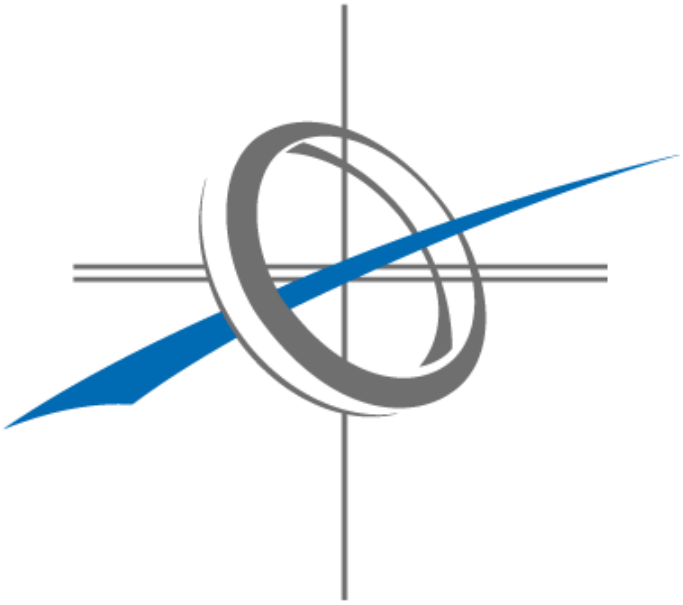
Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos



Grado en Ingeniería del Software

Curso 2023/24

Sistema de cronometraje para tramos WRC a escala

Autor: Pedro Javier Miranda Tejada

Tutor: José Eugenio Naranjo Hernández

Resumen

Este proyecto recoge el desarrollo de un Sistema informático para la organización, cronometraje y clasificación de Tramos de Rally, enfocado a vehículos a escala 1:10.

Este proyecto surge como una búsqueda de mejora y modernización de un sistema llevado a cabo por entusiastas del WRC a escala. El principal objetivo es mejorar la organización de este tipo de eventos. Para ello se ha desarrollado un sistema lo más automatizado posible, ya que el sistema anterior requería mucha interacción por parte de los administradores. Se conocen las plataformas utilizadas en el sistema anterior, pero se desconoce su implementación, por lo que se realizó un estudio de funcionalidad utilizando ingeniería inversa y se inició el desarrollo desde cero utilizando plataformas de implementación más actuales.

Si bien el sistema fue desarrollado para vehículos a escala, también se podría considerar su implementación para vehículos de tamaño completo, ofreciendo un costo más accesible que los sistemas comerciales que existen para ellos.

El sistema de cronometraje consta de un portal web desde el que se monitoriza el comportamiento del sistema, así como de varios dispositivos hardware que se encargan de interactuar con los vehículos y los usuarios.

Abstract

This project encompasses the development of a computer system for the organization, timing, and classification of Rally stages, focused on 1:10 scale vehicles.

This project emerged from a quest to improve and modernize a system managed by WRC enthusiasts on a scale. The main objective is to enhance the organization of this type of event. To this end, a highly automated system has been developed, as the previous system required extensive interaction from the administrators. Although the platforms used in the previous system are known, their implementation is not, so a functionality study using reverse engineering was conducted, and the development was started from scratch using more current implementation platforms.

While the system was developed for scale vehicles, its implementation could also be considered for full-size vehicles, offering a more affordable cost compared to existing commercial systems.

The timing system consists of a web portal from which the system's behavior is monitored, as well as several hardware devices that interact with the vehicles and users.

Tabla de contenido

Ilustraciones

Glosario

A continuación, se explican algunos términos para facilitar la lectura de esta memoria.

RC: «**Radiocontrol**, es la técnica que permite el control de un objeto a distancia y de manera inalámbrica mediante una emisora de control remoto.» [1]

Transpondedor: transpondedor o transponder, Aparato que emite una señal en una frecuencia determinada cuando lo estimula otra señal externa apropiada. [2]

RFID: «**identificación por radiofrecuencia** (del inglés *Radio Frequency Identification*) es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, tarjetas o transpondedores RFID. Las tecnologías RFID se agrupan dentro de las denominadas Auto ID (automatic identification, o identificación automática).» [3]

GB: **Gigabyte**, 1024 millones de Bytes, 1 Byte son 8 bits, bit unidad mínima de información

RAM: Memoria de Acceso Aleatorio, del inglés **Random Acess Memory**

SBC: «**Single-board computer** es una computadora completa construida sobre una sola placa de circuito, con microprocesador(es), memoria, entrada/salida (E/S) y otras características requeridas de una computadora funcional. Las computadoras de placa única se fabrican comúnmente como

sistemas de demostración o desarrollo, para sistemas educativos o para su uso como controladores de sistemas embebidos. Muchos tipos de ordenadores domésticos o portátiles integran todas sus funciones en una sola placa de circuito impreso.» [4]

ARM: «**ARM**, anteriormente **Advanced RISC Machine**, originalmente **Acorn RISC Machines**, es una arquitectura RISC (*Reduced Instruction Set Computer*, «Ordenador con Conjunto Reducido de Instrucciones») de 32 bits y, con la llegada de su versión V8-A, también de 64 Bits, desarrollada por ARM Holdings» [5]

big.LITTLE: « "ARM Big.LITTLE Technology" es una arquitectura de procesamiento heterogénea que utiliza hasta tres tipos de procesadores. Los procesadores "LITTLE" están diseñados para la máxima eficiencia energética, mientras que los procesadores "Big" están diseñados para proporcionar un rendimiento de cómputo eficiente y sostenido.» traducido de ARM [6]

API: «API o **Application Programming Interface**, que en español quiere decir Interfaz de Programación de Aplicaciones, es un conjunto de funciones y procedimientos que permite integrar sistemas, permitiendo que sus funcionalidades puedan ser reutilizadas por otras aplicaciones o software. » [7]

Framework: «(informática) Conjunto estandarizado de criterios, prácticos, conceptos y herramientas para abordar una problemática particular.» [8]

ORM: «**Mapeo relacional de objetos** u ORM (también O/RM, sigla en inglés de **Object-Relational Mapping**, es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional» [9]

ODM: **Object Document Mappers**, «MongoDB proporciona una serie de bibliotecas similares a ORM, y nuestra comunidad ¡Y los socios también lo han hecho! A veces se denominan ODM (Object Document Mappers), ya que MongoDB no es un sistema de gestión de bases de datos relacionales. Sin embargo, existen para resolver el mismo problema que los ORM y la terminología se puede usar indistintamente.» Traducido de [10]

WebSockets: «es un protocolo estándar que permite que un navegador web o una aplicación cliente y una aplicación de servidor web utilice una conexión dúplex para comunicarse.**»** [11]

DOM: «(**Document Object Model**, en español Modelo de Objetos del Documento) es una API definida para representar e interactuar con cualquier documento HTML o XML.**»** [12]

SPA: «(Una **single-page application** (**SPA**), o aplicación de página única, es una aplicación web o un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios, como si fuera una aplicación de escritorio.**»** [13]

UART: «(**universal asynchronous receiver / transmitter**, por sus siglas en inglés) define un protocolo o un conjunto de normas para el intercambio de datos en serie entre dos dispositivos. UART es sumamente simple y utiliza solo dos hilos entre el transmisor y el receptor para transmitir y recibir en ambas direcciones.**»** [14]

Tiempo Unix: cantidad de segundos transcurridos desde la medianoche del 1 de enero de 1970.

1 Introducción

Este proyecto se centra en desarrollar un sistema que tiene como finalidad cronometrar tramos de rally, el cual incluye la gestión en un portal web para el registro de las pruebas, los tiempos de los corredores, así como permitir registrar sus coches.

Además, este sistema deberá de disponer de una vista pública para consultar los resultados de las pruebas sin tener que registrarse previamente, para así poder compartirlos.

1.1 Objetivos

En las fases iniciales del desarrollo se han definido unas metas que se han considerado razonables para el proyecto. Estas metas se describirán a continuación.

1.1.1 Generales

El presente trabajo pretende desarrollar un método eficaz para cronometrar un tramo de Rally a escala e identificar los pilotos que lo realizan, basándonos en las funcionalidades de un sistema previamente existente creado por un grupo de aficionados al Rally RC (Crono2.0).

Se pretende realizar mejoras en dicha implementación, debido a que algunos participantes u organizadores, han visto que hay ciertas incomodidades en el sistema previo las cuales son las siguientes:

- Identificación vehículo – piloto, saber que coche corresponde a cada piloto en todo momento

- Visualización de los resultados en dispositivos móviles o de escritorio

- Asignación de penalizaciones

1.1.2 Específicos

Al margen de lo enunciado anteriormente, el presente proyecto propone la implementación de unos objetivos más concretos, definiendo con mayor precisión los objetivos generales.

- Mejorar la identificación de vehículo – piloto:
    - En el sistema antiguo, la identificación debe hacerse manualmente haciendo clic con el ratón del ordenador de control.
    - Pretendo implementar una automatización que lea una etiqueta RFID la cual portara el conductor en todo momento y debe pasar por un lector al inicio de la etapa y otro al final de la etapa.
- Automatización de detección de situaciones de adelantamiento:
    - En el sistema antiguo, el piloto debe estar en seguimiento visual de un director de carrera y un comisario, si un piloto adelanta a otro, el director de carrera cambiaba de posición el cursor del ratón en el ordenador de control
    - Gracias a la mejora anterior si dos pilotos están en la situación de adelantamiento, al pasar por la línea de meta el orden de escaneo del RFID será diferente al que se

hizo al inicio de la etapa, entonces el sistema podrá saber quién está delante y asignar automáticamente las mediciones a quien corresponda.

- Desarrollo de un sistema web de registro y cronometrado. que dispone de las siguientes funcionalidades:
    - Acceso como usuario Invitado
        - Listado de resultados, este es uno de los puntos a reforzar en del sistema anterior, puesto que el refresco de los datos interfería con la visualización de estos ya que revertía la vista web al scroll inicial.
    - Acceso como usuario Piloto
        - Acceso a las funciones anteriormente mencionadas
        - Registro en el sistema
        - Registro de vehículos personales
        - Listado de pruebas
        - inscripción a las pruebas
    - Acceso como Administrador director de Carrera
        - Acceso a las funciones anteriormente mencionadas
        - Registro de pruebas
        - Asignación de Roles
        - Asignación de penalizaciones
- Registro de usuarios y tiempos en una base de datos.
    - Dotar al sistema de una base de datos independiente de la aplicación web y los dispositivos físicos, para que en caso de necesitar ser modificado haya una persistencia de los resultados y los datos de los usuarios.
- Implementación en Python y MicroPython de las siguientes funcionalidades en dispositivos portátiles:
    - Comunicación de microcontroladores con una aplicación web.
    - Captura de paso de vehículo mediante ultrasonidos, para el cronometraje de la etapa.
    - Identificación vehículo – piloto mediante tarjetas RFID.

2 Sistemas similares

2.1 Sistemas similares a escala

Los sistemas de cronometraje para vehículos a escala ya existen desde hace muchísimo tiempo y las variantes comerciales analizadas a continuación ofrecen rendimiento más que suficiente para las competiciones, pero este trabajo se enfoca en encontrar una manera económica de obtener resultados similares.

2.1.1 Desarrollo previo basado en Arduino

Desarrollo previo personal como prueba de concepto y prueba de eficacia de medición por ultrasonidos, consta de un dispositivo capaz de capturar el paso de un vehículo y mandar por puerto serie Bluetooth el tiempo desde que pasa la primera vez hasta la siguiente. Únicamente posee un sensor, por lo tanto, salida y meta deben ser la misma puerta de paso.

Se observo en pruebas sobre terreno, la siguiente situación, si el suelo es muy irregular, debido a los ecos producidos, el sistema no es capaz de permanecer en estado de espera ya que se ve influenciado por falsas mediciones.

2.1.2 Crono2.0

implementación propia de Rally Rc Madrid, sistema en el cual está inspirado este proyecto, consta de un puente de captura de paso el cual utiliza fotocélulas de garaje para detectar los vehículos , internamente están conectadas a un ratón de ordenador y cuando el vehículo pasa activan el “clic” del puntero , esto se complementa con una interfaz web , implementada con el lenguaje PHP alojada en WordPress , en la cual un director de carrera debe posicionar correctamente el puntero usando otro ratón conectado al ordenador de control. [15]

2.1.3 LapMonitor

Sistema disponible de manera comercial, consta de un dispositivo receptor infrarrojo y de un transpondedor infrarrojo en cada vehículo.

Cuando el transpondedor es detectado por primera vez por el receptor este comienza a contar el tiempo, tras esto cuando se detecta otra vez el transpondedor detiene el conteo y envía el tiempo resultante a una aplicación que se conecta al receptor mediante bluetooth. la interfaz de control está disponible como aplicación para smartphones y como web. [16]

Ilustración 1 : LapMonitor

2.1.4 Mylaps RC4 para RC

Ilustración 2 : Mylaps rc4

Sistema de medición profesional ampliamente utilizado por clubs de modelismo.

funciona mediante utilización de pequeños módulos transpondedores y de antenas receptoras por radiofrecuencia, al igual que LapMonitor detecta el paso de los transpondedores y envía los tiempos a la estación de control, implica un mayor costo que los anteriores y requiere de una instalación mediante obra en caso de circuitos de asfalto, ya que se entierra bajo la pista una antena receptora. [17]

2.2 Sistemas similares para uso en Tamaño real

Para competiciones a tamaño real existen bastantes soluciones, al ser distancias mayores apare de utilizar sensores de paso como los vehículos a escala, pueden utilizar receptores GPS y otro tipo de mediciones más comunes.

2.2.1 Mylaps X2 para uso general

Ilustración 3 : Mylaps X2

La empresa Mylaps está ampliamente reconocida, y a la vez que ofrece los dispositivos para vehículos a escala también dispone de diversos sistemas para vehículos de tamaño real. En la ilustración adyacente se muestra a el sistema X2 el cual funciona de manera similar al RC4 a diferencia de que los transpondedores y las antenas son mucho mayores en tamaño y además no está limitado a la automoción ya que puede usarse en diversas modalidades deportivas. [18]

2.2.2 Rabbit Rally

Ilustración 4 : Rabbit Rally

Consiste en un receptor GPS en el coche el cual permite conocer su posición en todo momento y a través de esta se determina los pasos por cada “checkpoint” o zonas necesarias, Se complementa con las medidas de un odómetro (mide la rotación de las ruedas del coche) implementado mediante sondas magnéticas. [19]

3 Tecnologías Disponibles

A continuación, se desglosarán las distintas tecnologías que pueden estar implicadas en el desarrollo del proyecto y se decidirá cuáles serán las finalmente utilizadas.

3.1 Hardware

Componentes físicos implicados en el proyecto, ya sean los sensores, tanto como los dispositivos en los cuales se ejecutarán los procesos de medida

3.1.1 Detección e identificación de objetos

Técnicas las cuales permitirán la detección del coche al cruzar las puetas de paso o estaciones, analizamos los siguientes tipos de sensores

3.1.1.1 Distancia por Infrarrojos

Los sensores de distancia IR funcionan a través del principio de triangulación, miden la distancia en función del ángulo del haz de luz reflejado. Es común que en proyectos pequeños de electrónica se utilicen los sensores del fabricante Sharp, estos sensores tienen la desventaja de que pueden ser influenciados por la luz solar ya que esta también es emitida en el espectro infrarrojo, por lo tanto, su uso queda mayormente delegado a interiores. [20]

3.1.1.2 Distancia por Ultrasonidos

Miden distancia utilizando ondas de sonido, concretamente en las frecuencias ultrasónicas las cuales son las que están por encima de los 20 kilohercios, estas ondas no son percibidas como audibles. La medición se realiza emitiendo un puso y conociendo la velocidad a la que viaja el sonido, calculando cuanto tarda en rebotar esa onda y regresar al punto de emisión.

los sensores ultrasónicos son utilizados en aplicaciones donde los sensores infrarrojos no pueden funcionar, como por ejemplo la medición de nivel de líquidos, o la detección de objetos claros, ya que estos absorben la luz. también se debemos saber que estos sensores no pueden operar en el vacío ya que el sonido necesita un medio de propagación y respecto a la dificultad de detección de objetos también pueden verse afectados por la absorción de la onda de sonido, debido a la forma del objeto a detectar o de no ser capaz de reflejar esa onda debido a su tamaño. [21, 22]

3.1.1.3 Radio frecuencia RFID

Se puede distinguir 2 tipos en función la alimentación energética de las tarjetas o módulos de identificación:

- Activo: Se realiza utilizando emisores de radiofrecuencia alimentado desde el propio emisor y detectando estas emisiones con antenas en el receptor. Esta tecnología tiene un alto rango de distancias de funcionamiento en función de la potencia de la señal emitida y de la capacidad de recepción de las antenas.
- Pasivo: Se utilizan tags o tarjetas que no disponen de alimentación, funcionan median inducción magnética, recibiendo primeramente un campo electromagnético el cual provee de energía al tag, en el momento que ese se activa envía al mismo emisor del campo una señal con la información que contiene el tag o tarjeta , el alcance de los sistemas pasivos se limita a unos pocos centímetros

3.1.2 Single-Board Computer (SBC)

A continuación, muestro una comparativa de 4 SBC, para ser instalada como nuestra estación principal o puesto de salida, todas disponen de conectividad inalámbrica wifi y bluetooth

Tabla 1 : comparativa SBC [23, 24, 25, 26]

Modelo	Arquitectura de la CPU	Núcleos	Frecuencia	RAM
Raspberry Pi 4	ARM Cortex-A72 (4 núcleos)	4	1.8GHz - 2.1GHz	1GB, 2GB, 4GB o 8GB
Raspberry Pi Zero 2 W	ARM Cortex-A53 (4 núcleos)	4	1GHz	512MB
Orange Pi 4 LTS	ARM Cortex-A53 (4 núcleos) + ARM Cortex-A72 (2 núcleos)	6	1.8GHz	4GB
Odroid N2	ARM Cortex-A53 (2 núcleos) + ARM Cortex-A72 (4 núcleos)	6	1.8GHz - 2.4GHz	4GB

En la tabla primeramente observamos que las 4 opciones investigadas disponen de arquitectura ARM, pero cada una de ellas de distintos tipos de núcleo siendo los A53 menos potentes que los A72, haciendo uso de la arquitectura ARM big.LITTLE , además la mayoría disponen de 4GB de Memoria RAM por lo tanto esta será la elección , Tras observar las opciones barajadas nos vamos a decantar por la Raspberry Pi 4 de 4gb de RAM ya que ofrece un rendimiento adecuado.

3.1.3 MCU (Microcontroller Unit) y SoC (System on a Chip)

Para la reducción de costes se usará un MCU para la estación de llegada puesto que solo tendrá que controlar el paso de los vehículos y comunicárselo a la estación de partida ,he agrupado las distintas opciones en la siguiente tabla

Tabla 2 : comparativa MCU [27, 28, 29, 30]

Modelo	Microcontrolador	Arquitectura CPU	Frecuencia Reloj	RAM	Wifi	Bluetooth	Nº pines GPIO	Interfaces
Arduino Uno R3	ATmega328P	AVR	16MHz	2KB	No	No	20	UART, I2C, SPI
Raspberry Pi Pico	RP2040	ARM big. LITTLE	133MHz	264KB	No	No	26	UART, SPI, I2C, PIO, PWM, ADC
Espressif ESP32	Tensilica Xtensa LX6	Dual-core Xtensa	160MHz	520KB	Sí	Sí	34-38	UART, SPI, I2C, PWM, DAC, ADC, RMT, CAN
Espressif ESP8266	Tensilica Xtensa LX106	RISC	80MHz	160KB	Sí	No	17-19	UART, SPI, I2C
STMicroelectronics STM32	ARM Cortex-M desde M3 hasta M33	ARM RISC	Hasta 480MHz	Hasta 2MB	No	Sí	Variante por modelo	UART, SPI, I2C, CAN, PWM, ADC, DAC

Debido a que se va a necesitar una conexión inalámbrica wifi a la estación principal, solo podemos barajar los modelos de Espressif y ya que necesitamos de una mayor velocidad además de un corto tiempo de respuesta se va a elegir el ESP32 ya que tiene mayor frecuencia para poder operar más rápidamente y más memoria RAM para poder utilizar un software más complejo.

3.2 Software

3.2.1 Entorno de Desarrollo

Como principal Entorno de va a utilizar Visual Studio Code ya que ofrece una lata flexibilidad gracias a sus muchas extensiones , en él se va a desarrollar la aplicación web , el api para la base de datos y el software de la estación principal mediante SSH , además de servir como apoyo para la realización del software de la estación de llegada (ESP32) este último utiliza además como entorno Thonny Python ide ya que es necesario para acceder a los contenidos del MCU y actualizar su código fuente

3.2.2 Frameworks 1, focalizados en Python

en frases tempranas del desarrollo se decidió usar Python por tener una curva de aprendizaje sencilla y una sintaxis más cómoda en cuanto a los lenguajes que conocía previamente , ya que no es necesario utilizar el carácter ';' como delimitador de instrucciones y '{' , '}' como delimitadores de bloque de código . para ello se pensó en usar alguno de los siguientes framework

3.2.2.1 FastAPI:

Conocido por su velocidad y rendimiento excepcionales. Utiliza la librería Pydantic para la validación de datos y la serialización, lo que garantiza una manipulación de datos más segura y eficiente. Además, Ofrece soporte integrado para WebSockets, lo que lo hace ideal para aplicaciones en tiempo real. Pero como contra partida este framework está pensado para API backend [31]

3.2.2.2 Django:

Es un framework web de alto nivel que incluye muchas características integradas, como autenticación, ORM, y administración de URL, lo que facilita el desarrollo de aplicaciones web complejas. Ofrece una capa de abstracción de base de datos poderosa y fácil de usar a través de su ORM, lo que simplifica las operaciones de base de datos. Tiene incorporadas medidas de seguridad como la protección contra CSRF e inyección SQL, lo que lo hace ideal para proyectos que requieren altos estándares de seguridad. [32]

3.2.2.3 Pyramid:

Es conocido por su enfoque minimalista y su gran flexibilidad, lo que permite a los desarrolladores elegir y combinar las herramientas y bibliotecas que mejor se adapten a sus necesidades. Es altamente escalable y adecuado para proyectos de cualquier tamaño, desde pequeñas aplicaciones hasta grandes sistemas empresariales. Dispone de soporte para una variedad de bases de datos a través de SQLAlchemy, lo que permite a los desarrolladores elegir la base de datos que mejor se adapte a sus necesidades. [33]

3.2.2.4 Flask:

Es un MicroFramework minimalista que sigue el principio de "hazlo simple pero no más simple". Es ideal para proyectos pequeños y aplicaciones simples. A pesar de su tamaño pequeño, Flask ofrece una gran flexibilidad, permitiendo a los desarrolladores integrar fácilmente diferentes extensiones y bibliotecas según sea necesario. Tiene una comunidad activa y vibrante que contribuye con una amplia gama de extensiones y herramientas para una variedad de casos de uso. [34]

3.2.3 Frameworks 2, focalización en Javascript

Tras este análisis se empezó a utilizar Flask, pero a medida que avanzaba el desarrollo, al encontrarme con el problema de que Flask utiliza plantillas estáticas para la interfaz de usuario, me veía forzado a usar JavaScript cuando quería introducir funciones de vista dinámica , puesto que las funcionalidades de vista dinámica que ofrece Flask obligan a recargar la página web entera para mostrar cambios y JavaScript puede interactuar dinámicamente con el DOM solo afectando a los componentes necesarios.

Por lo que llegado a este punto se valoró cambiar de framework y de lenguaje, por lo que se decidió usar JavaScript, conserva la curva de aprendizaje sencilla y el tipado débil, y también es flexible con la sintaxis ya que el carácter ';' delimitador de instrucción no es obligatorio utilizarlo, pero se acepta cuando aparece, como contra parte si son necesarios los delimitadores de bloque. Como parte del servidor JavaScript, se utilizará el entorno de ejecución Node.js, los Frameworks valorados anteriormente no disponen de variantes para JavaScript por lo tanto se valoró estas otras opciones:

3.2.3.1 Express.js

Es un framework de aplicaciones web Node.js mínimo y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles, cuenta con miles de métodos de utilidad HTTP y middleware a su disposición, la creación de una API sólida es rápida y sencilla. [35]

3.2.3.2 Angular.js

Desde Google se presenta Angular.js como conjunto de herramientas y entorno para construir interfaces web de usuario, muy completo está enfocado en ofrecer todo lo que el desarrollador necesita, tiene muchas características integradas. con respecto a la sintaxis separa claramente el HTML de la lógica de negocio de Javascript ya que cada componente utiliza archivos independientes [36, 37]

3.2.3.3 React.js

Por parte de Facebook se nos presenta React, centrado en el minimalismo y muy enfocado en la interfaz de usuario, al contrario que angular este framework no dispone de tantas integraciones , con lo cual dependiendo de las funcionalidades deseadas se le van agregando dependencias , pero esto hace que sea más adaptable si no necesitas todas las funciones , respecto a su manejo integra el HTML y el Javascript todo junto en archivos denominados jsx. [36, 38]

3.2.3.4 Vue.js

Vue no es ofrecido por una gran empresa como los dos anteriores, se encuentra mantenido por la comunidad, se encontraría a medio camino entre los dos mencionados anteriormente dispone de características intermedias de ambos, el núcleo es sencillo como el de React también necesita dependencias , pero no tantas . con respecto a los componentes utiliza ficheros Vue con una estructura que separa la lógica del HTML siendo más cercano a Angular.js. [36, 39]

3.2.4 Frameworks 3, los finalmente seleccionados.

Para la realización del servidor y el api se decidió utilizar Express.js casi sin valorar otras opciones ya que solo se va a realizar un api sencillo y no se necesita un framework complejo.

Para la realización del portal web se va a utilizar React y como herramienta complementaria se va a utilizar Vite «Vite (palabra en francés para "rápido", pronunciado como /vit/ , como "veet") es una herramienta de compilación que tiene como objetivo proporcionar una experiencia de desarrollo más rápida y ágil para proyectos web modernos.» esta herramienta permite a su vez utilizar un entorno "hot reload" el cual muestra los cambios según se trabaja sin necesidad de compilar todo el proyecto cada vez, también esta optimizado para recargar solo el componente afectado mediante las interacciones del usuario y no obliga a recargar la página web completa, a su vez cuando se pasa a fase de producción permite empaquetar los componentes de React.js en únicamente 3 ficheros para su fácil despliegue. [40]

3.2.5 Software de Periféricos

Para los dispositivos periféricos utilizados que serán Raspberry pi 4b para la estación de salida, y esp32 para controlar el paso al final del recorrido, se utilizara Python y MicroPython respectivamente ya que ofrecen soporte nativo y diversas librerías específicas.

3.2.6 Tecnologías para aplicaciones en la nube

Se ha barajado el uso de las 3 más conocidas y con gran infraestructura

3.2.6.1 Google Cloud Platform

Se Busco inicialmente información acerca de un plan estudiantil pero no está disponible para esta institución, se intentó entrar también usando el periodo de prueba gratuito de 90 días , pero requiere de realizar una inversión, por lo que se pasó a mirar en otras plataformas. [41]

3.2.6.2 Amazon Web Services (AWS)

Una de las grandes alternativas de Saas (Software as a Service, Software como servicio), el plan de prueba AWS también requiere de inversión, pero si dispone de plan educativo valido, pero está limitado a solo vista de cursos y no da acceso a la consola de servicios. [42, 43]

3.2.6.3 Microsoft Azure

Gracias a los acuerdos de la UPM con Microsoft se dispone de licencia educativa y uso para estudiantes, ofrece una gran cantidad de servicios y distintas maneras de despliegue.

En el desarrollo inicial se utilizó, "Azure web apps" y "Azure Cosmos DB for MongoDB" ya que disponía de CI/CD (integración continua y despliegue continuo) junto con GitHub (plataforma de repositorios en la cual está alojado el código fuente) [44] , pero esta implementación tenía el inconveniente de que no permitía hacer pruebas en local y requería del despliegue cada vez que se quería comprobar su funcionamiento y tardaba demasiado tiempo . [45, 46, 47]

3.2.7 Base de Datos

Al principio utilizamos MongoDB ya que ofrecía la posibilidad de empezar rápidamente , pero con forme paso el tiempo se vio que utilizar una base de datos NoSql para este proyecto estaba complicando el backend más de lo debido , por lo que se pasó a una base de datos Relacional SQL ya que los datos que va a tener no van a variar de estructura y son conocidos , dentro de las plataformas de gestores de bases de datos se valoró usar MySQL o PostgreSQL , finalmente me

decante por la segunda opción ya que PostgreSQL ofrece un rango de tipos más variado que MySQL , también a la hora de afrontar el manejo ambas bases de datos van a resultar indiferentes puesto que se va a utilizar un ORM [48, 49]

4 Metodología

Primero se realiza un estudio de las funciones básicas del sistema a observar Crono 2.0, posteriormente se utiliza un desarrollo ágil basado en prototipos, empezando por implementar y probar funcionalidades sencillas y posteriormente integrándolas en un sistema más complejo

5 Desarrollo

5.1 Breve análisis inicial

Siguiendo la metodología descrita anteriormente el estudio de características del sistema Crono 2.0 [15] es el siguiente:

5.1.1 Gestión de acceso

Cada rol dispone de direcciones web diferentes para entrar, y no se mezcla nunca el tráfico de usuarios con diferente rol. Si un usuario desea entrar en la dirección que no le corresponde simplemente no puede acceder ya que hay un Login separado para cada dirección. A su vez la vista de invitado no es accesible de manera interactiva sino a través de conocer su dirección especifica, esta se les proporciona a través de un servicio externo de mensajería.

5.1.2 Detección de vehículos

se utilizan fotocélulas de garaje en un mismo punto de paso tanto como entrada y salida, no pudiendo usarse simultáneamente

5.1.3 Diferenciación de los vehículos

Se realiza de forma manual cambiando la posición de un mouse inalámbrico conectado al ordenador de control, las fotocélulas anterior mente mencionadas están conectadas a otro mouse desencadenando las pulsaciones.

5.1.4 Almacenaje de tiempos

no se dispone de acceso para poder analizar. Pero se conoce que el sistema está montado en la plataforma WordPress y utiliza ficheros PHP.

5.1.5 Gestión de puntaciones y penalizaciones

Se conoce que el sistema dispone de esta característica, pero no se dispone de acceso o información para poder analizar.

5.2 Prototipo V1

Sistema compuesto por un Arduino uno r3, un módulo de pantalla LCD de 16 caracteres por 2 filas, un módulo bluetooth hc06, y un módulo ultrasónico hy-srf05.

5.2.1 Medición de distancia

Se creo una versión básica de un cronometro de paso, centrándonos en la detección del vehículo por ultrasonidos mediante el sensor hy-srf05 cuenta con 5 pines, VCC y GND, los cuales son para la alimentación, Trigger (disparo) y Echo (eco) , los cuales sirven para enviar las señales y recibirlas , por último, el pin Out (Modo), permite seleccionar el modo de funcionamiento , utilizando un solo pin para enviar y recibir señales o utilizando 2 pines. Ya que disponemos de pines suficientes en la placa Arduino R3 , se optó por usar el modo de 2 pines , el funcionamiento de la comunicación es el siguiente , en el instante inicial se inicializan los pines de disparo y eco a nivel TTL bajo , para compensar la medición se envía un pulso de nivel alto de 10 microsegundos y posteriormente se vuelve a poner ese pin en bajo , en ese momento el sensor emite un tren de 8 pulsos de ultrasonido a 40khz (Ilustración 5 : Pulsos , este haz cónico tiene un ángulo de 15 grados , tras ello si el la onda ultrasónica rebota en alguna superficie el sensor emite un pulso que varía de 100 micro segundos a 25 milisegundos en función de la duración del tiempo que viaje el sonido , en caso de no colisionar el pulso echo es de 30 milisegundos, sabiendo la duración del pulso podremos saber cuánto tiempo viajo la onda de ultrasonidos por el medio , su vez conociendo la velocidad del sonido en el aire y el tiempo que tardo la onda en propagarse podemos conocer la distancia , el cálculo final sería el siguiente , sabiendo que la velocidad del sonido es aproximadamente 343,2 m/s y convirtiéndola a cm/µs tenemos 0,03432 cm/µs o $1/29{,}13$ cm/µs y como hay que tener en cuenta que el tiempo dado es de la ida al objeto y su rebote hay que dividir el tiempo obtenido entre 2 , por lo tanto el cálculo final quedaría así : $distancia = \frac{tiempo\ \mu s}{29{,}13*2}$ [52, 51]

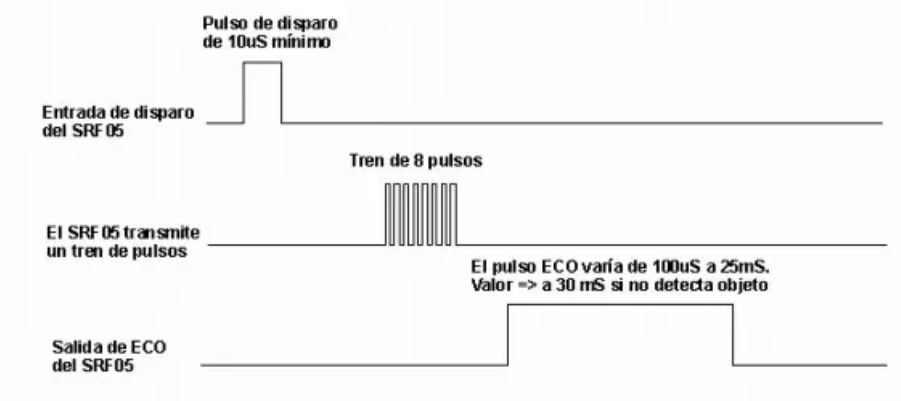


Ilustración 5 : Pulsos [50, 51]

5.2.2 Medición de tiempo y funcionamiento

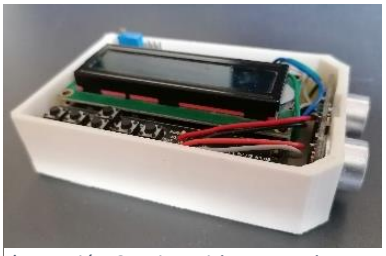


Ilustración 6 : Dispositivo prototipo v1

El comportamiento global del software del dispositivo, funciona de la siguiente manera , todo comienza con el encendido de la placa , la cual primero inicializa el sensor, el Bluetooth y la pantalla , una vez carga , muestra por pantalla un menú descriptivo en dos líneas con los ajustes de margen de error y el modo de operación (continuo para vueltas a un circuito , o con pausa , para medida de tiempo en tramo abierto , por limitación de tener un sensor solamente, la salida y meta deben estar en el mismo lugar) , una vez que se establecen las opciones deseadas , al pulsar el botón "select" se entra en el bucle de ejecución principal , aquí es donde se disparan las mediciones del ultrasonido hasta que detecta un objeto , si el objeto está más cerca de los márgenes de error configurados se inicia el contador de tiempo y se para cuándo se vuelve a detectar otro cambio. Cuando se obtiene el tiempo de vuelta / tramo, seguidamente se envía mediante comunicación UART al módulo bluetooth y se muestra por pantalla, el módulo bluetooth automáticamente si tiene algún dispositivo vinculado envía los datos que recibió, lo anteriormente descrito se puede observar en el siguiente diagrama de flujo.

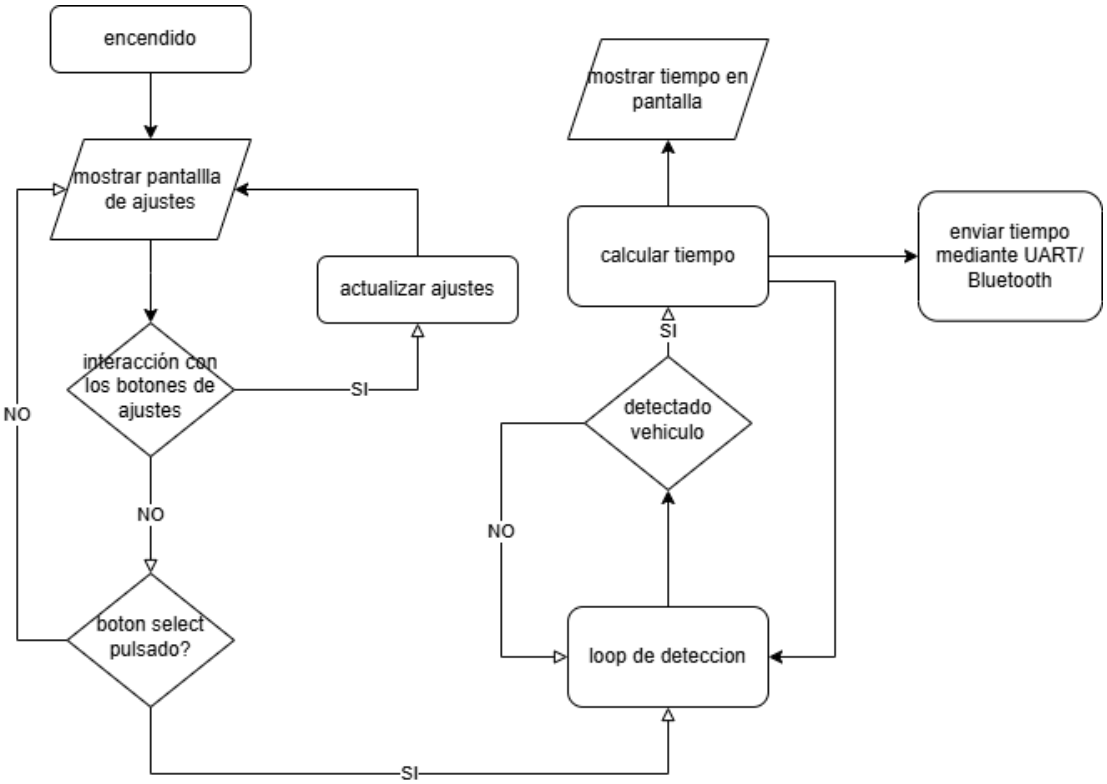


Ilustración 7 : Flujo v1

5.2.3 Análisis posterior

Tras haberlo usado realmente y haber probado distintas condiciones, se observan los siguientes inconvenientes, en suelos rugosos o con piedras el eco resultante se ve alterado por estas superficies, ya que el sensor se encuentra demasiado cerca del suelo si únicamente se usa el dispositivo sin soportes.

La pantalla acoplada al mismo a veces es difícil de leer desde lejos, para futuros prototipos es necesario separar la interfaz del dispositivo que realiza as mediciones

5.3 Prototipo V2

Analizando las necesidades se decide implementar primero el Api y el servidor que se va a dedicar a recibir las peticiones y un principio de interfaz de usuario web para poder ver si se envían los datos correctamente a la aplicación. se realizó el siguiente diagrama de casos de uso correspondiente al comportamiento de la aplicación web como punto de partida.

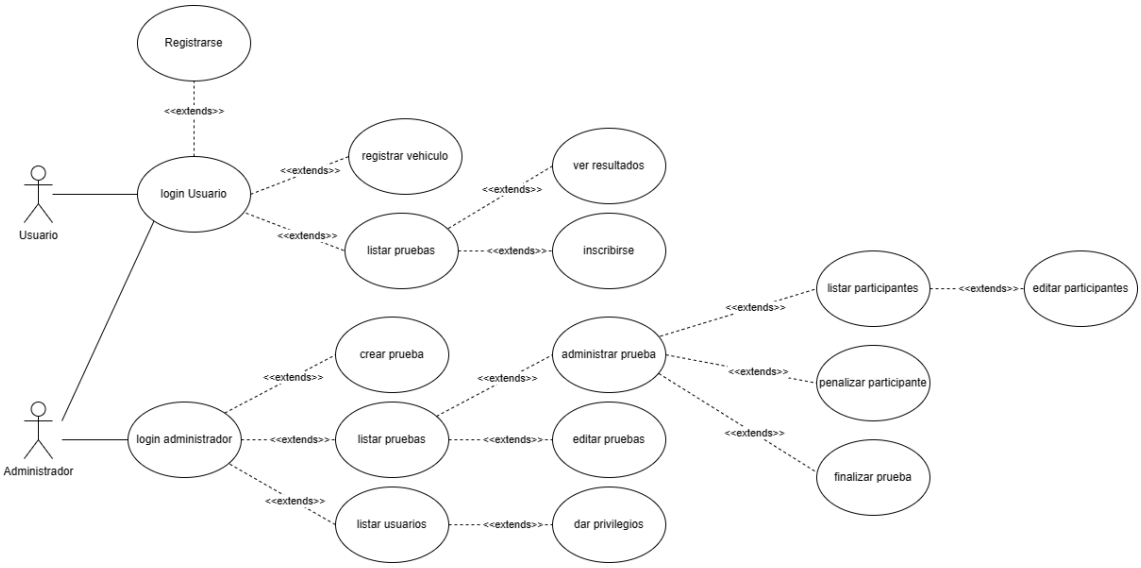


Ilustración 8 : Casos de uso v2

Se puede observar que para poder agilizar el desarrollo solo se definieron 2 actores en el sistema ya que se unificaron los roles de invitado y usuario, como aquellos que podrían acceder libremente a la aplicación, en este diagrama observe que bastantes casos de uso no están en el lugar adecuado, como por ejemplo “ver resultados”, según el diagrama esas obligado a regístrate para poder ver tus tiempos. De todas formas, no se llegó a esos niveles en la implementación inicial, solo se desarrolló el Login de usuarios, y el registro en el sistema, además, hay casos de uso que se repiten y solo deberían estar en un lugar como por ejemplo “listar pruebas” , ambos actores deberían poder ver todas las pruebas de la misma manera. Para el prototipo siguiente este diagrama se descartó.

Para empezar la implementación se utilizó la guía de app Service [46] disponible en la documentación de Azure para probar una versión inicial del portal y para comprobar la dificultad de la implementación tanto como la disponibilidad que ofrecía Azure como plataforma de host en la nube.

5.3.1 Casos de uso implementados

A partir de esta base se les añadió las siguientes funcionalidades correspondientes a algunos casos de uso anteriores.

- Registrarse, permite a los usuarios darse de alta en la aplicación
- Login usuario, permite a los usuarios sin privilegios de administrador iniciar sesión en el sistema, tras ello eran redirigidos a una vista especifica con hipervínculos a las demás pantallas.
- Login administrador, permite a los administradores entrar en el sistema y acceder a un panel de control específico para ellos, en el que se listaban los usuarios dados de alta en el sistema y podían modificar ese listado.

en este momento fue cuando empezaron a surgir problemas con la presentación y el comportamiento de la vista dinámica ya que algunas funciones no se podían abordar correctamente, debido a que Flask utiliza plantillas estáticas y obligan a renderizar toda la página web cada vez que se quiere cambiar algo, por lo tanto, se estudió los Frameworks web existentes, eligiendo React.js por su modularidad y renderizado dinámico, y se el comienzo de migrar las funcionalidades de lo ya existente. A su vez de decidido cambiar la arquitectura interna del backend y el despliegue originalmente se estaba usando Azure app Service.

5.4 Prototipo V3 , Sistema sin gestión de penalizaciones.

Lo primero que se planto fue construir la aplicación web siguiendo el patrón modelo vista controlador, utilizando un contenedor Docker para cada elemento es decir base de datos PostgreSQL , aplicación web REACT y api REST. Cada uno independiente de los demás.

Se comenzó por la creación de un nuevo diagrama de casos de uso de la aplicación web

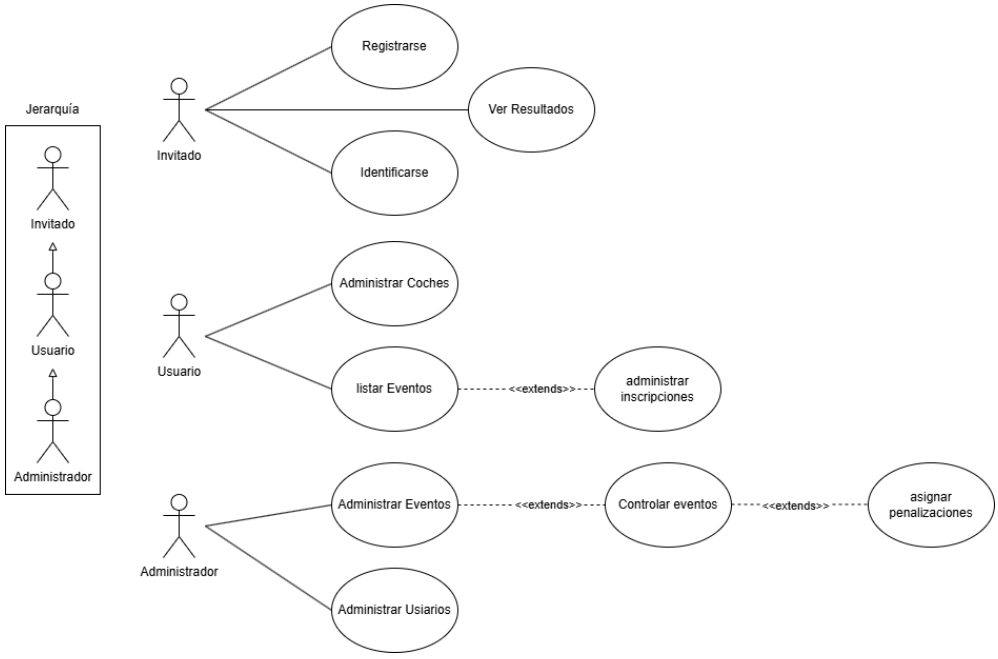


Ilustración 9 : Casos de uso prototipo v3

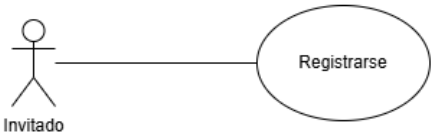
primero se definieron bien los roles que realmente van a interactuar con la aplicación web teniendo entonces 3 actores.

- Invitado, aquellos que interactuaran con el portal web sin haber entrado previamente o sin haberse identificado en la misma.
- Usuario, este será el rol correspondiente a los pilotos de los vehículos.
- Administrador, rol correspondiente a los directores de carrera y administradores del sistema web.

Una de las novedades que presentara este sistema con respecto al anterior es que estos actores presentan una jerarquía de herencia por lo cual se establece que Administrador podrá realizar todas las acciones de Usuario y as u vez Usuario puede realizar todas las funciones de Invitado. Teniendo en cuenta esto el diagrama que se presenta es el correspondiente a (Ilustración 9 : Casos de uso prototipo v3) , también se reduce el número de casos de uso ya que se evitan repeticiones y se generalizan procesos.

5.4.1 Casos de uso Actor Invitado

Como se observa en el diagrama anterior este actor principalmente solo puede ver el inicio de sesión, el registro de la aplicación y acceder a los resultados de las pruebas. Para entender mejor estos casos de uso se muestra sus análisis en detalle.



Nombre :	Registrarse

Descripción: Permite al invitado Darse de alta en el sistema.
Este caso de uso permite a un usario invitado Darse de alta en la applicacion para poder acceder a las funciones del rol Usario y poder recordar sus datos , al registrase se crea una entrada en la Tabla Usuarios de la base de datos.

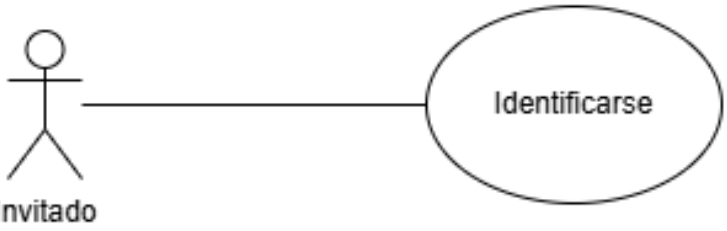
Actores: Invitado

Precondiciones: Ninguna

Flujo de Eventos :

1. el invitado presiona el botón **registro** situado en el centro derecha de la pantalla , alternativamente puede pulsar el hipervínculo **registro** en la barra superior de navegación
2. se le mostrara el formulario de registro
3. el invitado procederá a rellenar sus datos de registro , nombre y email a continuación procederá a crear una nueva contraseña
    1. A medida que el invitado comienza a rellenar el primer campo de contraseña se le mostrara las instrucciones sobre las distintas restricciones que posee la contraseña.
    2. posteriormente completara el segundo campo repitiendo la contraseña , a medida que lo rellena la coincidencia con la contraseña del campo 1 esta siendo analizada y se muestra al invitado.
4. finalmente se valida el formulario , si los datos son correctos se envía una petición POST a la api para que esta cree un nuevo usuario en la base de datos y si todo sucede exitosamente se informa al usuario de su creación.
    1. en caso de que algo no suceda exitosamente se le informara del problema

Postcondiciones: Se crea un usuario en la BBDD



Nombre :	Identificarse

Descripción: Permite al invitado Acceder a la aplicación en función del rol
Este caso de uso permite a un usuario invitado acceder al panel de control asignado ya sea de usuario básico o administrador

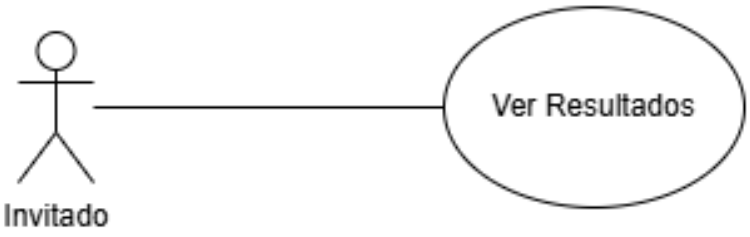
Actores: Invitado

Precondiciones: Estar dado de alta en el sistema

Flujo de Eventos :

1. el invitado presiona el botón **Identificarse** situado en el centro izquierda de la pantalla , alternativamente puede pulsar el hipervínculo **Identificarse** en la barra superior de navegación.
2. se le mostrara el formulario para iniciar sesión.
3. el invitado procederá a rellenar sus datos de inicio : email y contraseña.
4. finalmente se valida el formulario , si los datos son correctos se mostrara el panel de control que corresponda de acuerdo a su rol asignado.

Postcondiciones: Se guarda en la cookie de sesión el estado de identificado



Nombre :	Ver Resultados

Descripción: Permite al invitado Ver los tiempos y posiciones de los eventos disputados o en su defecto ver las Inscripciones registradas.

Actores: Invitado

Precondiciones: Ninguna

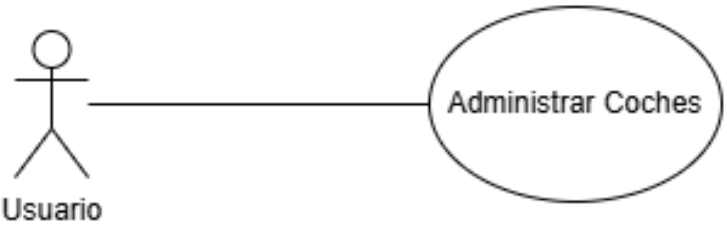
Flujo de Eventos :

1. el invitado pulsa el hipervínculo **Tiempos** en la barra superior de navegación
2. se le mostrara el listado de eventos registrados en la aplicación en forma de botones
3. el invitado proedera a selecionar el que desee ver
    1. cuando el evento es pulsado se muestra la tabla de tiempos corespondiente al evento
        1. alternativmente puede pulsar otro evento y la tabla se actualizara mostrando los datos correspondientes.

Postcondiciones: Ninguna

5.4.2 Casos de uso Actor Usuario

Como acciones específicas del rol de Usuario tendríamos administrar coches y listar eventos



Nombre :	Administrar Coches

Descripción: Permite al Usuario organizar su inventario de coches , ya sea añadiendo o borrando.

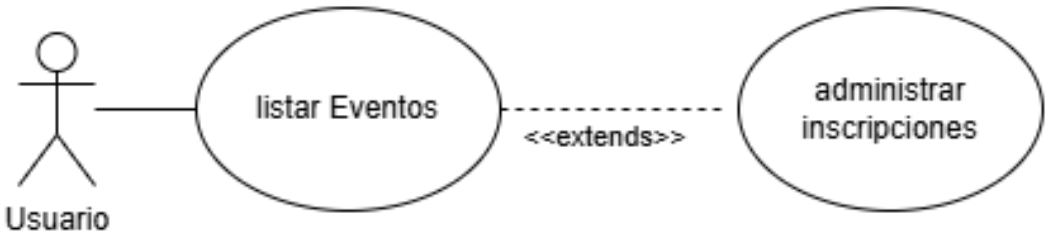
Actores: Usuario

Precondiciones: Dado de alta e identificado en la app

Flujo de Eventos :

1. el Usuario pulsa el hipervínculo **Mis Coches** en la barra superior de navegación
2. se le mostrara la pantalla Mis Coches , en ella aparecerán en horizontal un conjunto de botones que representaran cada uno de sus coches (vacío en caso de no tener ninguno) y bajo ellos el formulario para añadir nuevos coches.
3. A continuación si el usuario desea añadir un coche deberá rellenar el formulario y pulsar el botón añadir ,
    1. tras ello se actualizara automáticamente la lista de coches
4. si el usuario desea eliminar un vehículo , deberá clicar sobre dicho vehículo y pulsar el botón borrar
5. tras ambas operaciones se actualiza la DB con la correspondiente operación en la tabla de coches

Postcondiciones: Se actualiza la tabla coches de la DB



Nombre :	listar eventos

Descripción: El usuario puede ver los eventos disponibles y mediante el caso extendido puede gestionar sus inscripciones

Actores: Usuario

Precondiciones: Dado de alta, estar identificado en la app, poseer al menos un coche y que el administrador haya creado un evento.

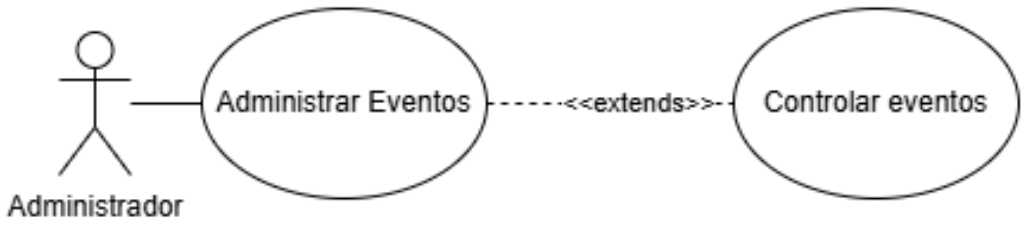
Flujo de Eventos :

1. al identificarse como usuario , se abrira la vista "dashboard" donde se mostraran en el lado izquierdo en forma de botones con una disposicion vertical la liasta de eventos existentes en la app.
    1. si desea inscribirse en una evento debe selecionar uno los eventos , a continuacion elegir un coche en el desplegable y pulsar el boton inscribirse.
    2. si desea borrar una inscripcion de la lista "Mis inscripciones" debe pulsar el icono de papelera que aparece en el lado derecho de la misma.

Postcondiciones: Se actualiza la tabla coches de la DB

5.4.3 Casos de uso Administrador

Este rol es el que más funciones desempeña dentro del portal web y por lo tanto tiene más casos de uso , los cuales le permitirán organizar bien las pruebas disputadas a los directores de carrera.



Nombre :	Administrar Eventos

Descripción: Permite al administrador crear , editar y borar los eventos o pruebas , este caso cuenta con 2 extensiones , controlar eventos y administrar participantes.

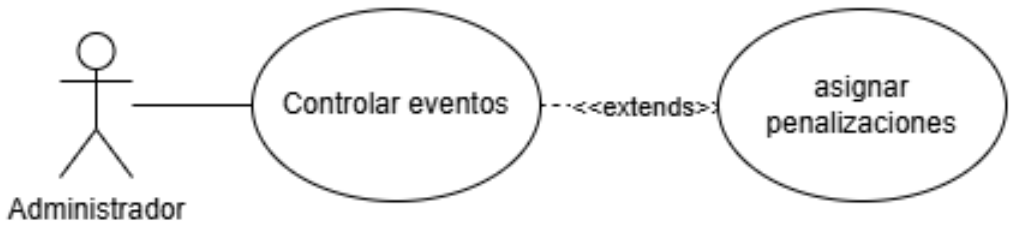
Actores: Administrador

Precondiciones: Administrador identificado en la aplicacion.

Flujo de Eventos :

1. el Usuario pulsa el hipervínculo **Eventos** en la barra superior de navegación
2. se le mostrara la pantalla Eventos , en ella aparecerán en el lado izquierdo en disposición vertical el un conjunto de botones que representaran cada uno de los eventos (vacío en caso de no existir ninguno) y en el lado derecho un formulario para añadir nuevos eventos , bajo estos controles los botones de borrado y control de los mismos.
    1. A continuación si el usuario desea añadir un evento deberá rellenar el formulario y pulsar el botón añadir ,
    2. en el caso de eliminación deberá pulsar sobre un evento y el botón eliminar
    3. si desea pasar a la vista de control del evento deberá pulsar sobre el y el botón controlar caso <<extends>> controlar
3. tras estas operaciones operaciones se actualiza la DB con el correspondiente resultado en la tabla de eventos

Postcondiciones: lista de eventos modificada y en vista de la misma



Nombre :	Controlar eventos

Descripción: Permite al administrador gestionar un evento , marcando el comienzo del mismo y monitorizar cada paso por las puertas, asi como mediante el caso de uso extendio asignar penalizaciones , añadir los tiempos correspondientes a dichas penalizaciones

Actores: Administrador

Precondiciones: existencia de eventos , pantalla de eventos abierta

Flujo de Eventos :

1. el Administrador pulsa sobre el evento que desea controlar
2. pulsa el boton controlar
3. se mostrara la tabla de participantes bajo ella los controles para conectar la estacion principal a la aplicacion , el control para escanear tajetas y el control para iniciar el evento

Postcondiciones: Ninguna

5.4.4 Interfaz de usuario, aplicación React+Vite

Una vez se ha tenido claro el comportamiento definido en los casos de uso se comenzó su desarrollo, está construida siguiendo el patrón SPA ya que es servida íntegramente al navegador en la primera llamada al api, todas las rutas web de React son internas y virtuales.

Como apariencia de la aplicación se eligió fondo oscuro y letras claras, la mayoría de los estilos provienen de la librería Material UI [53] , además de contar con algunos retoques hechos usando CSS y Sass [54] estas son algunas de las ventanas de la interfaz.

Ilustración 10: Landing Portada

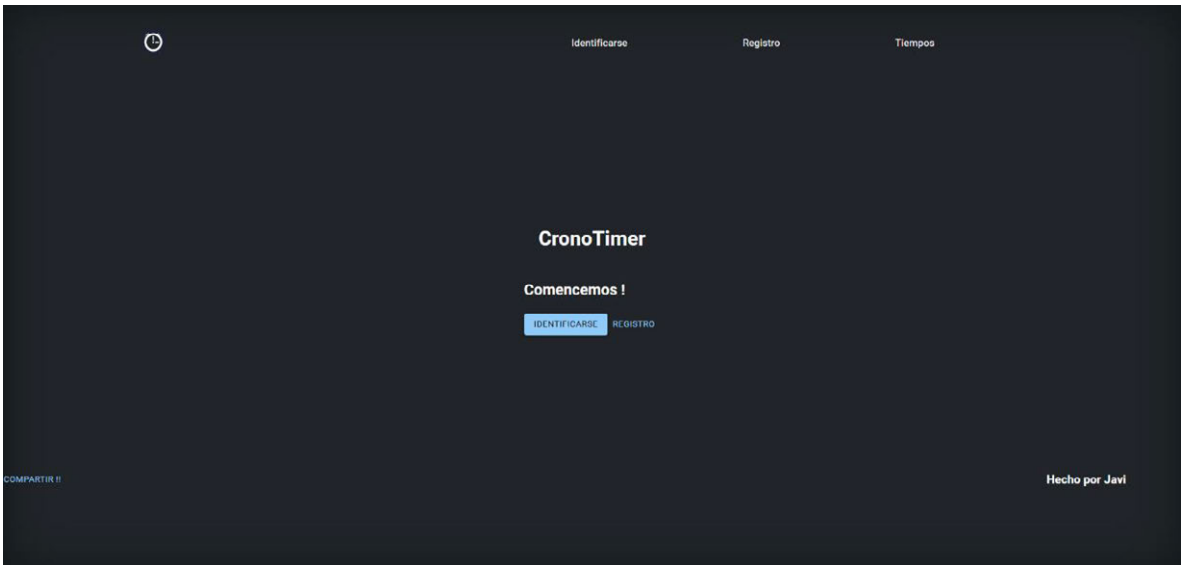


Ilustración 11: Formulario inicio de sesión

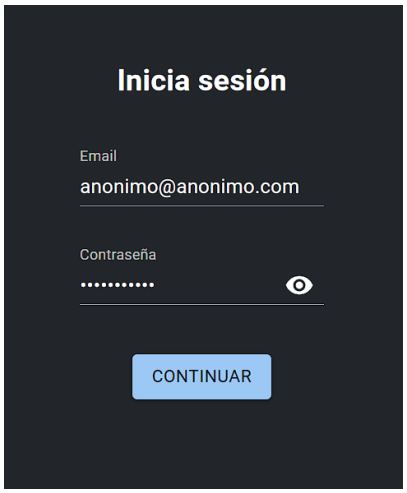


Ilustración 12: Formulario de registro

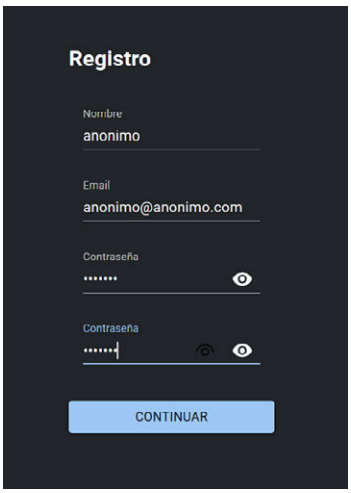


Ilustración 13: Dashboard

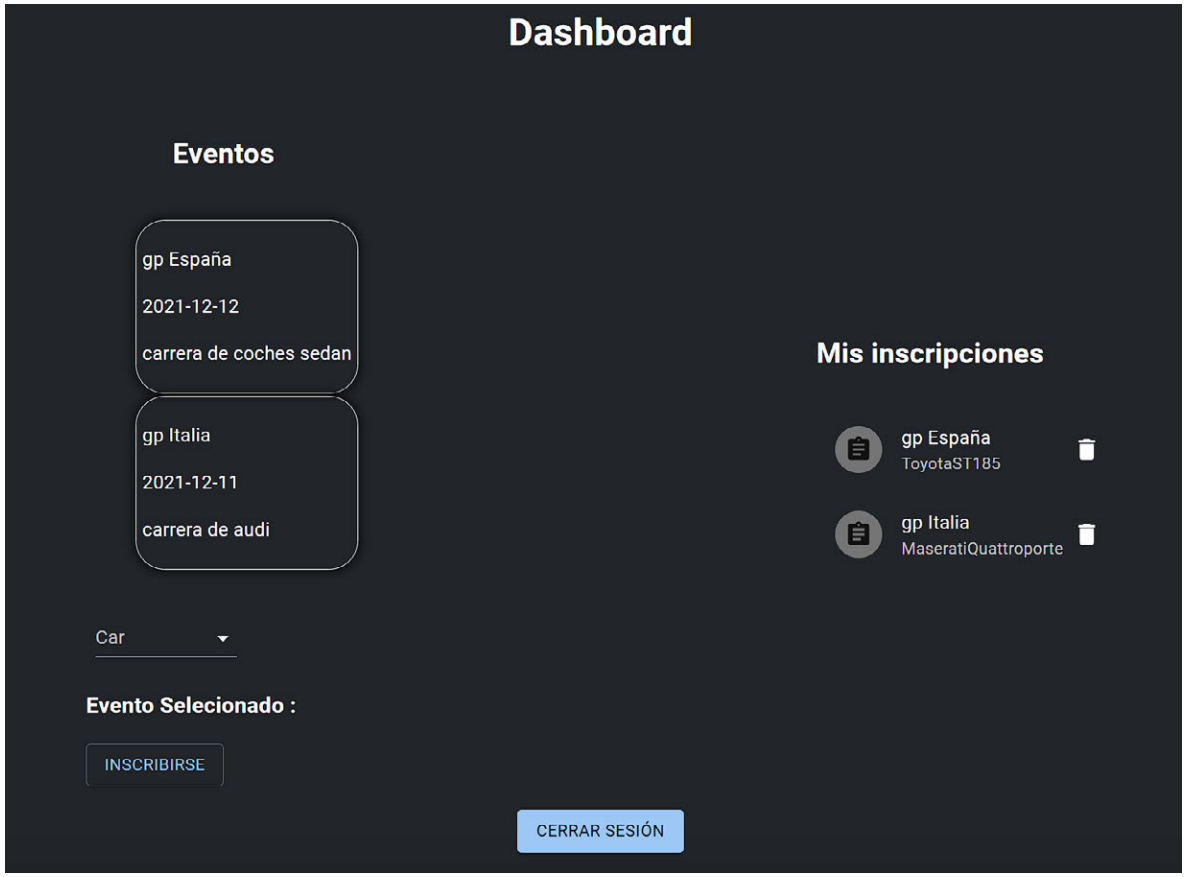


Ilustración 14: Manager de Coches

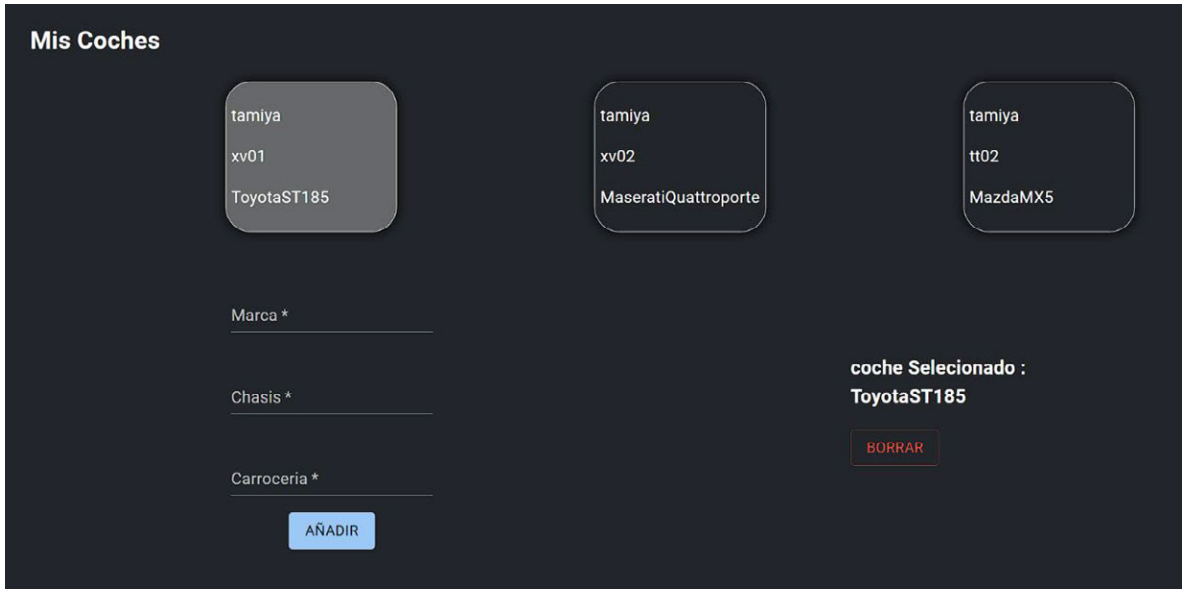


Ilustración 15: Manager de Usuarios

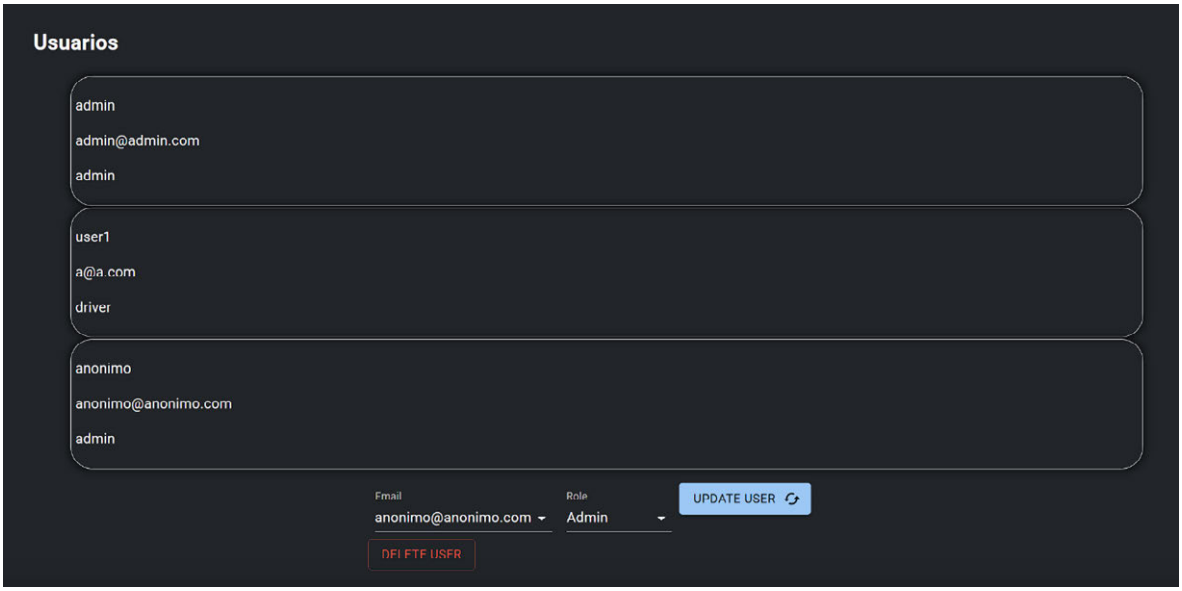


Ilustración 16: Manager de Eventos

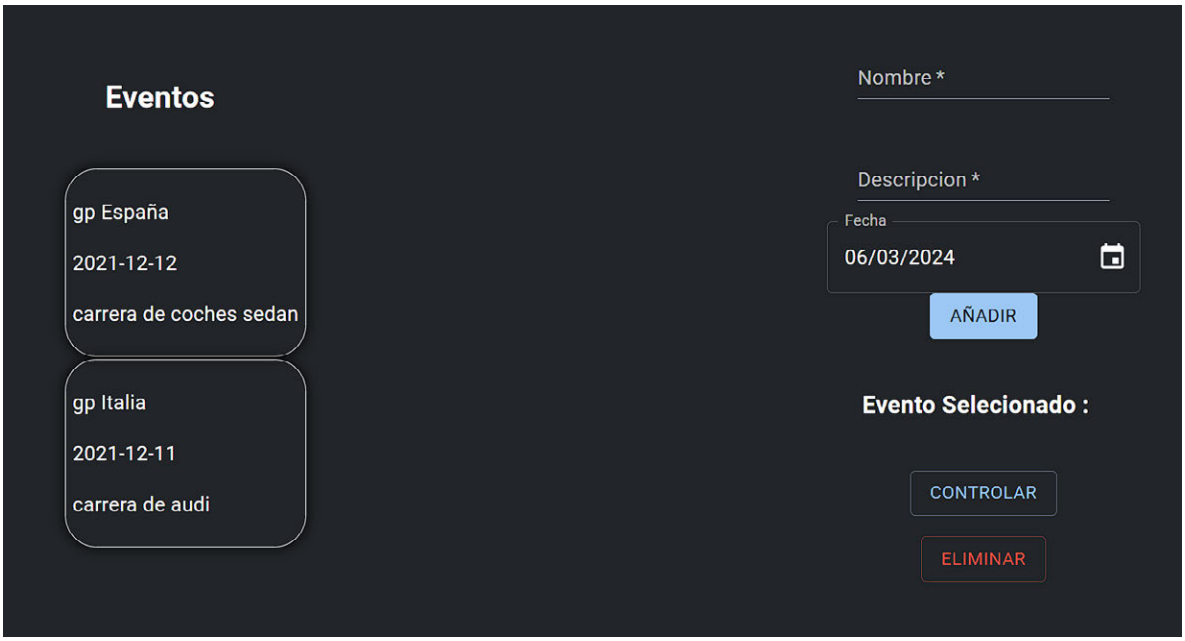


Ilustración 17: Panel de control de Eventos

Ilustración 18: Pantalla de resultados

Estas ilustraciones se harán referencia en el punto 6 de este documento, Flujo de funcionamiento para la celebración de una prueba, donde se explica el curso de eventos que trascurre para la realización completa de una prueba

5.4.5 Base de Datos

Para almacenar los datos de la aplicación en esta tercera iteración del desarrollo vamos a tener 4 tablas, (Ilustración 19 : Base de datos v3)

- Users, esta tabla se usará para almacenar los datos de registro y acceso de la aplicación, almacenara el nombre de usuario , el email , su contraseña y el rol que tiene en la app.
  Como medida de seguridad las contraseñas son almacenadas de manera cifrada como hashes utilizando el algoritmo bcrypt [55]
- Cars, almacena los coches de cada usuario, teniendo los atributos, Marca (brand) , Modelo (model) , carrocería (body) , además del id del usuario al que pertenece
- Events, aquí se almacenará los eventos registrados en la aplicación, de ellos se guarda el nombre, una descripción y la fecha en la que van a suceder.
- Participations, esta tabla será la encargada de guardar los tiempos y las inscripciones en cada evento, se crea una fila de participación por cada coche inscrito y al evento que pertenece, así como los 6 tiempos que puede hacer en cada tramo , así como el acumulado de todos ellos para poder ordenar la clasificación en función de este campo , inicialmente los tiempos están vacíos y se van rellenando según es necesario

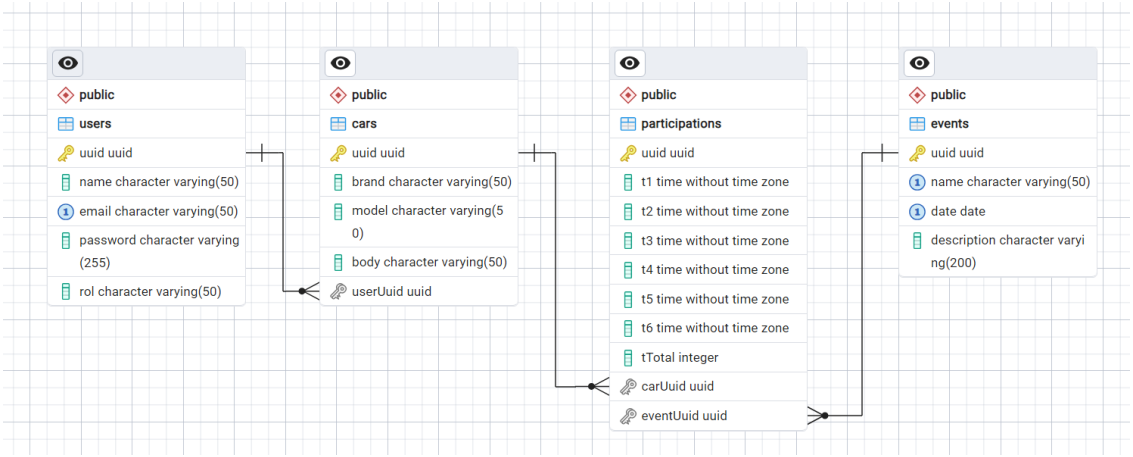


Ilustración 19 : Base de datos v3

5.4.6 Api Node.js

A continuación, se muestran las rutas disponibles para consultar datos ,las cuales se encargan de hacer de mediador entre la vista y la base de datos , se encuentra online accesible desde cualquier navegador, cuenta con medidas de seguridad como middlewares de control de acceso mediante JWT [56], API Key [57] con 10 usos y cifrado de datos sensibles.

Tabla 3 : Rutas Login jwt

Método HTTP	Ruta	Acceso y autenticación	Parámetros de entrada	Acción
GET	/api/login/logout	Abierto permitido a todos	ninguno	Vacia la cookie de JWT lo cual desencadena el cierre de sesión
POST	/api/login	Abierto permitido a todos	Body: email, password	Establece la Autenticación JWT , inicio de sesión

Tabla 4 : Rutas Gestión usuario

Método HTTP	Ruta	Acceso y autenticación	Parámetros de entrada	Acción
GET	/api/users/:email	Autenticación JWT , Rol administrador	Params: email	Retorna Usuario con el email correspondiente, si se deja vacío devuelve todos los usuarios
POST	/api/users/create	Api key	Body: nombre, email, password, role	Crea un nuevo usuario , permitiendo elegir su rol
POST	/api/users/register	Abierto permitido a todos	Body: nombre, email, password	Crea un nuevo usuario con rol “driver” por defecto
PUT	/api/users/privileges	Autenticación JWT , Rol administrador	Body: email, role	Permite cambiar el rol de un usuario con el correspondiente email
DELETE	/api/users/delete	Autenticación JWT , Rol administrador	Body: email	Permite borrar un usuario con el correspondiente email

Tabla 5 : rutas Gestión Coches

Método HTTP	Ruta	Acceso y autenticación	Parámetros de entrada	Acción
GET	/api/getfromUser/:email	Autenticación JWT	Params: email	Retorna los vehículos correspondientes al usuario con el email indicado
POST	/api/cars/addtoUser	Autenticación JWT	Body : email, carBrand, carModel, carBody	Crea un nuevo vehículo con el modelo , marca y carrocería especificados y lo asigna al usuario con el email indicado
PUT	/api/cars/update	Autenticación JWT	Body : email , role	Permite cambiar el rol de un usuario con el correspondiente email
DELETE	/api/cars/remove	Autenticación JWT	Body : email	Permite borrar un usuario con el correspondiente email

Tabla 6 : rutas Gestión Eventos

Método HTTP	Ruta	Acceso y autenticación	Parámetros de entrada	Acción
GET	/api/events/all	Abierto permitido a todos	ninguno	Devuelve todos los eventos
GET	/api/events/byName/:name	Autenticación JWT, Rol driver	Params: name	Devuelve evento con nombre indicado
POST	/api/events/new/	Autenticación JWT , Rol administrador	Body : Name , description, date	Crea un nuevo evento , a partir de su nombre , fecha de inicio y descripción
PUT	/api/events/update/	Autenticación JWT , Rol administrador	Body : uuid, name , description, date	Permite cambiar la información de un evento existente mediante su uuid

DELETE	/api/events/remove/	Autenticación JWT , Rol administrador	Body : name	Permite borrar un evento con el nombre

Tabla 7 : Rutas Gestión Participaciones

Método HTTP	Ruta	Acceso y autenticación	Parámetros de entrada	Acción
GET	/api/participations/ getMyParticipations/:email	Autenticación JWT	Params: email	Retorna todas las particiones pertenecientes al email dado
GET	/api/participations /getParticipations/:eventName	Abierto permitido a todos	Params: eventName	Devuelve todas las participaciones pertenecientes a un evento
POST	/api/ participations /new	Autenticación JWT	Body: carUuid, eventUuid	Crea una nueva participación En el evento correspondiente y con el coche dado
PUT	/api/ participations /addTime	Abierto permitido a todos	Body: carUuid, eventUuid, time, index	Permite añadir tiempos a la participación cuyo coche y evento coincida
DELETE	/api/ participations /delete	Autenticación JWT	Body: uuid	Permite borrar una participación mediante su id

5.5 Diagrama de aplicación

Tras definir la aplicación web se empezó a pensar en cómo se iba a realizar la conexión de cada uno de los componentes del sistema por separado, se estudió su integración como conjunto y se plantea el siguiente diagrama de la arquitectura general.

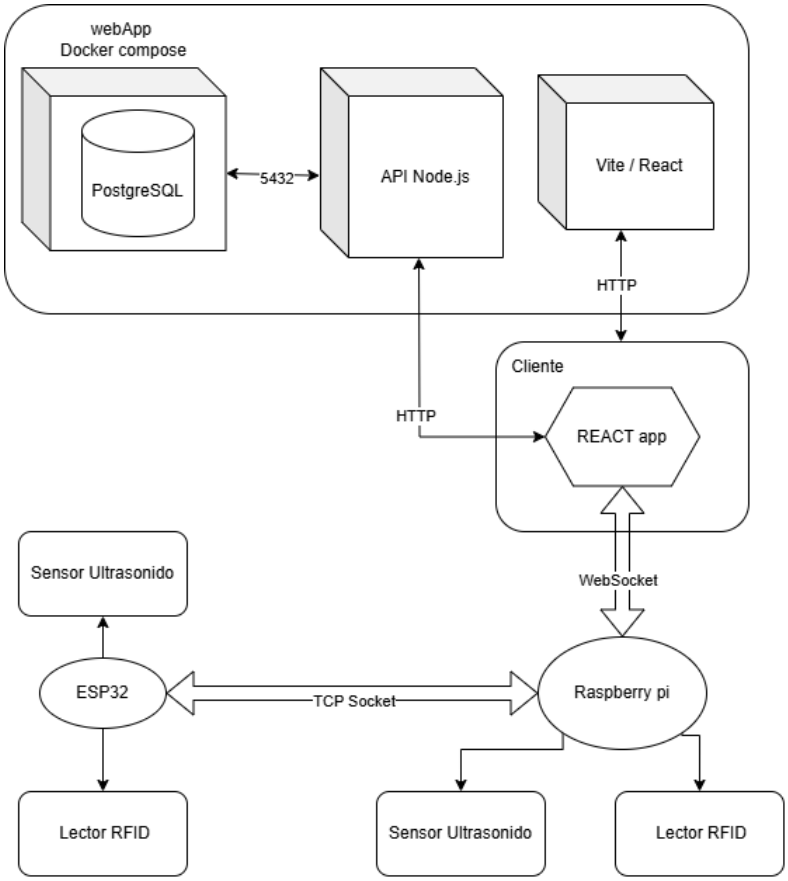


Ilustración 20 : Arquitectura v3

Se observan 3 contenedores Docker [58] para la web app orquestados por Docker Compose [59], un contenedor para la base de datos conectado mediante el puerto 5432 al contenedor de la api , este permanece escuchando peticiones http sobre el puerto 3000, por otro lado tenemos un contenedor que tiene un servidor vite con la aplicación React escuchando el puerto 80 (número de puerto usado normalmente para las peticiones de navegador sin especificar el número de puerto), la aplicación REACT será servida al cliente por lo tanto se encontrara fuera del servidor en el momento de su ejecución, y se conectara a un Raspberry pi ubicado dentro de su misma red local mediante Websocket , a su vez la Raspberry se conectará automáticamente al ESP32 que también deberá estar en la misma red local, usando un Socket TCP tradicional, estos últimos tendrán conectados físicamente los sensores correspondientes

5.5.1 Estación salida (Raspberry Pi)

La estación de salida cuenta con el sistema operativo Raspberry pi os (basado en Debian) sin escritorio, cuenta con la instalación de Python 3.11 como entorno de ejecución, cuenta con el siguiente comportamiento recogido en este diagrama de flujo:

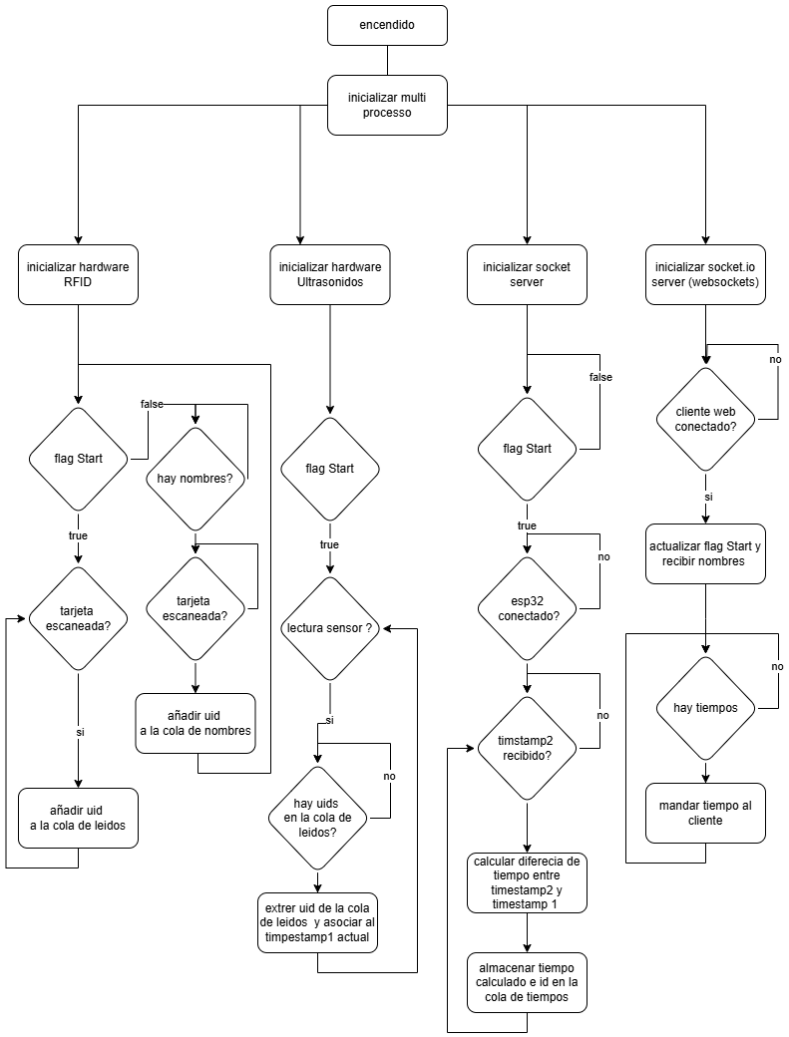


Ilustración 21 : Flujo Estación Salida

Como fuente de datos para nuestra aplicación web y monitoreo real de los vehículos tenemos la Raspberry pi la cual presenta el funcionamiento siguiente, al momento del encendido del dispositivo mediante una tarea programada en el sistema operativo inicia nuestro programa (Ilustración 21 : Flujo Estación Salida) el cual hace lo siguiente, inicia 4 procesos concurrentes los cuales realizan estas funciones simultáneamente

- Proceso RFID, inicializa los sensores y permanece a la espera hasta que un flag interno indica el comienzo de la prueba, mientras no está iniciada la prueba espera la llegada de nombres por parte del Websocket. Si hay nombres en la cola de nombres escucha en busca de una tarjeta y asocia el nombre con la tarjeta. Por otro lado, cuando empieza el evento

es espera por el escaneo de tarjetas y almacena sus id en una cola para que el ultrasonido asocie tiempo Unix.

- Proceso Ultrasonidos, al igual que el proceso anterior espera el flag de inicio, después mediante uso de variables compartidas gestionadas por un manager , analiza la cola de RFID escaneados , cuando hay algún elemento , entonces se inicia el escaneo ultrasónico , y cuando detecta un vehículo , guarda el tiempo Unix actual en una cola asociándole el id de la tarjeta que se escaneo , a su vez elimina este id de la cola de RFID.
- Proceso Socket TCP , como anteriormente , espera a iniciar mediante el flag, una vez marcado el inicio , espera a que el socket reciba mensajes provenientes del Esp32 , este enviara un timestamp de tiempo Unix y un id asociado de tarjeta , en ese momento , la estación busca en su almacén de tiempos Unix el id de la tarjeta recibido , y calcula la diferencia de los dos timestamp , el que tenía guardado y el que acaba de recibir, una vez obtenida esa diferencia se convierte a cadena de texto con formato “minutos:segundos.milisegundos” y lo almacena en una cola de tiempos definitivos con el id de la tarjeta
- Proceso WebSockets, este comienza directamente ya que es el encargado de comunicarse con la aplicación web y el que actualiza el flag de inicio, también es un proceso que funciona escuchando eventos del Websocket , primeramente, espera la conexión de la aplicación , después espera dos eventos
    - Iniciar, este es el evento que cambia el estado del flag de inicio.
    - Escanear, mediante este evento se envía desde la aplicación un nombre de vehículo.

  También analiza la cola de tiempos definitivos, cuando existe una entrada busca en la cola de nombres , el id de la tarjeta asociada, entonces manda el tiempo con el nombre que corresponde a la tarjeta, es decir por ejemplo “01:14.327|Nombre1” para que la aplicación web pueda actualizar la pantalla de control (Ilustración 17: Panel de control de Eventos) y la pantalla de tiempos (Ilustración 18: Pantalla de resultados)

5.5.2 Estación Meta (ESP32)

La estación de meta cuenta entorno de ejecución MicroPython
Cuenta con el siguiente comportamiento recogido en este diagrama de flujo:

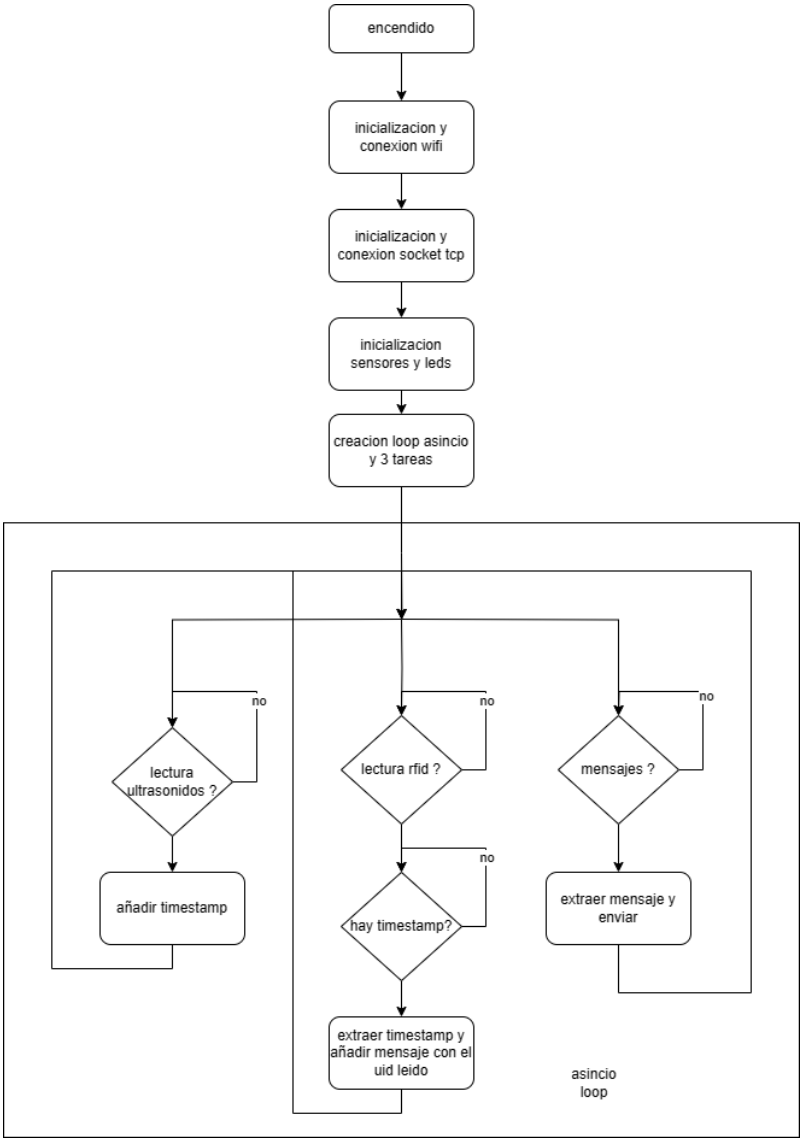


Ilustración 22 : Flujo Estación meta

La estación de llegada tiene un comportamiento que es el siguiente, al encender el esp32 primero se conecta a la red local wifi, después conecta el socket TCP a la estación salida, seguidamente inicializa los sensores y algunos leds , después inicia un bucle con 3 tareas asíncronas.

- Tarea ultrasonidos, escucha permanente mente el sensor de ultrasonidos y si detecta un vehículo, almacena el timestamp de tiempo Unix actual en una cola.
- Tarea RFID, espera a que haya entradas en la cola de timestamp , cuando hay alguno lo extrae de la cola y espera a que escaneen una tarjeta , para asociar el id de esa tarjeta al timestamp , almacena esta asociación en un diccionario de Python.

- Tarea socket, espera a que haya entradas en el diccionario de asociaciones , cuando detecta una la extrae del diccionario y lo envía mediante el socket a la estación principal.

6 Flujo de funcionamiento para la celebración de una prueba

Una vez desarrollado el comportamiento individual de cada componente del sistema ya se puede estudiar el funcionamiento de la integración de todos ellos, por lo tanto voy a explicar cómo transcurriría una prueba para un supuesto campeonato.

Primero un director de carrera ha de estar registrado en el sistema , después inicia sesión en la aplicación web (Ilustración 11: Formulario inicio de sesión), seguidamente registra un nuevo evento en la aplicación y comunica a los pilotos que quieran participar que se pueden ir apuntando , en este momento los pilotos interesados ,deben registrarse (Ilustración 12: Formulario de registro) o iniciar sesión en la app (Ilustración 11: Formulario inicio de sesión), al acceder en su dashboard verán el nuevo evento (Ilustración 13: Dashboard), si ya tenían coches registrados podrán inscribirse, si no ,deberán registrar un coche (Ilustración 14: Manager de Coches), a partir de aquí el director de carrera puede ir viendo quienes se apuntan al evento en la vista de control (Ilustración 17: Panel de control de Eventos) llegado el día de la prueba el director de carrera colocara las estaciones en los puntos correspondientes y los conectara al dispositivo de control (móvil , tableta o pc con la aplicación de web abierta) , a continuación desde el panel de control de eventos y haciendo uso de la estación principal ira escaneando una tarjeta para cada piloto ,la pantalla mostrara el nombre del vehículo ya escaneado en color verde y se la entregara a cada uno , tras ello cuando decida el comienzo de la prueba pulsara el botón iniciar en la aplicación web , a partir aquí podrán ir saliendo los coches de la siguiente forma , se colocara un vehículo en la línea de salida , el piloto correspondiente escaneara su tarjeta en la estación principal , seguidamente podrá salir y recorrer el tramo cuando llegue a la meta deberá escanear otra vez su tarjeta en la estación secundaria , automáticamente el sistema calcular sus tiempo y actualizara la aplicación , tanto el comisario en la vista de control como cualquier persona podrán ver los resultados según vayan pasando los coches , cuando se produzcan adelantamientos el sistema los detecta por el orden de escaneo de las tarjetas y asociara los tiempos correctamente cuando se desee finalizar la prueba el director de carrera solo tiene que desconectar los dispositivos y cerrar sesión en el sistema . Finalmente, los usuarios en cualquier momento de los eventos pueden ver los resultados clicando en el evento correspondiente desde la pantalla tiempos (Ilustración 18: Pantalla de resultados) como ejemplo se puede observar en la Ilustración un evento con 2 participantes.

7 Problemas que han ido apareciendo durante el desarrollo

7.1.1 Etapa temprana Python y Flask

originalmente se había decidido establecer un socket local entre la estación Raspberry pi y el cliente web, para la comunicación de las lecturas de RFID mientras se asignaban a los usuarios , utilizando la librería socketio , pero se encontró que la infraestructura de Azure daba problemas en esta conexión y no se podía realizar , ya que socketio se auto configuraba en modo long polling y se desconectaba con mucha frecuencia , se intentó forzar el modo a WebSockets pero hay alguna incompatibilidad con la plataforma de host y el cliente no establece conexión con la estación . Al migrar de App Service a Virtual machines como el entorno de ejecución no depende de Azure ya no existe ese problema.

7.1.2 Modelado de Datos MongoDB-> PostgreSQL

Modelado de datos en código porque mongo en Python no usaba ODM lo cual generaba un fuerte acoplamiento entre el código y la DB, se pasa a usar PostgreSQL y Sequelize [60] un ORM en JavaScript , de esta manera la DB se elimina ese acoplamiento, ya que conociendo el esquema de datos se puede utilizar esta DB. En caso de que se quiera abordar con otra aplicación o interfaz web.

7.1.3 Concurrencia en Raspberry pi

Para ello se estudió los distintos métodos de concurrencia y asincronía [61], para ello se hizo una prueba de proceso ya que se quería conocer las capacidades disponibles y el tiempo que gasta cada uno de los métodos, por lo tanto, se codifico un ejemplo rápido en el que se pedía incrementar dos contadores mediante un bucle de 100000000 iteraciones y se comparó los tiempos de ejecución de método asíncrono, método multihilo y método multiproceso.

Con el método asíncrono, ambos bucles se reparten tiempo de la CPU para realizar la cuenta, pero la utilizan secuencialmente, es decir primero uno y luego el otro por lo que el tiempo fue mayor.

Con el método multi hilo cada hilo de ejecución depende del proceso padre y comparten recursos, también depende de la implementación de esa gestión en Python, estos hilos también operaban de manera secuencial porque dependían de las ordenes de ejecución de Python, pero tardaban un poquito menos que la asincronía.

Con el método multiproceso cada proceso se separa del proceso padre y utiliza sus propios recursos, son gestionados de manera independiente al programa principal por el sistema operativo, también en caso de contar con un sistema con varios núcleos de CPU se ejecutan cada uno de manera separada y paralela con lo cual tarda en realizarse el conjunto de acciones muchísimo menos, por lo tanto, fue este el método seleccionado.

7.1.4 Concurrencia en ESP32

En un principio se pensó que al tener dos núcleos iba a poder realizar cierta concurrencia, pero al habilitar la comunicación wifi o bluetooth uno de los núcleos se utiliza íntegramente para ello con lo cual solo queda disponible para desarrollo el otro núcleo. Ya que en este caso es un microcontrolador mono núcleo solo se puede utilizar técnicas de asincronía (tiempos de espera y uso de la CPU). [62]

8 Conclusiones

8.1 Metas y objetivos

Según los objetivos definidos al inicio el grado de conformidad con el proyecto es bastante alto ya que se pudieron solventar situaciones bastante relevantes, se ha podido cumplir la mayoría de las metas propuestas, las principales funciones de cronometro con identificación de vehículo y detectar adelantamientos se han implementado con un buen resultado, algunas como el poder asignar penalizaciones que se ha omitido por plazos de tiempo.

8.2 Conocimientos Adquiridos

Durante la realización de este proyecto, se han afianzado y adquirido una gran variedad de conocimientos técnicos y habilidades prácticas que han sido fundamentales para el desarrollo y la implementación de este. A continuación, se detallan los conocimientos más relevantes:

Docker y despliegue: Se ha profundizado en el uso de Docker para la contenedorización y el despliegue de aplicaciones, permitiendo gestionar múltiples canales de comunicación y dispositivos de manera eficiente.

Desarrollo con React: Se ha desarrollado una aplicación personal completa y de gran envergadura utilizando React, integrando tanto bases de datos como WebSockets. Esta experiencia ha sido crucial para entender la creación de aplicaciones web interactivas y dinámicas.

Backend: Se ha adquirido experiencia en el desarrollo del backend utilizando JavaScript, incluyendo la creación de rutas API y la gestión de la lógica del servidor.

Concurrencia y Asincronía en Python y MicroPython: Se ha trabajado con conceptos avanzados de concurrencia y asincronía, lo que ha permitido mejorar la eficiencia y el rendimiento de las aplicaciones desarrolladas.

Coordinación de Subsistemas y Procesos: Se ha aprendido a coordinar diferentes subsistemas y procesos, como la web, API, servidor TCP y servidor WebSockets, gestionando simultáneamente hasta tres dispositivos. Esta capacidad ha sido esencial para asegurar el correcto funcionamiento y la integración de todos los componentes del proyecto.

Estos conocimientos adquiridos no solo han sido aplicados en el contexto del proyecto, sino que también han contribuido significativamente al crecimiento profesional y técnico, proporcionando una base sólida para futuros desafíos en el campo de la ingeniería y el desarrollo de software.

8.3 Planes de futuro

Como principales motivos de seguir con el desarrollo del proyecto uno de los mas claros seria terminar de implementar los objetivos acordados al inicio , con lo cual para el siguiente prototipo se desarrollaría el sistema de penalizaciones del que carece actualmente , a su vez acorde a las pruebas de uso que se realicen del mismo y la conformidad que presenten los usuarios de realizaran las modificaciones necesarias para que sea de su completo agrado.

9 Bibliografía

[1] «Radiocontrol,» 22 febrero 2024. [En línea]. Available: https://es.wikipedia.org/wiki/Radiocontrol.

[2] «Diccionario de la lengua española RAE,» [En línea]. Available: https://dle.rae.es/transpondedor.

[3] «RFID,» 23 enero 2024. [En línea]. Available: https://es.wikipedia.org/wiki/RFID.

[4] «Single-board computer,» 3 mayo 2024. [En línea]. Available: https://en.wikipedia.org/wiki/Single-board_computer.

[5] «Arquitectura ARM,» 10 abirl 2024. [En línea]. Available: https://es.wikipedia.org/wiki/Arquitectura_ARM.

[6] «ARM,» 2024. [En línea]. Available: https://www.arm.com/technologies/big-little.

[7] «¿Qué es API?,» 26 septiembre 2023. [En línea]. Available: https://www.sydle.com/es/blog/api-6214f68876950e47761c40e7.

[8] «enclavedeciencia.rae,» [En línea]. Available: https://enclavedeciencia.rae.es/framework.

[9] «Mapeo Relacional de Objetos,» 17 abril 2024. [En línea]. Available: https://es.wikipedia.org/wiki/Mapeo_relacional_de_objetos.

[10] A. Bevilacqua, «MongoDB ORMs, ODMs, and Libraries,» 31 octubre 2022. [En línea]. Available: https://www.mongodb.com/developer/products/mongodb/mongodb-orms-odms-libraries/.

[11] «Websocket,» 23 marzo 2024. [En línea]. Available: https://www.ibm.com/docs/es/was/9.0.5?topic=applications-websocket.

[12] «DOM,» 13 noviembre 2023. [En línea]. Available: https://developer.mozilla.org/es/docs/Glossary/DOM.

[13] «Single-page application,» [En línea]. Available: https://es.wikipedia.org/wiki/Single-page_application.

[14] «UART rohde-schwarz,» [En línea]. Available: https://www.rohde-schwarz.com/es/productos/test-y-medida/essentials-test-equipment/digital-oscilloscopes/que-es-uart_254524.html#:~:text=UART%20(universal%20asynchronous%20receiver%20%2F%20transmitter,y%20recibir%20en%20ambas%20direcciones..

[15] «Crono 2.0,» [En línea]. Available: http://crono.rallyrcmadrid.com/.

[16] «lapmonitor,» [En línea]. Available: https://lapmonitor.com/store/en/.

[17] «Mylaps rc4,» [En línea]. Available: https://www.mylaps.com/es/systems/sistema-de-cronometraje-rc4/.

[18] «mylaps x2,» [En línea]. Available: https://www.mylaps.com/es/timing-solutions-motorized/x2-system/.

[19] «rabbitrally,» [En línea]. Available: https://www.rabbitrally.com/.

[20] «proyectosconarduino,» [En línea]. Available: https://proyectosconarduino.com/sensores/sensores-de-distancia-infrarojos-para-arduino/#:~:text=%C2%BFC%C3%B3mo%20funciona%20un%20sensor%20de,PSD%20(Position%20Sensing%20Device)..

[21] «solectroshop,» 10 diciembre 2021. [En línea]. Available: https://solectroshop.com/es/blog/como-funciona-el-sensor-de-ultrasonidos-medidor-de-distancia--n99.

[22] «Sensores de medida,» [En línea]. Available: https://sensores-de-medida.es/medicion/sensores-y-transductores/sensores-de-distancia/sensores-de-distancia-por-ultrasonidos/.

[23] «Raspberrypi,» [En línea]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

[24] «Raspberrypi,» [En línea]. Available: https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/.

[25] «orangepi,» [En línea]. Available: http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/orange-pi-4-LTS.html.

[26] «hardkernel,» [En línea]. Available: https://www.hardkernel.com/shop/odroid-n2-with-4gbyte-ram-2/.

[27] «Arduino,» [En línea]. Available: https://store.arduino.cc/products/arduino-uno-rev3.

[28] «raspberrypi,» [En línea]. Available: https://www.raspberrypi.com/products/raspberry-pi-pico/.

[29] «Expressif,» [En línea]. Available: https://www.espressif.com/en/products/socs.

[30] «st,» [En línea]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html.

[31] «fastapi.tiangolo,» [En línea]. Available: https://fastapi.tiangolo.com/.

[32] «Django,» [En línea]. Available: https://www.djangoproject.com/.

[33] «trypyramid,» [En línea]. Available: https://trypyramid.com/.

[34] «Flask,» [En línea]. Available: https://flask.palletsprojects.com/en/3.0.x/.

[35] «expressjs,» [En línea]. Available: https://expressjs.com/.

[36] «hiberus,» [En línea]. Available: https://www.hiberus.com/crecemos-contigo/angular-vs-react-vs-vue/.

[37] «angular,» [En línea]. Available: https://angular.dev/.

[38] «React,» [En línea]. Available: https://es.react.dev/.

[39] «vue,» [En línea]. Available: https://vuejs.org/.

[40] «vitejs,» [En línea]. Available: https://es.vitejs.dev/guide/.

[41] «Google Cloud Platform,» [En línea]. Available: https://cloud.google.com/.

[42] «AWS,» [En línea]. Available: https://aws.amazon.com/es/.

[43] «AWS Educate,» [En línea]. Available: https://www.awseducate.com/.

[44] «Github,» [En línea]. Available: https://github.com/.

[45] «VM Azure,» [En línea]. Available: https://learn.microsoft.com/en-us/azure/virtual-machines/.

[46] «App Service,» [En línea]. Available: https://learn.microsoft.com/es-es/azure/app-service/quickstart-python?tabs=flask%2Cwindows%2Cazure-cli%2Cvscode-deploy%2Cdeploy-instructions-azportal%2Cterminal-bash%2Cdeploy-instructions-zip-azcli.

[47] «Cosmos DB,» [En línea]. Available: https://learn.microsoft.com/es-es/azure/cosmos-db/.

[48] «Mysql vs PostgresSQL,» [En línea]. Available: https://dbconvert.com/blog/mysql-vs-postgresql/.

[49] «mongodb,» [En línea]. Available: https://www.mongodb.com/.

[50] «riobotics-test.weebly,» [En línea]. Available: http://riobotics-test.weebly.com/uploads/9/3/0/9/9309609/medidor_ultrasonico_srf05.pdf.

[51] «hetpro-store,» [En línea]. Available: https://hetpro-store.com/TUTORIALES/hy-srf05-sensor-ultrasonico/.

[52] «tiempo ultrasonido,» [En línea]. Available: https://www.tecnosalva.com/sensor-de-ultrasonidos-en-arduino/.

[53] «Material UI,» [En línea]. Available: https://mui.com/.

[54] «Sass,» [En línea]. Available: https://sass-lang.com/.

[55] «bcrypt wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Bcrypt.

[56] «JWT,» [En línea]. Available: https://datatracker.ietf.org/doc/html/rfc7519.

[57] «API Key – What is an API Key?,» [En línea]. Available: https://rapidapi.com/blog/api-glossary/api-key/.

[58] «docker docs,» [En línea]. Available: https://docs.docker.com/.

[59] «docker compose,» [En línea]. Available: https://docs.docker.com/compose/.

[60] «sequelize,» [En línea]. Available: https://sequelize.org/.

[61] «Entering into the World of Concurrency with Python,» [En línea]. Available: https://www.freecodecamp.org/news/concurrency-in-python/.

[62] «esp32 docs,» [En línea]. Available: https://docs.micropython.org/en/latest/esp32/quickref.html.