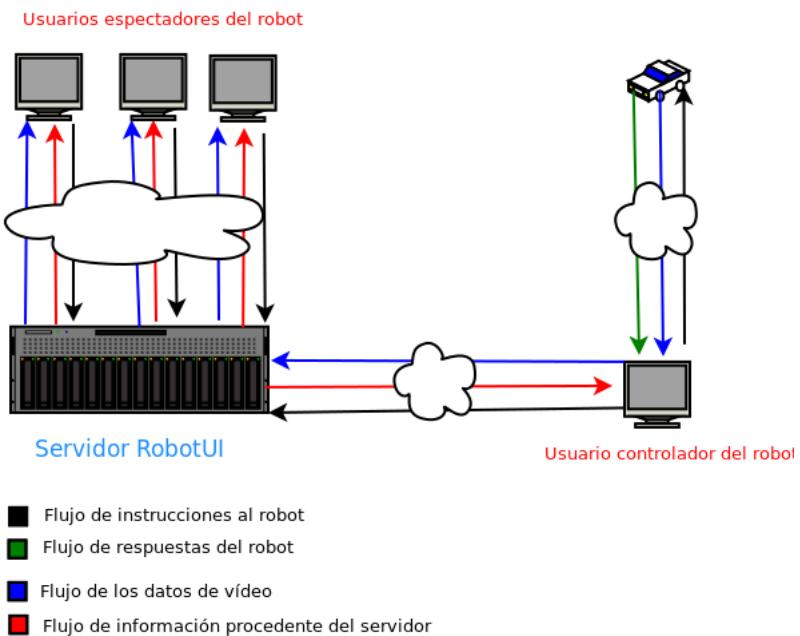


Figura 5.2: Esquema representativo del flujo de conexiones de RobotUI.

CAPÍTULO 5. COMUNICACIONES

Conexión

Cuando se realiza una conexión de un usuario al sistema, un nuevo cliente, realiza una petición al servidor, el cual captura la llamada y renderiza la vista correspondiente elaborando una respuesta.

Dicha respuesta se corresponde con el código Html y JavaScript que recibe el navegador. En el archivo *views/layout.js* se encuentra el siguiente código JavaScript de gran importancia para el funcionamiento de toda la aplicación por lo que resulta interesante su comprensión.

Primeramente al realizarse una nueva conexión de un cliente en el sistema el servidor debe conocer y registrar a dicho cliente con la finalidad de seguir su comportamiento, sus acciones y movimientos por toda la aplicación.

```
1 //Funcion para tener en todo momento almacenado en la tabla session
  los sockets conectados junto con el usuario al que pertenece
2 function savesocket() {
3   io.socket.get("/session/saveSocketID");
4 }
```

Función que realiza el registro de un nuevo cliente en la base de datos:

```
1
2 //Almacenamiento en la base de datos la sesión de cada usuario de la
  página:
3 saveSocketID: function(req, res) {
4   if (!req.isSocket) return res.badRequest();
5
6   var socketId = sails.sockets.id(req);
7   // => "BetX2G-2889Bg22xi-jy"
8
9   if(req.session.User != undefined) {
10     var sessionObj = {
11       socket_id: socketId,
12       user_id: req.session.User.id
13     };
14
15
16   //Comprobar si el usuario tiene sockets abiertos:
17   Session.count({user_id: req.session.User.id}).exec(function
18     countUserSessions(error, n_sessions) {
19       console.log('There are ' + n_sessions + ' sessions of user ' +
20         req.session.User.id);
21
22       if (n_sessions == 0) {
23         //Cambio de estado del usuario a online:
24         User.update(req.session.User.id, {online: true}, function
25           (err) {
26             if (err) return res.badRequest();
27           }
28         );
29       }
30     }
31   );
32 }
33 }
```

```

25         //Informar a otros clientes (sockets abiertos) que el
26         //usuario esta logueado:
27         User.publishUpdate(req.session.User.id, {
28             loggedIn: true,
29             id: req.session.User.id
30         });
31     });
32 );
33
34 }else {
35     var sessionObj = {
36         socket_id: socketId
37     };
38 }
39
40 Session.create(sessionObj).exec( function (err, session) {
41     if (err) return res.badRequest();
42
43     console.log('\n.....');
44     console.log('Conectando con Sails
45         js.....');
46     console.log('Cliente conectado - id del socket: ' + socketId );
47     console.log('.....');
48
49     return;
50 });
50 ,

```

Una vez finaliza el registro en el servidor del nuevo cliente se comprueba si existe definida la función *subscribeAndListen* llamándose en caso afirmativo. Dicha función es específica para cada funcionalidad que queramos inicializar o a qué eventos nos deseemos suscribir. Por ejemplo, suscibirnos a los eventos de conexión y desconexión de usuarios del panel de administración descrito en el apartado 5.0.1 mediante la definición de la función *subscribeAndListen* en la vista correspondiente.

Por otro lado, siempre se realiza la llamada a la función *listenMessages* ya que independientemente de la vista en la que nos encontremos siempre nos mantendremos a la escucha de los mensajes recibidos a nuestra bandeja de entrada por parte de otros usuarios.

Finalmente mostramos el código JavaScript al completo localizado en el archivo *views/layout.js*:

```

1 <script type="text/javascript">
2
3
4 $.when(savesocket()).done(function(){
5     if (typeof subscribeAndListen == 'function') {
6         subscribeAndListen();
7     }
8     listenMessages();

```

CAPÍTULO 5. COMUNICACIONES

```
9  });
10
11 //Funcion para tener en todo momento almacenado en la tabla session
12 // los sockets conectados junto con el usuario al que pertenece
12 function savesocket() {
13   io.socket.get("/session/saveSocketID");
14 }
15
16 //Evento a la espera de recibir mensajes y notificar al usuario:
17 function listenMessages(){
18   io.socket.on('user', function messageReceived(message) {
19     switch (message.verb) {
20       case 'messaged':
21         var pathname = window.location.pathname;
22         if(pathname == '/message/index'){
23           location.reload();
24         }else{
25           new_msg_num_update('<%= i18n('messages') %>');
26         }
27         toaster.info('You have a new message! <a
28           href="/message/show/' + message.data.msg.id + '"><%
29             i18n('open_here')%></a>' , 'Message');
30         break;
31       default:
32         break;
33     }
34   });
35 }
36 </script>
```

Desconexión

En este apartado describiremos los puntos de mayor relevancia a la hora de la desconexión de alguna de las diferentes comunicaciones establecidas comenzando con la desconexión de un usuario.

Cuando un usuario realiza una desconexión, bien deslogueándose de la página o cerrando una de las ventanas abiertas, automáticamente se lanza una llamada a la función *afterDisconnect* localizada en el fichero *config/Socket.js* del servidor. Función que será ejecutada cada vez que un socket es desconectado del sistema.

Dicha función primeramente localiza qué sesión en la base de relacionada está relacionada con identificador del socket desconectado. Si este Socket estaba usando algún robot, éste se libera cambiando el estado del robot a *libre*. Y se informa a todos los sockets abiertos que el robot queda libreado y accesible al resto de usuarios.

```
1 Robot.update({id: session.robot_id}, {busy: false}, function
2   robotUpdated(err) {
3     if (err) return next(err);
```

```

3 //Informar a otros clientes (sockets abiertos) que el robot queda
4   liberado
5 Robot.publishUpdate(session.robot_id, {
6   busy: false,
7   id: session.robot_id
8 });
9 );

```

A continuación, se comprueba si el usuario no dispone de más sockets abiertos. Si fuera el caso de que no los disponga, entonces se procede a cambiar su estado a desconectado y se emite o se informa al resto de clientes (sockets abiertos) que el usuario ya no se encuentra en el sistema. Llamada al método *User.publishUpdate*:

```

1 User.publishUpdate(session.user_id, {
2   loggedIn: false,
3   id: session.user_id
4 });

```

Se comprueba si el usuario desconectado se encontraba en alguna *room*. Pongamos como ejemplo que el usuario ha abandonado la visualización de un robot por lo que se debe informar a todos los sockets integrantes de esa *room* que un usuario la ha abandonado con la función *sails.sockets.broadcast*. A continuación se muestra el fragmento de código:

```

1 //Emite a cada room que que se encuentre la sesión que un usuario la
2   ha abandonado
3
4 session.rooms.forEach(function (room) {
5   //sails.sockets.leave(session.socket_id, room.room_name,
6   //  function(err) {
7   //    if (err) {return res.serverError(err);}
8   //  });
9   sails.sockets.broadcast(room.room_name, {type: 'exit', msg:
10     {user_id: session.user_id}});
11 });

```

Finalmente se destruye la sesión:

```

1 Session.destroy(session.id, function sessionDestroyed(err) {
2   if (err) return cb();
3   return cb();
4 });

```

A continuación mostramos el código completo de la función *afterDisconnect*:

```

1
2 afterDisconnect: function(session, socket, cb) {
3   console.log('Cliente desconectado - id del socket: ' + socket.id);
4 }

```

CAPÍTULO 5. COMUNICACIONES

```
5 //Session del socket cerrado,
6 Session.findOne({socket_id:
7     socket.id}).populate('rooms').exec(function (err, session) {
8     if (err) return cb();
9     if (!session) return cb();
10
11     //Comprobar si el socket estaba usando algun robot para
12     //liberarlo:
13     if (session.robot_id) {
14         console.log('Socked was using a robot');
15
16         Robot.update({id: session.robot_id}, {busy: false}, function
17             robotUpdated(err) {
18                 if (err) return next(err);
19
20                 //Informar a otros clientes (sockets abiertos) que el robot
21                 //quedó liberado
22                 Robot.publishUpdate(session.robot_id, {
23                     busy: false,
24                     id: session.robot_id
25                 });
26             });
27     }
28
29
30     //Emite a cada room que que se encuentre la sesión que un
31     //usuario la ha abandonado ->
32     session.rooms.forEach(function (room) {
33         //sails.sockets.leave(session.socket_id, room.room_name,
34         //    function(err) {
35         //        if (err) {return res.serverError(err);}
36         //    });
37         sails.sockets.broadcast(room.room_name, {type: 'exit', msg:
38             {user_id: session.user_id}});
39     });
39
40
41     //Comprobar si el usuario tiene más sockets abiertos:
42     Session.count({user_id: session.user_id}).exec(function
43         countUserSessions(error, n_sessions) {
44         console.log('There are ' + n_sessions + ' users ' +
45             session.user_id);
46
47         //Cambiemos el usuario a offline si solo tenía una ventana o
48         //conexión abierta.
49         if (n_sessions == 1) {
50             User.update(session.user_id, {online: false}, function (err)
51                 {
52                     if (err) return cb(err);
53
54                     //Informar a otros clientes (sockets abiertos) que el
55                     //usuario ya NO se encuentra logueado
56                     User.publishUpdate(session.user_id, {
57                         loggedIn: false,
58                         id: session.user_id
59                     });
60                 });
61         }
62     });
63
64 }
```

```
48      });
49    });
50  }
51
52  Session.destroy(session.id, function sessionDestroyed(err) {
53    if (err) return cb();
54    return cb();
55  });
56  });
57  });
58 }
```

Capítulo 6

Robot de pruebas

Con la finalidad de demostrar y hacer un primer uso de la aplicación se ha decidido abordar la elaboración de un vehículo de pruebas. Dicho vehículo será utilizado de modelo o guía para el resto de personas que quieran crear un robot para su integración en la aplicación o bien para programar un robot del que y dispongan previamente.

En el presente capítulo se detalla los diferentes pasos que se han seguido a la hora de la construcción y programación del vehículo a modo de guía y ejemplo paso a paso ¹.

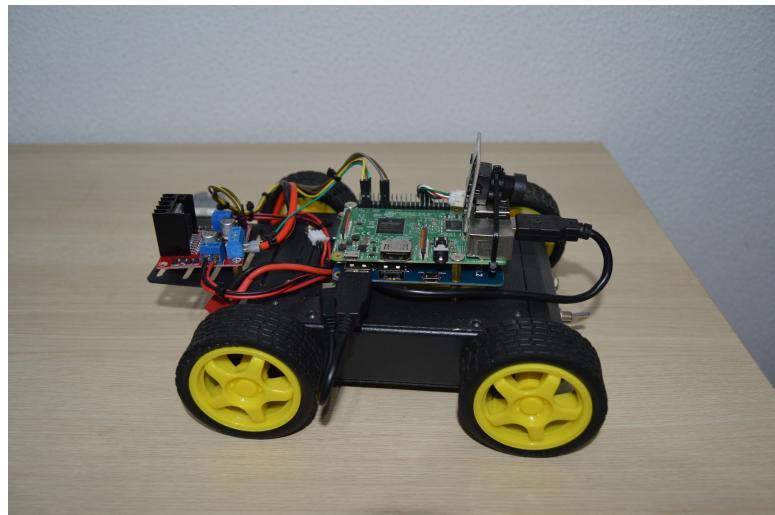


Figura 6.1: Imagen del robot de pruebas.

6.1. Análisis

??

¹aclaracion: esto es una aclaración.

6.1.1. Análisis de requerimientos hardware

Se ha optado por la construcción de un pequeño robot móvil dotado de un chasis de 4 ruedas donde cada una de ellas está accionada por un motor. Los motores seleccionados funcionan a corriente continua de manera que en función de la polarización de los terminales hace girar las ruedas en una dirección o la contraria (marcha adelante o atrás).

El chasis seleccionado permite añadir multitud de componentes que deseas para construir tu robot gracias a sus dos paneles para instalar las diferentes placas electrónicas y sensores.

El robot además necesita de una cámara para la obtención de vídeo y su transmisión. Para ello se ha empleado una cámara de pequeñas dimensiones de alta definición.

Por otra parte, todo robot necesita de una unidad de central de procesamiento donde se localizará el programa de control. Este programa tendrá la función de interpretar las diferentes señales recibidas, control de sensores y dispositivos conectados. Además, esta placa es la encargada de distribuir la alimentación por los diferentes componentes hardware que lo necesiten y recibir las señales de los sensores y enviarla a los motores. Utilizando para ello la placa Raspberry Pi modelo B cuya descripción se encuentra en la sección ??.

Este modelo de placa dispone de una serie de pines denominados GPIO (General Purpose Input/Output) que son, como su propio nombre indica, un sistema de E/S (Entrada/Salida) de propósito general, es decir, una serie de conexiones que se pueden usar como entradas o salidas para usos múltiples. Estos pines están incluidos en todos los modelos de Raspberry Pi, con la finalidad de ser utilizados en diferentes proyectos de una manera similar a la que se haría con Arduino²

6.2. Análisis de requerimientos de Software

Habiendo detallado los requerimientos hardware para la construcción del robot, pasamos al análisis del programa implementado para su correcto funcionamiento.

La programación del robot uno de los requisitos fundamentales era el de disponer de una vía de comunicación bidireccional.

Para ello se ha seleccionado el entorno de ejecución Node.Js, el cual permite escribir programas en JavaScript además de disponer de una gran cantidad de bibliotecas hechas por toda una comunidad que la respalda. Además de que permite escribir programas relativamente complejos en tan sólo unas pocas líneas de código.

²Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar.

se ha realizado haciendo uso de Websockets los cuales presentan algunas ventajas sobre simples solicitudes http:

- **Velocidad:** Una petición http normal tiene que establecer una conexión antes de comenzar las transacciones, la cual toma bastante tiempo. Los Websockets, una vez establecida la conexión, siempre están abiertos y listos para enviar o recibir datos. Esto significa que el retraso puede ser tan bajo como su ping, en torno a un milisegundo o dos en la mayoría de los casos.
- **Bidireccional:** Los Websockets permiten transmisión de datos en ambas direcciones permitiendo la activación de eventos en el cliente y viceversa.

Como podemos ver las propiedades anteriormente descritas resultan esenciales para nuestro proyecto que, como cabe recordar, queremos realizar lectura de sensores y envío de órdenes desde un servidor externo. Además de la transmisión de vídeo.

6.3. Montaje

En esta sección se recogen todas las descripciones y procedimientos seguidos y que han resultado de mayor interés a la hora de la construcción del robot y sus diferentes interconexiones.

6.3.1. Interconexión de elementos

En la sección ?? se describen los diferentes elementos hardware que han sido necesarios para la construcción del vehículo.

Para la placa Raspberry Pi Model B+, los pines GPIO se encuentran distribuidos de la siguiente manera:

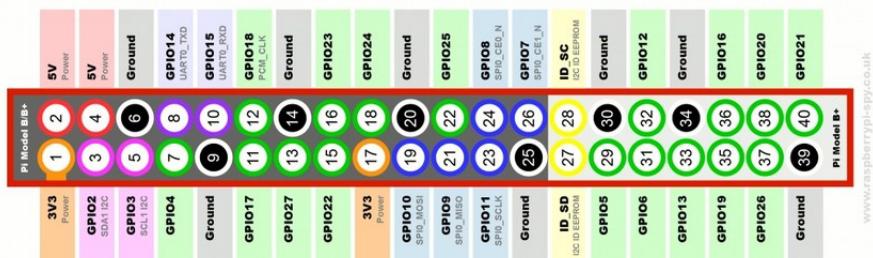


Figura 6.2: Esquema GPIO de una Raspberry Pi Model B+.

Los pines empleados son los siguientes:

CREAR TABLA CON PINES Y USOS

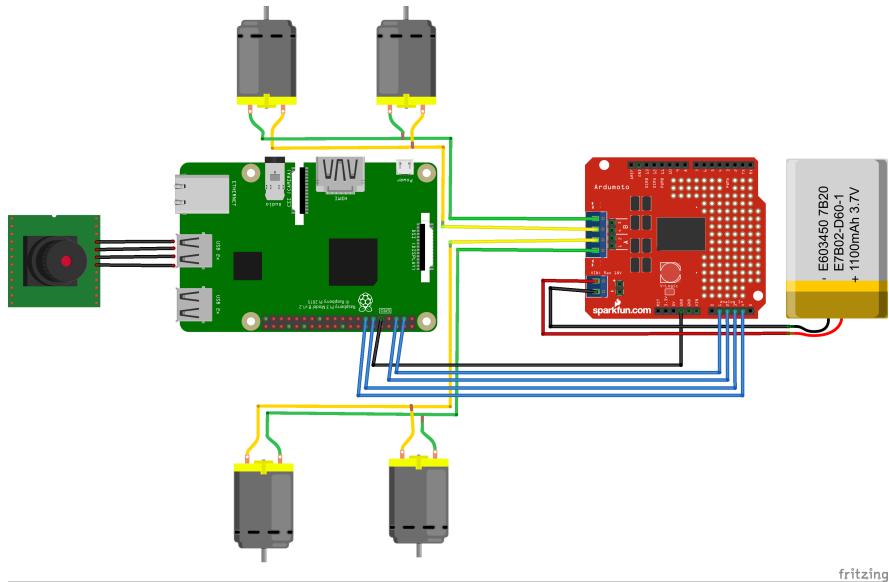


Figura 6.3: Esquema de conexiones del robot de pruebas.

6.4. Software de control

Para la programación del robot se ha empleado el lenguaje de programación JavaScript en un entorno de ejecución Node.js. A continuación se describirá aquellos aspectos más importantes referentes al código desarrollado para el control del robot.

Primeramente se ha realizado la carga de librerías necesarias, entre ellas encontramos:

- *pigpio*: Módulo para la comunicación y control de los pines GPIO.
- *child process*: El módulo *child_process* proporciona la capacidad de generar procesos secundarios. Se ha empleado para la captura de vídeo mediante el lanzado de comandos *ffmpeg*.
- *socket.io*: Biblioteca que establece enlaces bidireccionales en tiempo real y en comunicación basada por eventos.

Entrada/Salida

En segundo lugar se han definido los diferentes pines GPIO a utilizar y si serán empleados como pines de entrada o de salida:

```

1 //Pines de la Raspberry Pi conectados al vehículo remoto.
2 var gpio2 = new Gpio(2, {mode: Gpio.OUTPUT}),
3     gpio3 = new Gpio(3, {mode: Gpio.OUTPUT}),

```

```
4 gpio17 = new Gpio(17, {mode: Gpio.OUTPUT}),
5 gpio27 = new Gpio(27, {mode: Gpio.OUTPUT});
```

GPIO	Modo	Control
2	OUTPUT	Motores lado izquierdo
3	OUTPUT	Motores lado izquierdo
17	OUTPUT	Motores lado derecho
27	OUTPUT	Motores lado derecho

Tabla 6.1: Configuración establecida para los puertos GPIO.

Si para el vehículo que deseemos programar resultan necesarios más pines tan solo debemos inicializarlos e indicar si van a ser pines de entrada o de salida. Si existen dudas al respecto se puede acceder a la documentación de la biblioteca *pigpio* en el siguiente enlace: <https://www.npmjs.com/package/pigpio>

Sockets

Para la creación del socket basta con la siguiente instrucción, la cual recibe un puerto que utilizará para mantenerse a la escucha:

```
1 var io = require('./node_modules/socket.io').listen(8085, { log:
  false });
```

Con la finalidad de ir capturando los diferentes eventos, se han definido las siguientes funciones para la conexión y desconexión de los clientes:

```
1
2 io.sockets.on('connection', function (socket)
3 {
4
5   //Almacenamiento del número total de clientes conectados.
6   sockets[socket.id] = socket;
7   console.log("Total clientes conectados : ",
8     Object.keys(sockets).length);
9
10  //Envío de un saludo.
11  socket.emit('robotmsg', {msg: "!!!!HOLA!!!!"});
12
13  //Salida de un cliente.
14  socket.on('disconnect', function() {
15    console.log('Bye!');
16    stopStreaming(socket);
17  });
18
19 })
```

Cuando un evento *action* es recibido se activada la función que procesa el comando recibido y activa las salidas correspondientes, la cual establece los pines necesarios a

los valores 1 o 0 según el parámetro establecido. La tabla ?? muestra las diferentes combinaciones de salidas y su acción correspondiente:

Acción	GPIO 2	GPIO 3	GPIO 17	GPIO 27
UP	1	0	1	0
DOWN	0	1	0	1
LEFT	1	0	0	0
RIGHT	0	0	1	0
STOP	0	0	0	0

Tabla 6.2: Combinaciones de salida para los puertos GPIO y su acción correspondiente.

A continuación se muestra el ejemplo desarrollado para la activación de los motores en sentido de giro y dirección según la tabla anterior:

```

1 // Escucha de comandos.
2 socket.on('action', function (data){
3
4     console.log('Comando recibido: ' + data);
5
6     switch(data) {
7         case 'UP':
8             exec_command( 'echo 1 > ' + path + 'gpio2/value' );
9             exec_command( 'echo 0 > ' + path + 'gpio3/value' );
10            exec_command( 'echo 1 > ' + path + 'gpio17/value' );
11            exec_command( 'echo 0 > ' + path + 'gpio27/value' );
12            console.log('UP');
13            break;
14
15        case 'RIGHT':
16            exec_command( 'echo 0 > ' + path + 'gpio2/value' );
17            exec_command( 'echo 0 > ' + path + 'gpio3/value' );
18            exec_command( 'echo 1 > ' + path + 'gpio17/value' );
19            exec_command( 'echo 0 > ' + path + 'gpio27/value' );
20            break;
21
22        case 'LEFT':
23            exec_command( 'echo 1 > ' + path + 'gpio2/value' );
24            exec_command( 'echo 0 > ' + path + 'gpio3/value' );
25            exec_command( 'echo 0 > ' + path + 'gpio17/value' );
26            exec_command( 'echo 0 > ' + path + 'gpio27/value' );
27            break;
28
29        case 'DOWN':
30            exec_command( 'echo 0 > ' + path + 'gpio2/value' );
31            exec_command( 'echo 1 > ' + path + 'gpio3/value' );
32            exec_command( 'echo 0 > ' + path + 'gpio17/value' );
33            exec_command( 'echo 1 > ' + path + 'gpio27/value' );
34            break;
35
36        case 'STOP':

```

```

37     exec_command( 'echo 0 > ' + path + 'gpio2/value' );
38     exec_command( 'echo 0 > ' + path + 'gpio3/value' );
39     exec_command( 'echo 0 > ' + path + 'gpio17/value' );
40     exec_command( 'echo 0 > ' + path + 'gpio27/value' );
41     break;
42   default:
43     console.log('Comando no encontrado');
44   }
45 }
46 )

```

Streaming de vídeo

Para el streaming de vídeo desde el robot hacia el cliente conectado se ha empleado las librerías FFmpeg, haciendo uso de su herramienta de línea de comandos.

```

function startStreaming(socket) //ffmpeg -f video4linux2 -i /dev/video0 -s 300x150 -f
mjpeg pipe:1 -b:v 28k -bufsize 28k
if (running_camera == false)console.log('Starting streaming....');varargs = [" -f", "video4linux2",
ffmpeg_command.on('error', function(err, stdout, stderr)console.log("ffmpegstdout :" + stdout);con
ffmpeg_command.on('close', function(code)console.log('ffmpegexited' + code);running_camera = fals
ffmpeg_command.stderr.on('data', function(data)//console.log('stderr :' + data););
ffmpeg_command.on('end', function()console.log('Finished');running_camera = false);
ffmpeg_command.stdout.on('data', function(data)//console.log('stdout :' + data);varframe = newBu

```

Código de ejemplo completo

```

1 // Start a socket.io server that listens on port 8085.
2 var io = require('./node_modules/socket.io').listen(8085, { log: false
  });
3
4 // Load required modules.
5 var sys = require('util'), exec = require('child_process').exec,
6   path = require('path'), ffmpeg_command, running_camera = false,
7   Gpio = require('pigpio').Gpio;
8
9 var sockets = {};
10
11 // Pin numbers on the Raspberry Pi connected to the car's remote.
12 var gpio2 = new Gpio(2, {mode: Gpio.OUTPUT}),
13   gpio3 = new Gpio(3, {mode: Gpio.OUTPUT}),
14   gpio17 = new Gpio(17, {mode: Gpio.OUTPUT}),
15   gpio27 = new Gpio(27, {mode: Gpio.OUTPUT});
16
17
18 console.log('Waiting connection...');

20 io.sockets.on('connection', function (socket)
21 {
22   sockets[socket.id] = socket;

```

```

24 console.log("Total clients connected : ",
25   Object.keys(sockets).length);
26 socket.emit('robotmsg', {msg: "HELLO!!!"});
27
28 socket.on('disconnect', function() {
29   console.log('Bye!');
30   stopStreaming(socket);
31 });
32
33
34 socket.on('start-stream', function() {
35   startStreaming(socket);
36 });
37
38
39 // Listen for direction messages from the app.
40 socket.on('action', function (data, req, res){
41
42   switch(data) {
43     case 'UP':
44       gpio2.digitalWrite(1);
45       gpio3.digitalWrite(0);
46       gpio17.digitalWrite(1);
47       gpio27.digitalWrite(0);
48       console.log('UP');
49       break;
50     case 'DOWN':
51       gpio2.digitalWrite(0);
52       gpio3.digitalWrite(1);
53       gpio17.digitalWrite(0);
54       gpio27.digitalWrite(1);
55       console.log('DOWN')
56       break;
57     case 'STOP':
58       gpio2.digitalWrite(0);
59       gpio3.digitalWrite(0);
60       gpio17.digitalWrite(0);
61       gpio27.digitalWrite(0);
62       console.log('STOP');
63       break;
64     default:
65       console.log('command not found');
66   }
67
68 });
69 });
70
71
72
73
74 function stopStreaming(socket) {
75   delete sockets[socket.id];
76   // no more sockets, kill the stream
77   if (Object.keys(sockets).length == 0) {

```

```

78     if (ffmpeg_command){
79         ffmpeg_command.kill();
80         running_camera = false;
81         console.log('Stop streaming');
82     }
83 }
84 }
85
86 function startStreaming(socket) {
87     //ffmpeg -f video4linux2 -i /dev/video0 -s 300x150 -f mjpeg pipe:1
88     // -b:v 28k -bufsize 28k
89     if (running_camera == false){
90         console.log('Starting streaming....');
91         var args = ["-f", "video4linux2", "-i", "/dev/video0", "-s",
92             "300x150", "-f", "mjpeg", "pipe:1", "-b:v 28k", "-bufsize 28k"]
93         ffmpeg_command = require('child_process').spawn("ffmpeg", args);
94         running_camera = true
95     }
96     ffmpeg_command.on('error', function(err, stdout, stderr) {
97         console.log("ffmpeg stdout:\n" + stdout);
98         console.log("ffmpeg stderr:\n" + stderr);
99         running_camera = false
100    });
101
102
103    ffmpeg_command.on('close', function (code) {
104        console.log('ffmpeg exited' + code );
105        running_camera = false
106    });
107
108
109    ffmpeg_command.stderr.on('data', function (data) {
110        //console.log('stderr: ' + data);
111    });
112
113    ffmpeg_command.on('end', function() {
114        console.log('Finished');
115        running_camera = false
116    });
117
118    ffmpeg_command.stdout.on('data', function (data) {
119        //console.log('stdout: ' + data);
120        var frame = new Buffer(data).toString('base64');
121        socket.emit('canvas',frame);
122    });
123 }
124 }
```


Capítulo 7

Organización temporal

La planificación general del proyecto siguiendo un modelo SCRUM, empleando para ello el panel de tareas Trello; un gestor de proyectos que permite aplicar una metodología de desarrollo ágil.

El panel se encuentra accesible en el siguiente enlace: <https://trello.com/b/SpIbFI7k/robotui>.

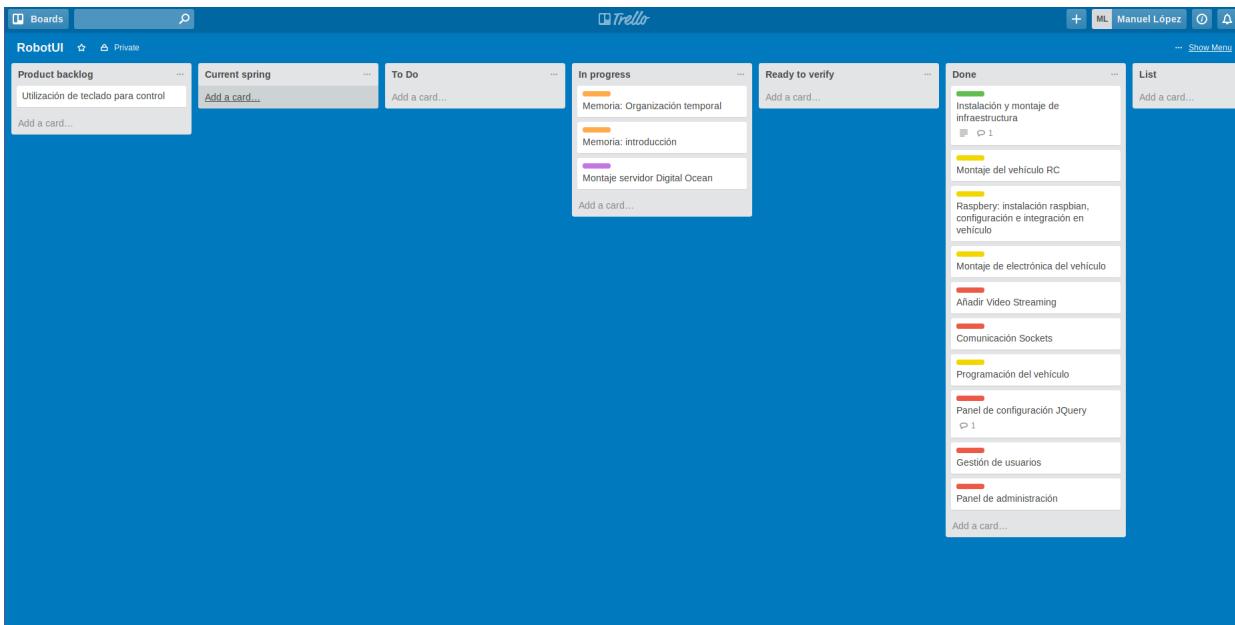


Figura 7.1: Panel de actividades - Trello

Cabe destacar que para el desarrollo de RobotUI ha sido necesario emplear varias herramientas, utilidades y bibliotecas. Algunas de ellas ya habían sido utilizadas en ciertas ocasiones, bien sea en el ámbito estudiantil o profesional. Sin embargo, otras han requerido un periodo de formación previo, en el que se han adquirido los conocimientos necesarios para poder desarrollar el presente proyecto.

La mayor parte del proceso de investigación fue dedicado al estudio de las diferentes tecnologías existentes para la programación web en tiempo real. Todo ello ha implicado

un esfuerzo bastante considerable en el uso, aprendizaje e investigación de las diferentes tecnologías existentes y comprobar su potencial.

Una vez determinadas las diferentes herramientas a utilizar se comenzó con la implementación de la aplicación, siendo seleccionada como herramienta principal el framework Sails.js. Un framework MVC en tiempo real para Node.js, el cual está muy enfocado al propósito de este proyecto.

Finalmente, tras la necesidad de probar la aplicación en un entorno real, se optó por elaborar un robot de pruebas, un pequeño vehículo elaborado con una Raspberry Pi con la finalidad de probar, testear y hacer demostraciones de la aplicación.

La figura [7.2](#) y [7.3](#) muestran una visión de las diferentes tareas desarrolladas para la elaboración del proyecto junto con la descomposición de cada una de ellas:

Los puntos más importantes del proyecto se han dividido en hitos, así como entregas que se definieron en cada reunión que se realizaba con el director del proyecto. También se ha definido una planificación temporal del desarrollo del proyecto mediante un diagrama de Gantt con la duración de las tareas recogidas en el panel de actividades de Trello.

CAPÍTULO 7. ORGANIZACIÓN TEMPORAL

WBS	Nombre	Trabajo
1	▼ RobotUI	135d 3h
1.1	Conocimiento del proyecto	22d
1.2	Planificación y estudio del proyecto	5d
1.3	▼ Análisis de herramientas existentes	17d
1.3.1	Estudio de Node.js	5d
1.3.2	Estudio de Sails.js	5d
1.3.3	Estudio de Socket.io	2d
1.3.4	Códigos y pruebas	3d
1.3.5	Estudio de MongoDB	2d
1.4	▼ Definición de requisitos	5d 6h
1.4.1	Arquitectura BBDD análisis	1d
1.4.2	Diseño de diagrama UML	2d
1.4.3	Requisitos funcionales	1d
1.4.4	Requisitos no funcionales	6h 45min
1.4.5	Reunión de planificación con director del proyecto	1d
1.5	▼ Desarrollo aplicación	30d 4h
1.5.1	Sistema de autenticación - registro	2d
1.5.2	Alta de dispositivos robóticos	2d
1.5.3	Internacionalización	1d
1.5.4	Módulo de mensajes entre usuarios	3d
1.5.5	Implementación de políticas de permisos	4d
1.5.6	Reunión de seguimiento con director del proyecto	3d 3h
1.5.7	▼ Elementos de la interfaz	5d 5h
1.5.7.1	Acciones	1d 1h
1.5.7.2	Sliders	1d
1.5.7.3	Labels	1d 7h
1.5.7.4	Video	1d 4h
1.5.8	▼ Personalización de la interfaz	9d 4h
1.5.8.1	Personalización de acciones	1d 4h
1.5.8.2	Personalización de sliders	2d
1.5.8.3	Personalización de labels	2d
1.5.8.4	Edición de la interfaz	4d
1.6	▼ Módulo de comunicaciones	18d
1.6.1	▼ Conexión cliente - robot	4d 7h
1.6.1.1	Envío de órdenes al robot	2d
1.6.1.2	Captura de vídeo del robot	2d 7h
1.6.2	▼ Conexión cliente - servidor	3d
1.6.2.1	Envío de vídeo capturado al servidor	1d 4h
1.6.2.2	Envío de órdenes lanzadas al servidor	1d 4h
1.6.3	Desarrollo de tests funcionales	8d
1.6.4	Frontend - detalles de estilos	2d

Figura 7.2: Descomposición de las tareas implicadas en el desarrollo del proyecto (Primera Parte).

7.1. PLANIFICACIÓN TEMPORAL DE TAREAS

1.7	▼ Conexión servidor - cliente	4d
1.7.1	Difusión de vídeo a espectadores	2d
1.7.2	Difusión de comandos a espectadores	2d
1.8	▼ Montaje del vehículo	3d
1.8.1	Búsqueda de elementos hardware	2d
1.8.2	Montaje y conexiones	1d
1.9	▼ Programación del vehículo	4d 7h
1.9.1	Gpio	2d
1.9.2	Video streaming	1d 3h
1.9.3	Fase de pruebas	1d 4h
1.10	Despliegue de la aplicación en producción	1d
1.11	▼ Documentación	23d
1.11.1	Memoria	19d
1.11.2	Resumen	2d
1.11.3	Presentación	2d
1.12	Reunión de seguimiento con el director del proyecto	1d

Figura 7.3: Descomposición de las tareas implicadas en el desarrollo del proyecto (Segunda parte).

7.1. Planificación temporal de tareas

A continuación se definirán los diferentes hitos que componen el diagrama de Gantt divididos en las diferentes subtareas principales de cada uno de ellos:

7.1.1. Hito 1: Planificación y análisis

En esta primera etapa de desarrollo del proyecto final de carrera se realizaron los estudios previos necesarios para abordar cualquier proyecto de cierta envergadura. Este hito se descompone en las siguientes tareas principales:

1. Planificación y estudio del proyecto. En esta fase se centró en la elaboración de un documento, a modo borrador, con la idea a desarrollar, objetivos del proyecto y su alcance.
2. Análisis de herramientas existentes, de las tecnologías a implementar, la arquitectura del sistema, las tecnologías de BD, visualización, para la selección de las herramientas más adecuadas para afrontar el desarrollo con garantías y no sea necesaria una “vuelta atrás” por necesidad imperiosa de cambio de tecnología. En definitiva se buscaba una herramienta libre, con un respaldo de una comunidad importante y que resuelva la problemática o necesidad de trabajar con eventos en tiempo real.

7.1.2. Hito 2: Definición de requisitos

Este segundo hito queda dividido en las siguientes etapas:

1. Elaboración de un documento formal con la propuesta de proyecto definiendo sus objetivos y alcance, para la aprobación por parte del director del proyecto.
2. Se definen las clases del sistema, el modelo de la base de datos y la definición de requisitos funcionales y no funcionales.

7.1.3. Hito 3: Comienzo de desarrollo de la aplicación

En este tercer hito, uno de los de mayor magnitud, se comienza con el desarrollo de la aplicación, el cual queda dividido en las siguientes módulos.

1. Implementación del módulo *Usuario* con las funciones de registro, autenticación, configuraciones de idioma, entre otras.
2. Implementación del módulo central de la aplicación *Robot*
3. Implementación del módulo de mensajes.
4. Implementación del módulo de políticas de permisos.

7.1.4. Hito 4: Desarrollo de la aplicación, módulo componentes

En este hito, se trabaja en el desarrollo de los diferentes elementos que compondrán la interfaz de control. Tanto su parte de configuración y personalización. Cada elemento integrante de la interfaz lo denominamos componentes.

En definitiva, el hito queda dividido en el desarrollo de los siguientes componentes:

1. *Acciones*
2. *Sliders*
3. *Labels*
4. *Vídeo*

7.1.5. Hito 5: Desarrollo de la aplicación, módulo interfaz

En este hito, se realiza la elaboración de la interfaz de control. Tanto su parte de configuración y personalización como la de visualización para su control. Este hito quedó dividido en las siguientes etapas:

1. Implementación de la funcionalidad de configuración
2. Implementación de la funcionalidad de control

7.1.6. Hito 6: Desarrollo del módulo de comunicaciones

El presente hito, clasificado como crítico debido a su importancia. Debía realizar la integración de todos los módulos anteriores y dotarlos de la funcionalidad principal para la que han sido diseñados. Estar interconectados entre sí además de la elaboración de los test funcionales.

1. Implementación de la conexión cliente - robot.
2. Implementación de la conexión cliente - servidor.
3. Desarrollo de test funcionales.

7.1.7. Hito 7: Construcción del vehículo de pruebas

Se realiza la construcción y montaje e instalación software del vehículo de pruebas y comprobación de las diferentes conexiones.

7.1.8. Hito 8: Programación del vehículo de pruebas

Se realiza la programación del vehículo y se realizan pruebas de todo el conjunto junto con las correcciones necesarias.

7.1.9. Hito 9: Documentación

Se finaliza la memoria para la revisión por parte del director del proyecto y su posterior impresión. Se prepara la presentación para la defensa ante tribunal.

7.2. Diagrama de Gantt

A continuación se muestra el diagrama de Gantt donde quedan reflejados los diferentes hitos descritos en el punto anterior.

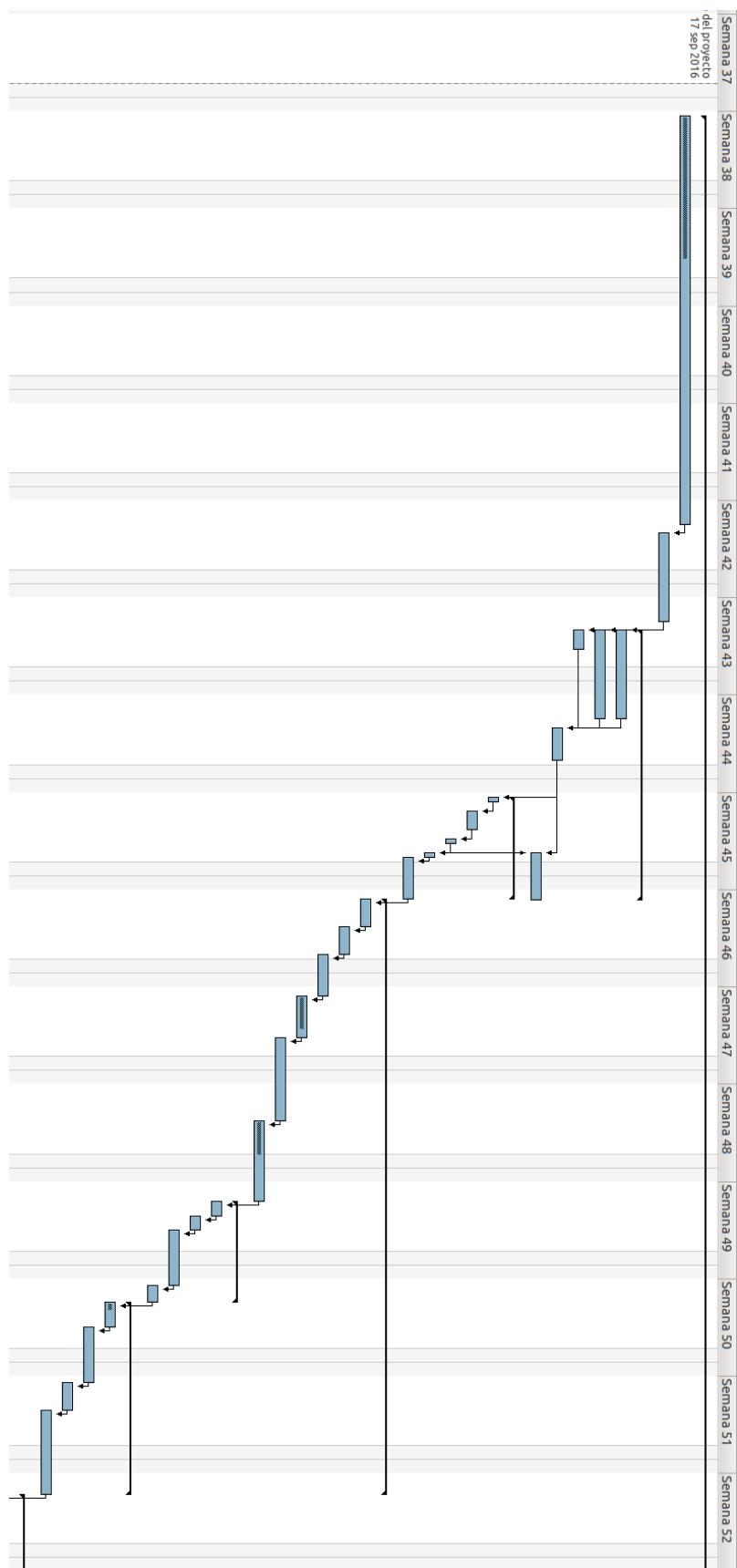


Figura 7.4: Diagrama de Gantt 1. Desarrollo del proyecto.

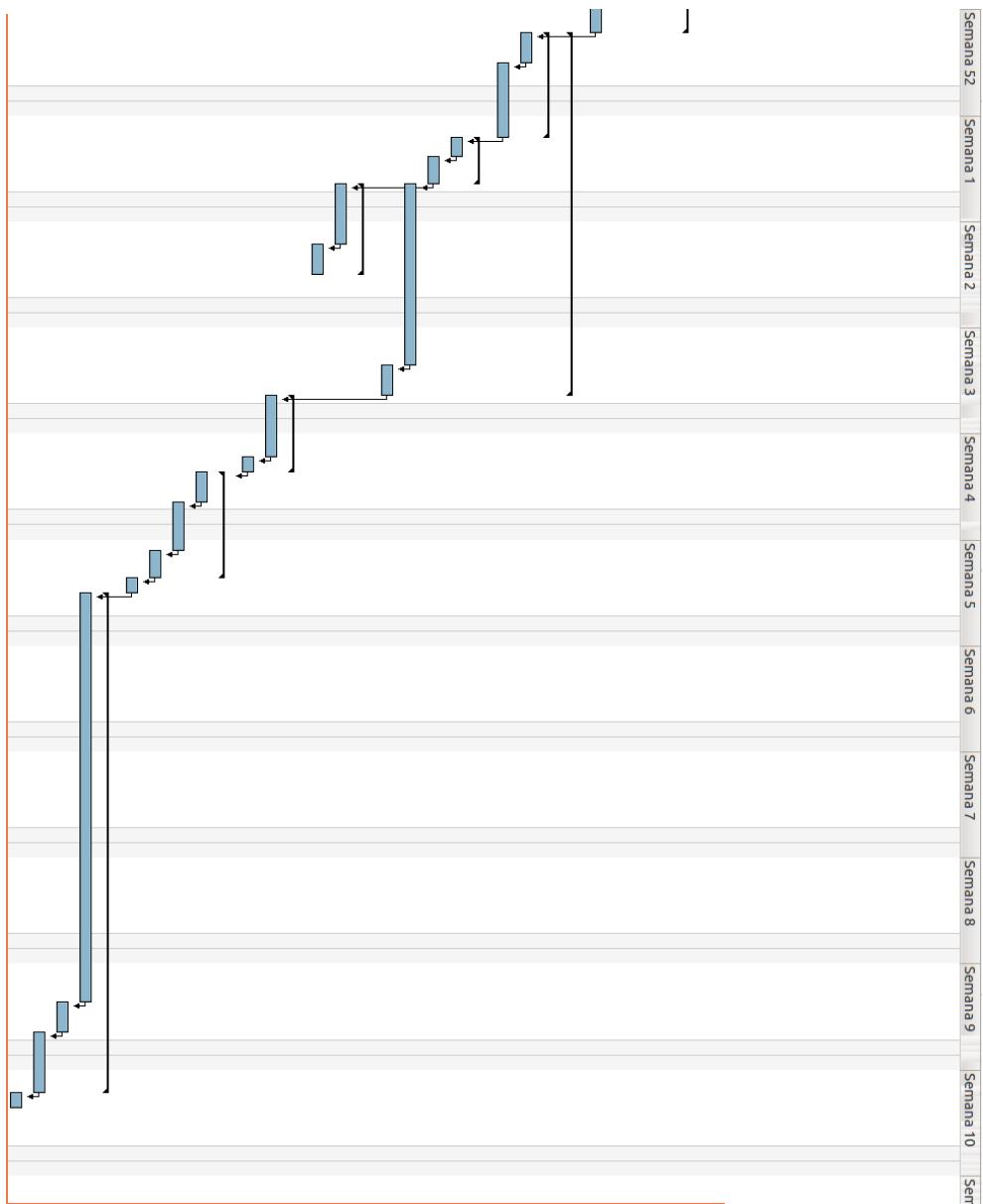


Figura 7.5: Diagrama de Gantt 2. Desarrollo del proyecto.

Capítulo 8

Guía de Usuario

RobotUI es una aplicación web creada con un claro propósito; el de proporcionar a los usuarios un medio donde compartir sus dispositivos robóticos con el resto de usuarios.

Para ello RobotUI proporciona una serie de herramientas para que, aquellas personas que no tengan los suficientes conocimientos de programación, puedan configurar un entorno para el manejo de sus proyectos robóticos y tener posibilidad de compartir sus experiencias con el resto.

La particularidad de RobotUI es que el usuario propietario del robot tiene la posibilidad de permitir el manejo de sus dispositivos robóticos al resto de usuarios que él mismo considere de una manera controlada además de él mismo o, por otra parte, permitir que otros usuarios visualicen, como si de espectadores se tratase, el control que un determinado usuario realiza de un determinado robot. Todo ello en tiempo real.

Por tanto, tras esta breve introducción en el ámbito de la aplicación, en este manual se describen los diferentes paso a realizar para configurar sus dispositivos correctamente en el sistema y abrirlo a toda una comunidad de usuarios. Además de tener abierto el acceso a otros muchos dispositivos de otros usuarios.

8.0.1. Objetivo de esta guía

Esta guía tiene como objetivo la de proporcionar al usuario un soporte de ayuda e iniciación a la utilización de RobotUI.

Esta sección comprende:

- Guía de acceso al código fuente de la aplicación.
- Guía de uso de la aplicación.
- Guía para la puesta en marcha y programación de un robot.

8.0.2. Dirigido a

Esta guía esta dirigida al usuario final del proyecto RobotUI. Tiene la finalidad de proporcionar una guía descriptiva de los procedimientos de creación, configuración y utilización de los diferentes dispositivos en sus dos modalidades disponibles, la de control y la de visualización.

8.0.3. Obtener RobotUI

El código fuente junto con la presente memoria se encuentra disponible en el repositorio GitHub en el enlace <https://github.com/lopi87/SAILS-RobotUI> o usando la herramienta Git, escribiendo en la consola el siguiente comando:

```
1 git clone git@github.com:lopi87/SAILS-RobotUI.git
```

8.1. Uso de RobotUI



Figura 8.1: Página principal RobotUI.



Figura 8.2: Página principal RobotUI.

8.1.1. Registro de usuario

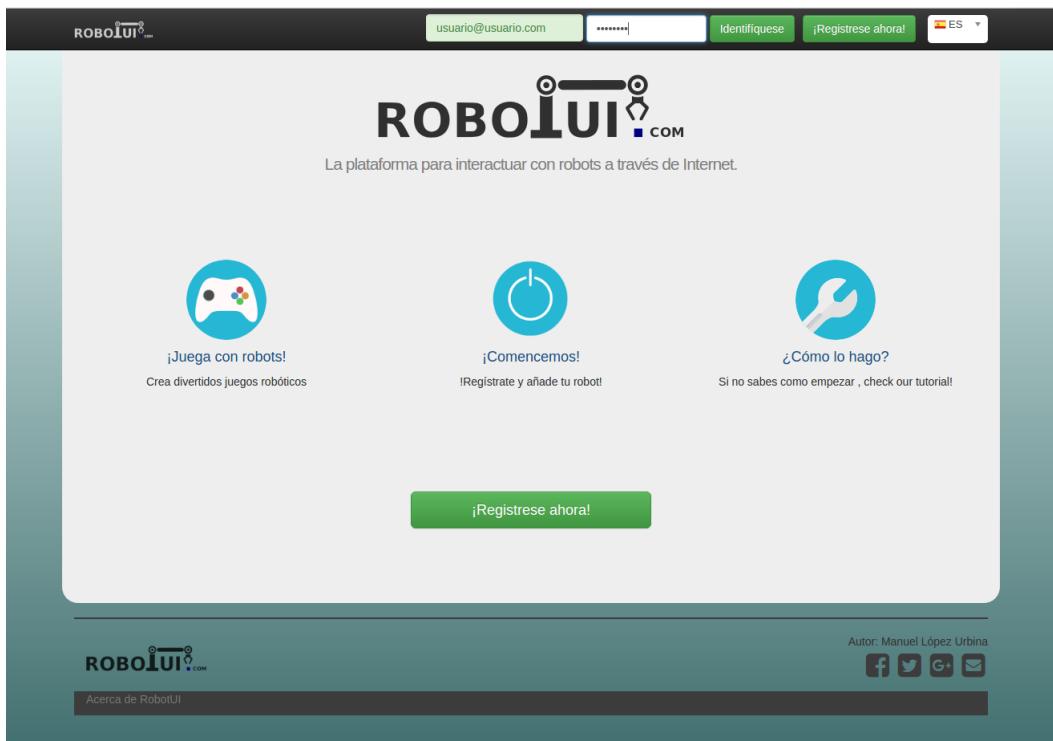


Figura 8.3: Página principal RobotUI.

8.1.2. Creación de un Robot

Para el proceso de creación de un robot hacemos click al menú *Robots* → *Mis robots* accediendo al formulario de creación.

En este caso registraremos nuestro robot de pruebas descrito en el capítulo . Para ello introducimos los siguientes datos:

1. Nombre: nombre de nuestro robot.
2. Dirección IP: dirección donde se encontrará accesible nuestro robot.
3. Puerto: puerto donde se encontrará nuestro robot a la escucha de conexiones.
4. Descripción: campo opcional en el que podemos añadir una descripción de nuestro robot y que será visible al resto de usuarios.
5. Imagen: campo opcional en el que podremos subir una imagen de nuestro robot.
6. Usuarios espectadores: selector en el que podemos especificar qué usuarios del sistema tendrán permisos para visualizar el funcionamiento de nuestro robot.
7. Usuarios controladores: selector para especificar qué usuarios podrán tomar control de nuestro robot.

8. Público: checkboxes para indicar si por el contrario, el robot que abierto a todos los usuarios para su control o abierto para su visualización si marcamos el primero o el segundo checkbox respectivamente y haciendo por tanto inválidos los parámetros de los selectores anteriores.

Una vez llenado los campos como se muestran en la imagen pulsamos en **Crear**.

The screenshot shows the 'Robot' creation form in the RobotUI interface. The form fields are as follows:

- Nombre * :** RC Car
- Dirección IP*:** 80.30.240.XXX
- Puerto*:** 4040
- Descripción:** Vehículo de pruebas RobotUI
- Por favor proporcione un archivo menor que 42 kb:** electric-buggy.jpg (with 'Cambiar' and 'Eliminar' buttons)
- Usuarios espectadores:** Seleccione usuarios
- Usuarios controladores:** Seleccione usuarios
- Público * :** Control: Ver:

At the bottom right of the form is a blue **Crear** button. The footer of the page includes the ROBOTUI logo, links for 'Acerca de RobotUI', 'Autor: Manuel López Urbina', and social media icons for Facebook, Twitter, Google+, and Email.

Figura 8.4: Formulario de creación de un robot.

8.1.3. Configuración de la interfaz

Una vez creado nuestro robot en el sistema ya podemos configurar su interfaz. En la figura 8.5 podemos ver el canvas sobre el que iremos añadiendo los diferentes elementos que conformarán nuestro panel de control.

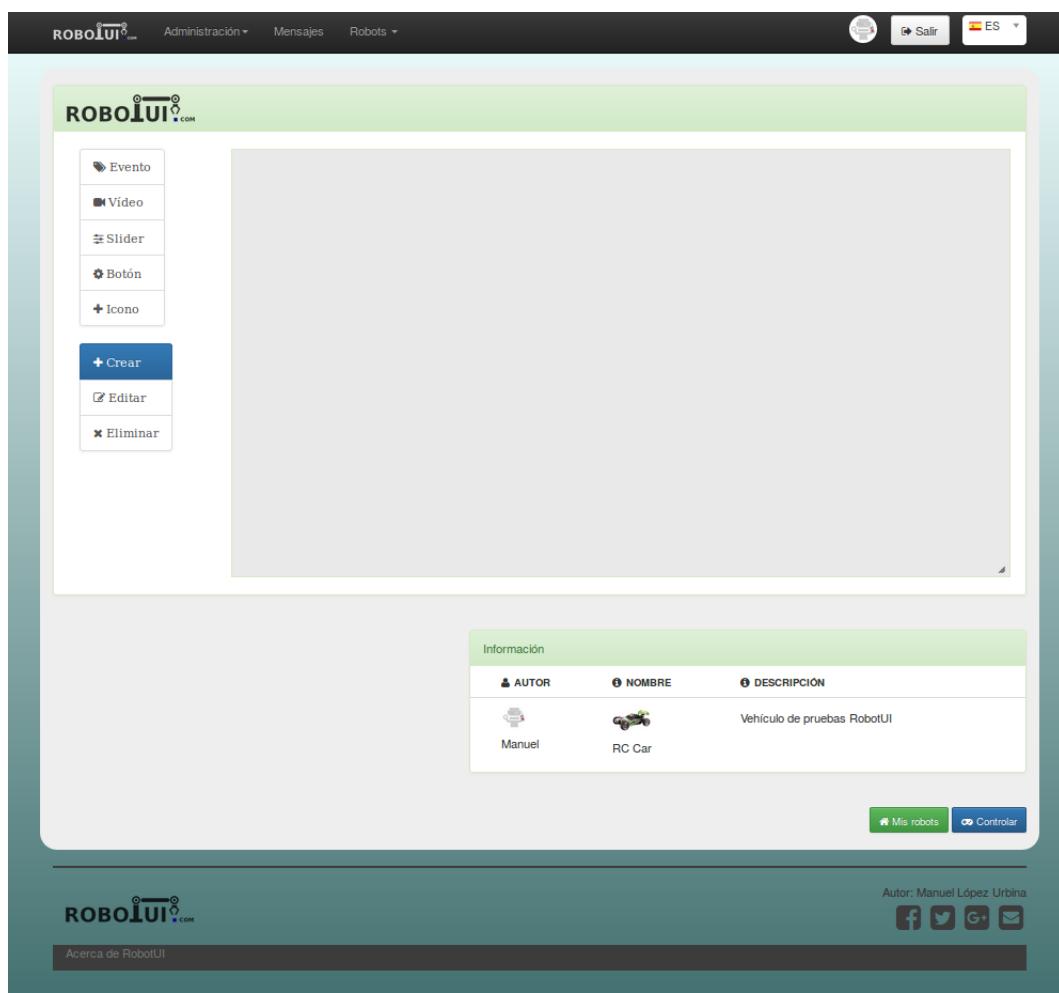


Figura 8.5: Panel de configuración de la interfaz.

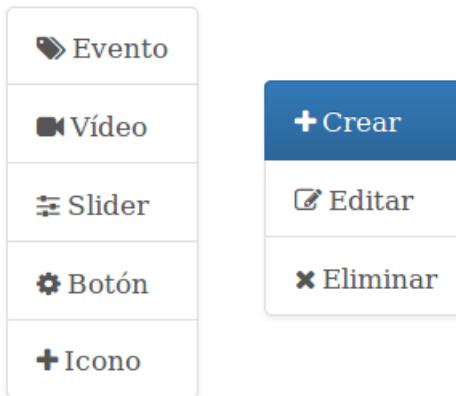


Figura 8.6: Panel de herramientas y panel de acciones para la configuración de la interfaz.

Botón	Significado
 Botón	Apertura del formulario para la creación de un botón.

Tabla 8.1: Elementos de una interfaz.

8.2. Programa tu robot

Capítulo 9

Comentarios finales

9.1. Presupuesto

Descripción	Unidades	Precio € unidad	Total €
Raspberry Pi 3 Modelo B	1	38,70	38,70
Cámara USB alta definición	1	39,90	39,90
Tarjeta de expansión con batería de Litio para Raspberry Pi	1	16,99	16,99
Lipo batería (3.7v, 600mAh Lipo)	1	13,99	13,99
Indicador Tester de baterías Lipo	1	8,90	8,90
Cargador baterías Lipo	1	21,90	21,90
Chasis vehículo Radiocontrol	1	54	54
Droplet DigitalOcean	2 meses	5/mes	10
Horas de programación	350 horas	50/hora	17500

Total bruto: 17745,38 €

I.V.A. %: 21 %

Total presupuesto: 21471,91 €

9.2. Conclusiones

La elaboración de este proyecto ha resultado muy gratificante a nivel personal. Uno de los motivos principales ha sido la necesidad de trabajar en numerosas áreas de conocimiento entre las que encontramos, por un lado la programación web, haciendo uso del framework Sails js, junto con el empleo de una base de datos no relacional. Todo ello combinado con la robótica. Algunas de las plataformas mencionadas eran desconocidas al inicio del desarrollo de proyecto y han sido adquiridas tras una amplia labor de investigación.

Entre los elementos desarrollados se destaca:

- La elaboración de un vehículo de pruebas haciendo uso de una Raspberry Pi 3 Modelo B
- Aprendizaje a la utilización del framework Sails.js
- Aprendizaje al trabajo con eventos en tiempo real mediante el empleo de Web-Sockets, tecnología nunca utilizada por mí hasta la fecha
- Empleo de una base de datos no relacional como Mongo DB.
- Transmisión de gran cantidad de datos entre cliente servidor y servidor cliente. Streaming de vídeo y emisión de comandos entre otros datos.

Pienso que el resultado final del proyecto es ideal para aquellas personas aficionadas a la robótica y programación proporcionando una herramienta sea utilizable por la gran comunidad poseedora del de cualquier proyecto robótico y que puedan compartirlo con el resto del mundo.

Una vez presentado podré continuar añadiendo mejoras y muchas cosas que tengo pensadas y que, posiblemente, se realicen para el proyecto del máster de Ingeniería de Sistemas y Computación que me encuentro realizando en la actualidad.

9.3. Mejoras futuras

La aplicación puede mejorarse en diversos aspectos. A continuación, se citan algunas de las mejoras que pueden llevarse a cabo:

- Incorporación de advertencias acústicas tras la detección de una señal de tráfico.
- Mejora del diseño de la interfaz gráfica.
- Permitir la definición de las teclas de control del teclado e incorporación de dispositivos tales como gamepads o joysticks.
- Elaborar un generador de código para la exportación en los dispositivos robóticos, reduciendo por tanto las labores de programación.

Anexos

Anexos con las instrucciones para la instalación de todos los componentes software empleados en el desarrollo del proyecto.

.1. Instalación de Node.js

Instalación de los requisitos:

```
1 sudo apt-get install python-software-properties python g++ make
```

Si está utilizando Ubuntu 12.10, necesitará hacer los siguiente:

```
1 sudo apt-get install software-properties-common
```

Añadimos el repositorio:

```
1 sudo add-apt-repository ppa:chris-lea/node.js
```

Actualizamos la lista de paquetes:

```
1 sudo apt-get update
```

Instalación de Node.js:

```
1 sudo apt-get install nodejs
```

.2. Instalación Sails.js

Esta guía proporciona las pautas necesarias para la configuración de un entorno de trabajo para el desarrollo de aplicaciones Sails. Esta guía no cubre la instalación de en un entorno de producción.

.2.1. Prerrequisitos

Partiendo de que se encuentra Node correctamente instalado en una máquina con Ubuntu 16.04.2 LTS. Ubuntu es una plataforma muy popular y utilizada en el desarrollo de Sails.js, al igual que otros sistemas operativos basados en Unix, como Mac OS X. La instalación es relativamente fácil y existe multitud de información gracias a su amplia comunidad de desarrolladores.

- Una máquina con Ubuntu 16.04.2 LTS
 - Node.js instalado.

2.2. Instalación

A continuación detallaremos los pasos para la instalación de Sails. Lo primero que haremos es instalar Sails haciendo uso de npm, el gestor de paquetes que viene con el propio Node. Para ello, lo que vamos a hacer es ir directamente a la terminal y e introducir lo siguiente:

```
1 sudo npm install sails -g
```

La opción `-g`, que significa global, lo que hace es instalar Sails a nivel global, la cual nos permitirá acceder a las funcionalidades de Sails que emplearemos para la creación de nuestros proyectos.

Es posible que necesite permisos de administrador para instalar Sails a nivel global.

Para comprobar que la instalación se realizó correctamente, crearemos un proyecto inicial y levantaremos el servidor. Para ello introduciremos en la terminal:

```
1  sails new my_first_app
```

Cambie de directorio (cd) al nuevo directorio creado, en el cual nos aseguraremos de que Sails está correctamente instalado. Para ello arrancaremos el servidor y comprobaremos en nuestro navegador en localhost 1337 (localhost: 1337) que Sails está funcionando correctamente.

```
1 cd my_fisrt_app  
2 sails lift
```

Tras introducir en nuestro navegador `localhost: 1337` debemos obtener el siguiente resultado:

```
manuel@manuel-ThinkPad-E560:~/Escritorio/my_first_app:ruby-2.3.1$ sails lift

info: Starting app...

Info:          .....
Info:          Sails      <|      .....
Info:          v0.12.11
Info:          / \ \
Info:          |   |
Info:          :   ==|/-\'
Info:          -----
Info:          -----
Info: Server lifted in  /home/manuel/Escritorio/my_first_app
Info: To see your app, visit http://localhost:1337
Info: To shut down Sails, press <CTRL> + C at any time.

debug: -----
debug: :: Sat Feb 04 2017 17:48:42 GMT+0100 (CET)

debug: Environment : development
debug: Port       : 1337
debug: -----
```

Figura 1: Iniciando Sails¹.

Y en nuestro navegador:

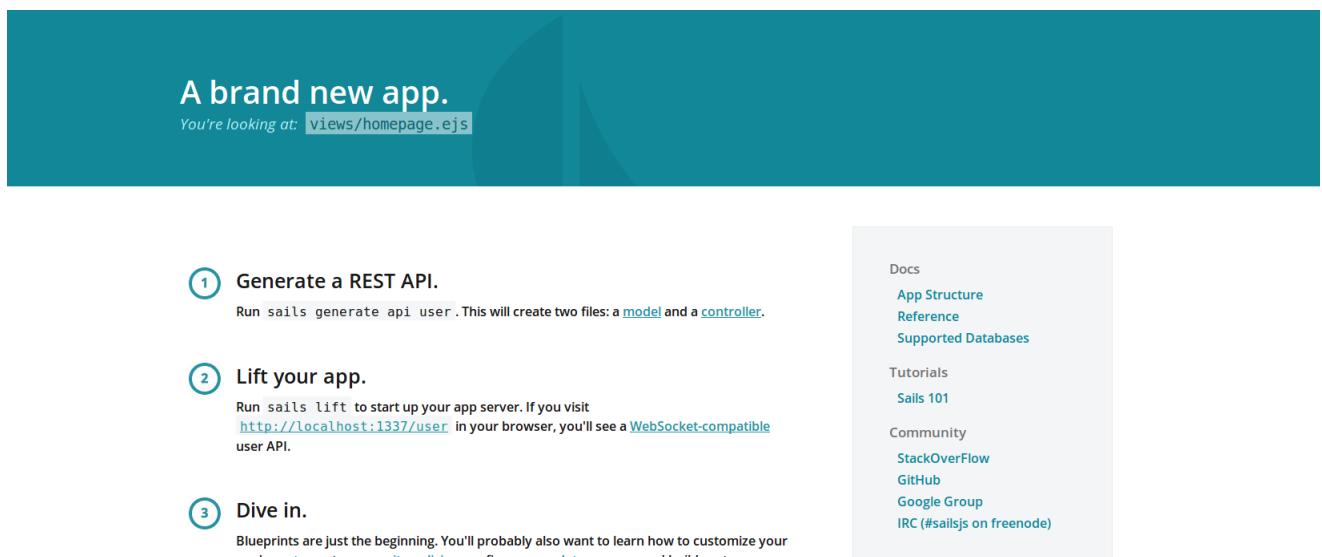


Figura 2: Sails en funcionamiento ².

.3. Instalación de MongoDB

MongoDB es una base de datos libre y de código abierto NoSQL utilizada comúnmente en aplicaciones web modernas. Esta guía le ayudará a configurar MongoDB en su máquina para un entorno de aplicación de producción.

.3.1. Prerrequisitos

Para seguir esta guía es necesario:

- Una máquina con Ubuntu 14.04.
- Un usuario con permisos de administrador (no root).

.3.2. Instalación

MongoDB ya está incluido en los repositorios de paquetes de Ubuntu, pero el repositorio oficial MongoDB proporciona la versión más actualizada y es la forma recomendada de instalar el software. Ubuntu garantiza la autenticidad de los paquetes de software al verificar que están firmados con las claves GPG, por lo que primero tenemos que importar la clave para el repositorio oficial de MongoDB.

Para ello, ejecute:

```
1 sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv  
7FOCEB10
```

Después de importar con éxito la clave, verá lo siguiente:

```
1 Gpg: Número total procesado: 1  
2 Gpg: importado: 1 (RSA: 1)
```

A continuación, tenemos que actualizar los detalles del repositorio de MongoDB para que APT sepa de dónde descargar los paquetes.

Introduzca el siguiente comando para crear un archivo de lista para MongoDB.

```
1 echo "deb http://repo.mongodb.org/apt/ubuntu \"\$(lsb_release  
-sc)\"/mongodb-org/3.0 multiverse" | sudo tee  
/etc/apt/sources.list.d/mongodb-org-3.0.list
```

Después de agregar los detalles del repositorio, necesitamos actualizar la lista de paquetes.

```
1 sudo apt-get update
```

Ahora podemos instalar el propio paquete MongoDB.

```
1 sudo apt-get install -y mongodb-org
```

Este comando instalará varios paquetes que contengan la última versión estable de MongoDB junto con útiles herramientas de administración para el servidor MongoDB.

Después de la instalación del paquete, MongoDB se iniciará automáticamente. Puede comprobarlo ejecutando el siguiente comando.

```
1 service mongodb status
```

Si MongoDB se está ejecutando, verá una salida como la siguiente (con un ID de proceso diferente).

```
1 mongodb.service - An object/document-oriented database  
2   Loaded: loaded (/lib/systemd/system/mongodb.service; enabled;  
3     vendor preset: enabled)  
3   Active: active (running) since s b 2017-02-04 13:07:01 CET; 11h ago  
4     Docs: man:mongod(1)  
5   Main PID: 807 (mongod)  
6     Tasks: 10  
7     Memory: 84.8M  
8       CPU: 4min 7.494s  
9     CGroup: /system.slice/mongodb.service  
10           - 807 /usr/bin/mongod --config /etc/mongodb.conf
```

También puede detener, iniciar y reiniciar MongoDB utilizando los siguientes comandos

```
1 service mongodb stop  
2 service mongodb start
```

.4. Instalación del control de versiones Git

Para seguir esta guía es necesario disponer de una máquina con Ubuntu 14.04.

La forma más sencilla de tener Git instalado y configurado para su utilización es mediante el uso de los repositorios predeterminados de Ubuntu. Este es el método más rápido, pero, por contra puede ser que la versión disponible no sea la más reciente. Si necesita la última versión, deberá seguir los pasos para compilar Git desde el origen.

Si no necesitamos dispoer de la última versión podemos instalarlo desde el repositorio de Ubuntu. Para ello introducimos los siguientes comandos en una terminal:

```
1 sudo apt-get update  
2 sudo apt-get install git
```

Esto descargará e instalará Git en el sistema. A continuación se describe los pasos para su configuración.

.4.1. Configuración

Ahora que tiene git instalado, necesita hacer algunas cosas para que los mensajes de confirmación que se generarán para usted contendrán su información correcta.

La forma más sencilla de hacerlo es a través del comando *git config* proporcionando nuestro nombre y dirección de correo electrónico. Esto es debido a que Git incorpora esta información en cada commit que hacemos. Por ejemplo, si nuestro nombre es *RobotUI* y nuestro email *email@robotui.com*, los comandos de configuración serían los siguientes:

```
1 git config --global user.name "RobotUI"  
2 git config --global user.email "email@robotui.com"
```

Finalmente podemos comprobar todos los valores de configuración establecidos escribiendo:

```
1 git config --list
```

Existen muchas más opciones configurables pero estos son los dos esenciales necesarios.

.5. Despliegue de una aplicación Sails

Primeramente, debemos definir el entorno de producción para nuestra aplicación Sails. Para ello editamos el fichero */config/env/production.js* introduciendo los siguientes datos:

- Nombre de nuestra conexión de la base de datos la cual hemos definido en nuestro archivo *config/connections.js*
- Dirección IP de nuestro servidor.
- Puerto por el que queremos correr nuestra aplicación.

Un ejemplo de configuración es el siguiente:

```

1 /**
2  * Production environment settings
3 *
4  * This file can include shared settings for a production environment,
5  * such as API keys or remote database passwords. If you're using
6  * a version control solution for your Sails app, this file will
7  * be committed to your repository unless you add it to your .gitignore
8  * file. If your repository will be publicly viewable, don't add
9  * any private information to this file!
10 *
11 */
12
13 module.exports = {
14
15 /*********************************************
16   * Set the default database connection for models in the production *
17   * environment (see config/connections.js and config/models.js )  *
18  *****/
19
20  models: {
21    connection: 'MongodbServer'
22  },
23
24 /*********************************************
25   * Set the port in the production environment to 80 *
26  *****/
27
28  port: 1337,
29
30 /*********************************************
31   * Set the log level in production environment to "silent" *
32  *****/
33
34  log: {
35    level: "silent"
36  }
37
38  proxyHost: '46.101.102.33',
39  proxyPort: 1337
40
41 };

```

En el servidor de producción se ha utilizado Nginx³ con la siguiente configuración:

³Nginx es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico (IMAP/POP3)

```

1 server {
2   listen      80;
3   server_name 46.101.102.33;
4
5
6   listen 443 ssl;
7
8   ssl_certificate /etc/nginx/ssl/nginx.crt;
9   ssl_certificate_key /etc/nginx/ssl/nginx.key;
10
11  location / {
12    proxy_pass          http://46.101.102.33:1337/;
13
14    proxy_http_version 1.1;
15    proxy_set_header Upgrade $http_upgrade;
16    proxy_set_header Connection "upgrade";
17    proxy_set_header Host $host;
18
19  }
20
21 }
22 }
```

Para arrancar la aplicación, se ha hecho uso del gestor de procesos Pm2, se introduce la siguiente orden:

```
1 sudo pm2 start app.js -x -- --prod
```

Para monitorizar la aplicación, Pm2 incorpora una herramienta de gestión de procesos. Para visualizarlo introducimos en una terminal:

```
1 sudo pm2 monit
```

Obteniendo un resultado similar al siguiente:

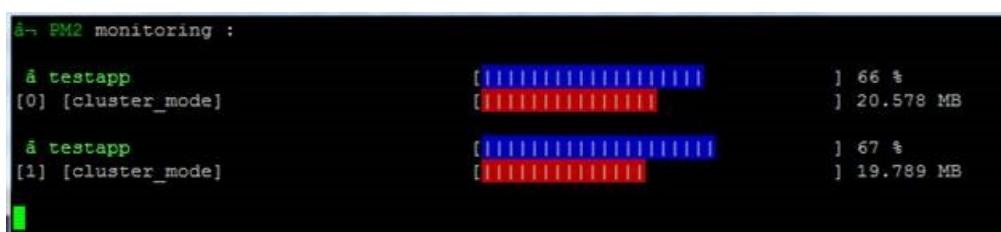


Figura 3: Utilización del gestor de procesos Pm2 en el entorno de producción.

Bibliografía

- [1] Beginner's Guide to Getting Started with Sails.js <https://www.codementor.io/codforgeek/how-to-setup-sailsjs-tutorial-beginners-du107nl5i>
- [2] Douglas Crockford. *JavaScript: The Good Parts*. O'Reilly, 2008.
- [3] Digital Ocean. Digital Ocean. <https://www.digitalocean.com/>.
- [4] Git Hub. Git Hub. <https://github.com>.
- [5] Marc Harter T.J. Holowaychuk Nathan Rajlich Mike Cantelon, Alex R. Young. *Node.js in Action*. Manning Publications, 2017.
- [6] Irl Nathan Mike McNeil. *Sails.js in Action*. Manning Publications, 2017.
- [7] Métrica v3. Métrica v3. <http://www.csi.map.es/csi/metrica3>.
- [8] Irl Nathan. Activityoverlord, an application to learn sails.js. <https://github.com/irlnathan/activityoverlord>.
- [9] JJ Merelo Pablo Hinojoda. *Aprende Git: ... y, de camino, GitHub*. 2015.
- [10] Rohit Rai. *Socket.IO Real-Time Web Application Development*. Packt, 2013.
- [11] Sails.js. Sails.js | Realtime MVC Framework for Node.js. <http://sailsjs.com/>.
- [12] Wikibooks. The Book of LaTeX. <http://en.wikibooks.org/wiki/LaTeX>.
- [13] Stefan Kottwitz. *LaTeX Beginner's Guide*. Packt Publishing, 2011.
- [14] Official documentation. Sails JS Documentation . <https://sailsjs.com/documentation/reference>.
- [15] Frantisek Korbel. *FFmpeg Basics: Multimedia handling with a fast audio and video encoder*. CreateSpace, 2015.
- [16] Andrew Lombardi. *WebSocket: Lightweight Client-Server Communications*. O'Reilly Media, 2012.
- [17] Sandro Pasquali. *Deploying Node.js*. Packt Publishing, 2015.
- [18] Eben Upton, Gareth Halfacree. *Raspberry Pi User Guide*. <http://www.cs.unca.edu/~bruce/Fall14/360/RPiUsersGuide.pdf>.

- [19] Kristina Chodorow. *MongoDB: The Definitive Guide, 2nd Edition*. O'Reilly Media, 2013.
- [20] NoSQL vs SQL: SQL is the new NoNoSQL <http://www.nosql-vs-sql.com/>
- [21] Digital Ocean Understanding SQL And NoSQL Databases And Different Database Models <https://www.digitalocean.com/community/tutorials/understanding-sql-and-nosql-databases-and-different-database-models>
endthebibliography

GNU Documentation Free License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page

as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.

APÉNDICE . GNU DOCUMENTATION FREE LICENSE

- I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document

within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

APÉNDICE . GNU DOCUMENTATION FREE LICENSE

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with ... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.