EDUINO

SISTEMA DE APOYO, MONITORIZACIÓN Y CONTROL EN EL SEGUIMIENTO ACADÉMICO DEL ALUMNO

Estudiante
Daniel Moreno Arellano
Grado en ingeniería informática
05.663 – TFG Arduino

Consultor
Oriol Jaumandreu Sellarès

Profesores
Pere Tuset Perió
Xavi Vilajosana Guillen

12 de Junio del 2016

Figura 1. Licencia Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	***Eduino**: Sistema de apoyo, monitorización y control en el seguimiento académico del alumno*
Nombre del autor:	*Daniel Moreno Arellano*
Nombre del consultor:	*Oriol Jaumandreu Sellarès*
Nombre del PRA:	*Pere Tuset Perió* *Xavi Vilajosana Guillen*
Fecha de entrega (mm/aaaa):	*25/05/2016*
Titulación	Grado en ingeniería informática
Área del Trabajo Final:	05.663 - Arduino
Idioma del trabajo	Castellano
Palabras clave	Arduino, TIC en el aula , Educación

Resumen del Trabajo (máximo 250 palabras):

El uso de las nuevas tecnologías en el aula se ha visto impulsado debido a la implantación de las tecnologías TIC. De este modo, herramientas de uso diario cómo pizarras o libros, se han visto modernizados a pizarras electrónicas, ordenadores portátiles o tabletas, convirtiéndolos a día de hoy en un complemento funcional más para la educación.

El objetivo que este proyecto persigue se basa en modernizar un elemento presente en el material educativo actual: la agenda escolar, además de integrar todas las funcionalidades extras que permite el uso de las nuevas tecnologías.

Dicho esto, el proyecto trata de acercar al alumno al uso de las TIC en edades tempranas mediante la interactuación con un robot el cual trabajará con una página web en base a los comandos de voz que reciba, ofreciendo al niño un control en tiempo real del sistema. De esta manera, el alumno tendrá acceso a las tareas pendientes asignadas por el tutor de las diferentes materias y a posibles contenidos que puedan reforzar su estudio.

Se desarrolla también un sistema de evaluación sobre el contenido de los temarios que utilizará la metodología de gamificación y permitirá recoger los resultados para la generación de estadísticas del alumno. Estos resultados junto con las tareas asignadas al alumno podrán ser consultados mediante una página web por los padres y el

profesor permitiendo obtener un control actualizado de la información sobre el progreso que experimenta cada estudiante en las diferentes materias y las tareas pendientes.

Abstract (in English, 250 words or less):

The use of new technologies in the classroom has been driven due to the implementation of ICT technologies. Thus, common tools such as blackboards or books, have been modernized on whiteboards, laptops or tablets, making it today in a functional complement for education.

The objective of this project is try to modernize a present element in the current educational material: the school agenda, moreover integrate all the extra features that allows the use of new technologies.

Thus, the project aims to bring students to the use of ICT at an early age by interacting with a robot which will work with a web page based on voice commands received, offering the child a real-time control of system. In this way, students will have access to outstanding tasks assigned by the teacher of the different subjects and possible content that can strengthen their study.

At the same time, also is develops an evaluation system based on the content of the subjects will use the methodology of gamification, and collects the results to allow the generation of student statistics. These results along with the tasks assigned to student may be consulted on a website by parents and the teachers allowing an updated information about the progress for each student experiences in different subjects and tasks remaining.

Notaciones y convenciones

A continuación se muestra un listado de los estilos tipográficos que se detallarán en la memoria para poder distinguir su función dentro del documento.

Título de primer nivel

Título de segundo nivel

Texto normal

Texto resaltado

Figura N. Descripción de la imagen

- **Indicación de una lista**
- Indicación de otra lista

```
Include de librerías en Arduino
Librerías y métodos en Arduino
Comentarios en Arduino
Código de programación en Arduino
Texto y tipos de datos en Arduino
```

```
Código de programación en NodeJS
Texto y variables en NodeJS
Funciones e inicializaciones en NodeJS
Mensajes de consola y definición de Schemas en NodeJS
Métodos de los módulos en NodeJS
Comentarios en NodeJS
```

```
Código en MongoDB
Identificación del ObjectId en MongoDB
```

Cabecera de una tabla
Celda Fila 1
Celda Fila 2

Índice

1. Introducción

1.1 Objetivos del Trabajo

El proyecto que se expone en esta memoria trata sobre el desarrollo de un robot que apoye al alumno durante la realización y preparación de sus tareas. Para llevar a cabo esto, se presentan los objetivos que el robot deberá realizar en esta versión actual del proyecto.

Objetivos

- Implementación y control de un sistema de reconocimiento de voz en Arduino para la interacción de comandos en tiempo real.
- Configuración del sistema de reconocimiento voz especifico al alumno y características del habla.
- Preparación de un sistema de sonidos para avisos o alertas.
- Configuración de un servidor que de servicio al sistema.
- Conectividad en tiempo real entre: Cliente, Arduino y Servidor.
- Implementación de una plataforma web que funcione mediante comandos de voz.
- Implementación de una plataforma web que permita:
    - Consultar la información almacenada en la base de datos.
    - Administración CRUD.
    - Gamificación.
    - Estadísticas gráficas del alumno.
    - Sistema de evaluación.

Objetivos ampliados

- Implementación de una App móvil bajo S.O Android que extienda las funcionalidades de la web para los padres y la gestión del alumno sobre las tareas pendientes.
- Implementación del módulo ESP8266 que aportaría conectividad Wifi al robot.
- Compatibilización con sistemas de accesibilidad para el uso de personas invidentes.
- Ampliación de las funcionalidades web y móvil para establecer un sistema de notificaciones entre el profesorado y los padres.
- Mejoras en la interfaces gráficas y funcionalidades de los diferentes aplicativos.
- Fabricación de un modelo 3D impreso para la carcasa del robot.

1.2 Contexto y justificación del Trabajo

Si se realiza un ejercicio de memoria, a lo largo de estos últimos años en la educación, se puede observar que el planteamiento respecto a la educación digital se ha establecido como una idea necesaria y dispuesta a estar muy presente en las aulas. De este modo, las competencias digitales han hecho replantearse la utilización de una nueva estrategia para el tratamiento de la información y cómo esta se trata. La estrategia se define como un conjunto de competencias digitales que involucren las tecnologías de la información y de la comunicación en la educación, es decir las TIC.

Con esta premisa, este proyecto pretende abordar uno de los aspectos fundamentales en la educación: el interés por el cómo se aprende. Debido a que la forma en que se presenta una asignatura en la escuela y su posterior estudio por parte del alumno, puede resultar crucial para el interés que este desarrolle en ella.

Por lo tanto, el objetivo de este proyecto es abordar la preparación de las evaluaciones y facilitar recursos útiles al alumno para que este pueda identificar el uso del prototipo Eduino como una herramienta tecnológica de ayuda en el aprendizaje. De este modo, al mismo tiempo que el alumno desarrolla un interés por realizar sus tareas, el sistema puede recopilar una serie de valores y resultados trabajados, de tal forma que permitan poder tener un control actualizado de la información sobre el progreso que experimenta en las diferentes tareas de cada materia. Con el objetivo de permitir tanto a padre cómo tutor detectar a tiempo un posible problema educativo, pudiendo contemplar la dedicación del alumno en la materia u observar aspectos que requieran atención durante su aprendizaje, entre otros casos.

Por lo tanto, si se procede a analizar el sistema de aprendizaje actual. En este, el alumno realiza el aprendizaje escuchando en clase el temario que el profesor explica; pero una vez llega a su hogar y debe estudiar este temario, únicamente tiene a su disposición los materiales educativos en papel que utiliza en el curso y las anotaciones que pueda haber obtenido durante las explicaciones. A su vez, la evaluación que puede realizar el alumno se basa en posibles preguntas planteadas en el libro o en documentos que le facilite el profesor. Estas evaluaciones se realizan mediante exámenes finales de la materia o de evaluaciones parciales que darán resultado a una nota final de la materia.

Con este planteamiento surgen algunas preguntas: ¿Qué motiva al alumno a aprender la lección por el mismo? ¿Qué herramientas tiene el alumno para su evaluación personal? ¿Cómo se controla la evolución del alumno? Estas preguntas producen una incertidumbre: ¿El alumno realmente tiene motivación por aprender y mejorar sus objetivos, o estudia por obtener calificaciones positivas que eviten represalias?.

Con todas estas observaciones, este proyecto se define como un trabajo que persigue el desarrollo de un sistema que permita ofrecer más recursos al niño para la preparación de la materia y le permita sentirse reforzado con su propia superación. De este modo, transformando el estudio o la realización de una tarea en algo ameno, al mismo tiempo que convierte el obtener resultados positivos en algo totalmente voluntario. Así, conseguir acercarle a la formación e-learning y a su auto-desarrollo

individual mientras toma contacto con las nuevas tecnologías y convierte su uso en algo cotidiano.

En síntesis, el proyecto supone un reto en ambos ámbitos: personal y educativo. Educativo debido a que la envergadura del proyecto y la magnitud de trabajo son superiores a trabajos anteriores realizados en el grado. Y a su vez a nivel personal debido a que se trabaja con tecnologías que resultan desconocidas, añadiendo todos los posibles problemas que puedan ir apareciendo durante el desarrollo del mismo.

1.3 Enfoque y método seguido

En la primera fase, se realizará un estudio de las diferentes soluciones existentes en la educación con robots, debido a que el objetivo es desarrollar un producto nuevo que siga la línea de otros casos de éxito en el mercado para asegurar su correcta integración. Estos ejemplos, servirán para analizar las necesidades que se plantean en la educación infantil, y así poder adaptar mejor la implementación a los requerimientos del pensamiento computacional en la enseñanza.

Teniendo en cuenta que el proyecto combina tareas que engloban funcionalidades que dependen entre ellas, el cumplimiento y especificación de un calendario de trabajo se convierte en una pieza fundamental. De la misma manera, se destacan una serie de puntos de control (hitos) que analicen si los objetivos propuestos se están llevando a cabo de la forma en que se planificaron o de lo contrario se han de aplicar medidas correctivas en dicho caso. Estos puntos de control se centran en las fechas de entregas propuestas por la asignatura.

Asimismo, se realizarán revisiones periódicas para comprobar si se está llevando a cabo la planificación inicial, quedando en estudio posibles cambios en el diagrama de Gantt.

También se ejecutará un juego de pruebas el cual tiene como objetivo detectar y corregir posibles incidencias. Por lo tanto el periodo de testeo será clave debido a que se interactúa con el sistema en tiempo real. Respecto a posibles evaluaciones con el consultor, ya qué no se realizan en un lugar físico se propone, si fuese necesario, una conexión con alguna herramienta de video llamada cómo Skype para comprobar la correcta ejecución del sistema.

Una vez completada y alcanzada la parte de conectividad, interactividad y funcionalidad del sistema, es decir, se obtiene un sistema completo, la atención se centra en el desarrollo del apartado visual y gráfico del sistema. La estrategia que se plantea es priorizar la usabilidad, ofreciendo una interfaz clara e intuitiva, hecho que va impuesto debido a que el usuario final será un niño que se inicia en las nuevas tecnologías. De este modo, se trabajará con wireframes de bajo nivel en el proceso de diseño inicial, para más tarde conseguir wireframes de alto nivel que cumplan las expectativas deseadas para el proyecto.

En definitiva, la estrategia empleada se rige por el conjunto de fases siguientes: estudio del mercado y situación, planificación, desarrollo, juego de pruebas, solución de incidencias y apartado gráfico. Se ha escogido esta metodología debido a que la parte funcional tiene prioridad ante el resto de partes; con este motivo se realiza un estudio previo de que se quiere hacer, se planifica la metodología de trabajo, revisiones, juegos de pruebas parciales y correcciones de las incidencias detectadas. De esta manera, el proyecto avanza mientras se van haciendo los controles oportunos, evitando únicamente una evaluación final del proyecto. Además, en el tramo final se realiza un juego de pruebas sobre el resultado obtenido, dónde el funcionamiento correcto determinará el comienzo del apartado visual y gráfico.

1.4 Planificación del Trabajo

1.4.1. Recursos de Hardware

Los elementos utilizados en el desarrollo del proyecto han sido los siguientes:

- **Microcontrolador Arduino UNO**: Es una placa electrónica basada en el micro controlador ATmega328. Cuenta con 14 entradas/salidas digitales y otras 6 son entradas analógicas. Además permite la ampliación mediante módulos. Es utilizado para dar funcionamiento al sistema y para la interconectividad entre los diferentes módulos.

- **Microcontrolador Arduino MEGA**: Es una placa electrónica basada en el micro controlador ATMega2560. Esta placa integra el mayor número de señales. Tiene 54 entradas/salidas digitales y 16 entradas/salidas analógicas. Amplia las características del microcontrolador Arduino UNO, su utilización sigue manteniendo las necesidades de la placa anterior.

- **Basic Starter Learning Kit**: Kit básico de iniciación en el aprendizaje para Arduino el cual contiene un amplio catálogo de componentes para la creación de proyectos. Se ha utilizado cómo punto de partida para la primera toma de contacto con Arduino y uso de sus componentes en las fases iniciales.

- **Protoboard (Incluida en el Basic Starter Kit)**: Componente con forma de tablero con orificios, en la cual se pueden insertar componentes electrónicos y cables para componer circuitos. Utilizado durante las primeras fases en el test del dispositivo Arduino con el Servidor.

- **Botones (Incluidos en el Basic Starter Kit)**: Componente que permite el uso de botones y el uso del control de pulsaciones de estos. Utilizados para implementar comandos estáticos simulando el reconocimiento de voz.

- **Display Module 12864 0.96 Inch White SPI OLED**: Pantalla que permite la visualización de contenido bitmap enviado desde Arduino. Su uso se ha centrado en la representación de los iconos de las materias y apartados del menú de la plataforma.

- **Ethernet Shield W5100**: Este módulo permite dotar a placas Arduino que no disponen de conexión a Internet de esta funcionalidad. Ha facilitado al sistema la conectividad para poder enviar y recibir mensajes utilizando sockets.

- **EasyVR Arduino Shield v3**: Módulo que permite implementar un reconocimiento de voz. Se le añade un altavoz de 8 ohm para la reproducción de avisos y sonidos. Ha permitido el reconocimiento de instrucciones de voz por parte del alumno y la emisión de notificaciones de audio.

- **HP Pavilion dv6-3040s**: 1,60 GHz, 4GB RAM DDR3, Intel Core i7, Windows 7 Professional 64bits. Se ha utilizado para desarrollar todo el proyecto.

- **Alimentador:** Alimentador de corriente de 12V DC - 1A para Arduino. Ha permitido alimentar el sistema Eduino durante las fases de evaluación y juego de pruebas.

- **Servidor web:** Procesador Xeon, 8GB RAM, Windows 10. Dara servicio a la plataforma.

1.4.2. Recursos de software

- **Arduino IDE 1.6.7:** Entorno de desarrollo para los scripts de Arduino

- **Dia:** Programa para el diseño de diagramas UML y modelo de Base de datos.

- **Paquete Office 2010:** Programa para la elaboración de la documentación y presentación.

- **GanttProject:** Utilizado para la realización del Diagrama de Gantt.

- **EasyVR Commander:** Utilizado para la configuración de comandos de voz.

- **Sublime Text 2:** Entorno de desarrollo para la programación del entorno web y la comunicación con Arduino.

- **Moqups:** herramienta gratuita utilizada para el diseño de los wireframes de baja fidelidad.

- **Robomongo 0.9.0-RC4**: Herramienta que permite administrar gráficamente bases de datos cómo MongoDB.

- **EasyVR Commander 3.10.0:** Herramienta que nos permite cargar archivos de audio y estructurar los comandos de voz en el módulo EasyVR 3 Shield.

- **QuickSynthesis 5.2.6:** Programa que permite la creación de la tabla de sonidos que se importa al módulo EasyVR 3 Shield.

- **Audacity:** Herramienta que permite convertir la grabación y conversión de pistes de audio al formato correspondiente para la tabla de sonidos de EasyVR 3.

- **LCD Assistant:** Programa utilizado para la conversión de imágenes al formato bitmap requerido por el display.

- **Camtasia Studio:** Programa utilizado para grabar el vídeo de la presentación.

1.4.3. Entregas establecidas (Puntos de control)

El proyecto está formado por 3 PACS evaluables que se establecen para el control y desarrollo del proyecto en base a un ciclo de vida real. Además de estas entregas se presentan un conjunto de evaluaciones posteriores en base al proyecto. Estas fechas han de tenerse en cuenta en la planificación temporal del proyecto.

En la siguiente tabla se reflejan estas entregas:

Fecha de entrega	Evaluación	Descripción
09/03/16	PAC1	Definición de proyecto Primera versión de la memoria
27/04/16	PAC2	Análisis funcional Diseño del entorno Diseño Arduino Diseño Servidor Prototipo Segunda versión de la memoria
25/05/16	PAC3	Implementación Tercera versión de la memoria
19/06/16	Entrega memoria	
23/06/16	Evaluación	
26/06/16	Presentación	

1.4.4 – Diagrama de Gantt

A continuación se muestra el diagrama de Gantt utilizado para planificar el proyecto:

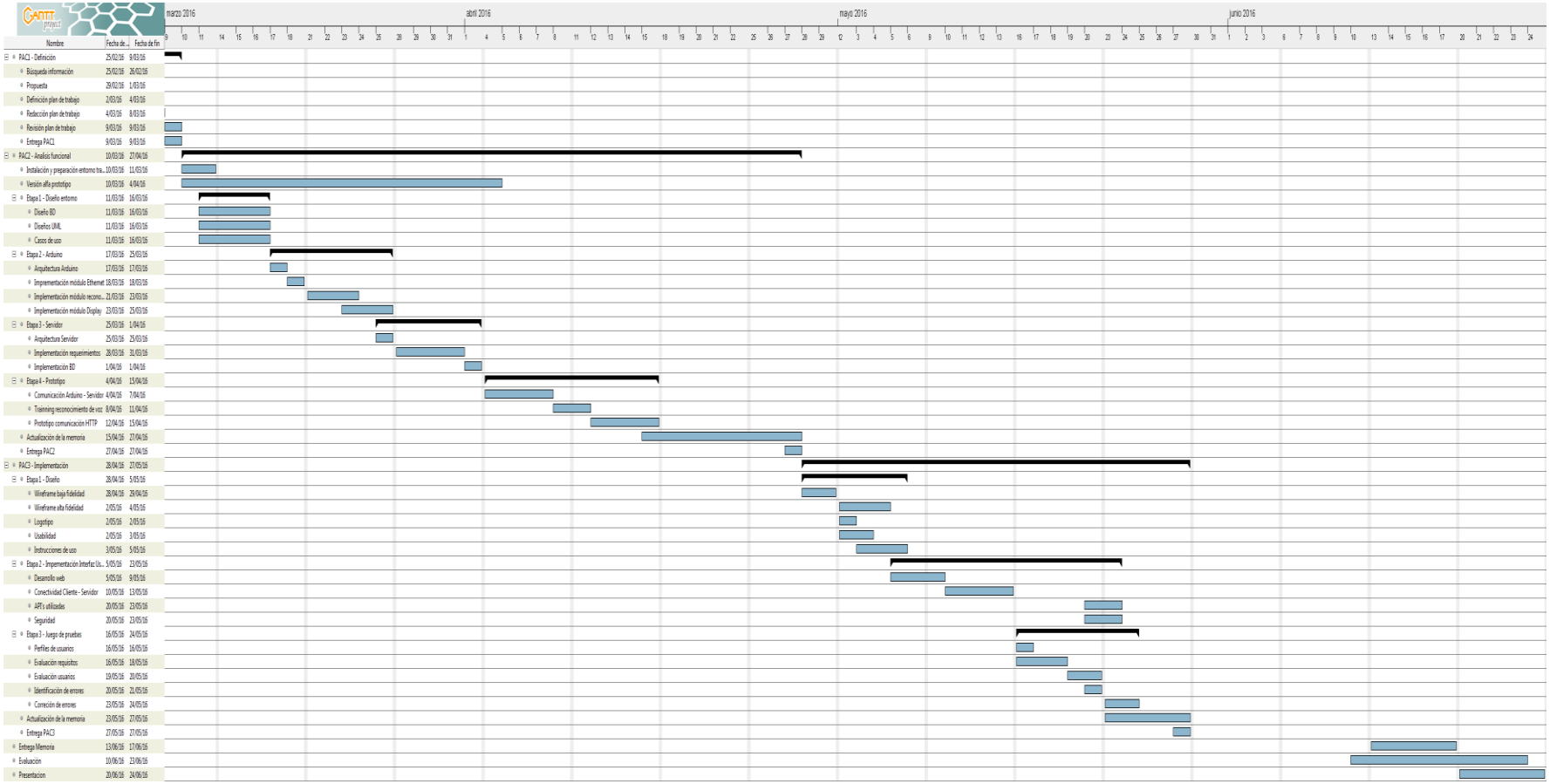


Figura 2 - Diagrama de Gantt

1.5 Breve sumario de productos obtenidos

El proyecto se centra en el desarrollo de un robot que permita la recepción de comandos de voz y envíe dicha información mediante el módulo Ethernet y el uso de sockets a una página web que utiliza la solución MEAN. La combinación de estas tecnologías consigue una respuesta en tiempo real que reacciona a las instrucciones de voz. A su vez, la comunicación es reciproca debido a que el robot interactúa con el usuario mediante sonidos y avisos ya programados en respuesta a los comandos recibidos.

En el apartado web el usuario podrá navegar entre los distintos apartados disponibles para consultar la información disponible o realizar un test de estudio. Además, esta plataforma reaccionará a los comandos del usuario, implementará las funcionalidades CRUD para su administración y la generación de estadísticas en base a los resultados de los test de estudio.

El servidor back-end, está basado en nodejs, express y sockets, utilizando la solución MongoDB cómo base de datos. A su vez, el servidor utilizará el protocolo TCP para recibir y enviar eventos de Arduino, mientras que un servidor HTTP dará servicio a peticiones web implementando sockets.

La parte de front-end estará basada en AngularJS y hojas de estilo CSS3, también se utilizará la librería Bootstrap para conseguir una página responsiva y FontAwesome para los iconos vectoriales. Esta parte utilizará conexiones a la base de datos mongoDB para el procesamiento de datos, junto con nodejs y express, formando el paquete MEAN.

Para asegurar el buen funcionamiento en la comunicación, se añade la complejidad de utilizar un sistema de login de usuarios que evite conexiones de usuarios externos con objeto de la manipulación malintencionada del sistema.

Con todo lo expuesto, la plataforma deja un amplio camino de expansión, ya que existen diferentes ampliaciones que el proyecto podría presentar en una idea futura, cómo el desarrollo de una aplicación móvil para su administración; disponibles para sistemas operativos como Android o IOS o la incorporación de conectividad Wifi para la conectividad, entre otras opciones.

1.6 Breve descripción de otros capítulos de la memoria

Una posible descripción de los capítulos en los que se basará la memoria del proyecto son los siguientes:

- Arquitecturas y tecnología
    - Cliente/servidor
    - Arduino
- Explicación Usos
    - Software/Hardware
    - Otros servicios utilizados.
- Explicación de desarrollo basándose en el diagrama de Gantt
- BBDD
- Diagramas UML
- Perfiles de usuario
- Casos de Uso
- Juego de pruebas
    - Evaluación de requisitos
    - Evaluación con usuarios
- Seguridad
- Prototipos de pantalla
    - Wireframe de baja fidelidad
    - Wireframe de alta fidelidad
- Usabilidad
- Requisitos de instalación
    - Cliente
    - Servidor
    - Arduino
- Instrucciones de uso y guía
- Corrección de errores
- Conclusiones
    - Valoración Personal
    - Proyección a futuro

2. Resto de capítulos

2.1 – Arquitectura

2.1.1 – Arquitectura Arduino

La arquitectura en la que se basa el proyecto para el sistema Arduino es la siguiente:

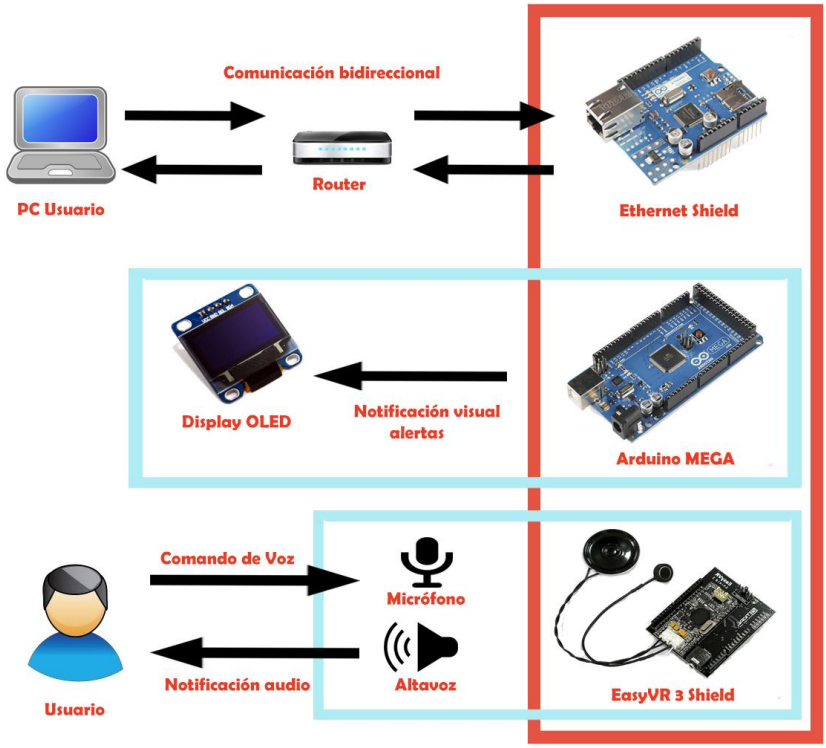


Figura 3 - Arquitectura Arduino

La justificación y uso de los módulos del proyecto se puede encontrar en el apartado **1.4.1. Recursos de Hardware.**

2.1.2 – Arquitectura Servidor

La combinación de la arquitectura anterior junto con la arquitectura del servidor ha seguido el esquema que se representa en la **figura 4**. Cabe decir, que esta arquitectura representa la configuración final del proyecto.

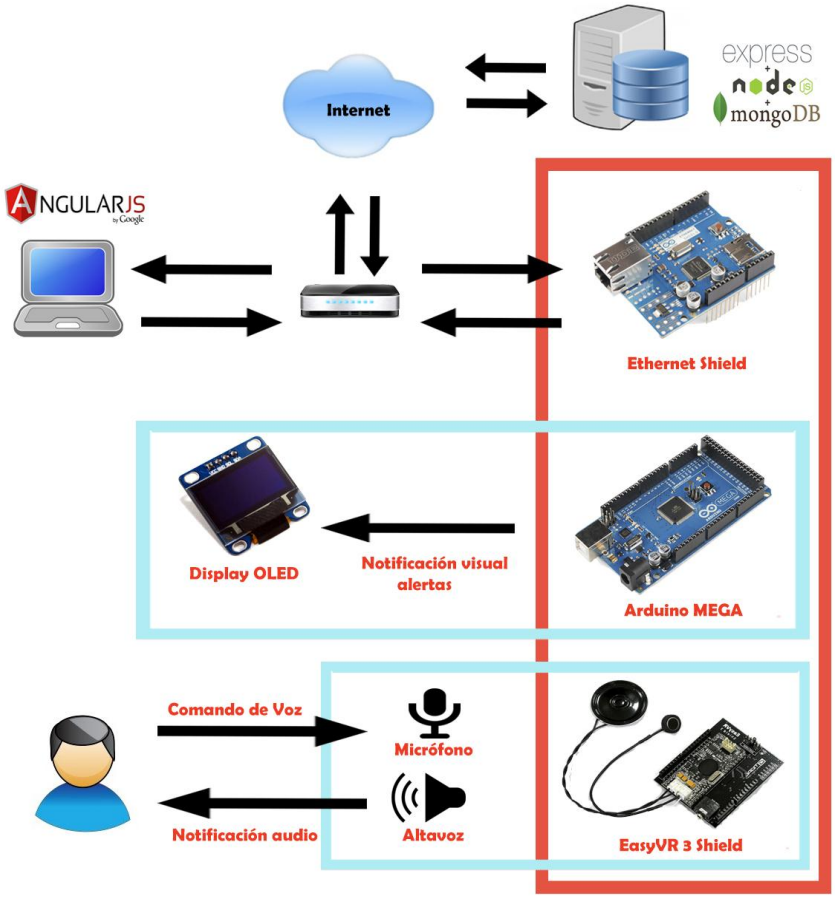


Figura 4 - Arquitectura Servidor

2.2 – Seguridad

Para abordar este apartado se ha implementado un sistema de login que permita la autenticación de usuarios prohibiendo así el acceso anónimo a la plataforma. También se ha creado un esquema de roles de usuario que permita otorgar determinados privilegios. Así, se pueden distinguir tres tipos de usuarios: administrador, alumno o progenitor.

Al mismo tiempo, para verificar la integridad de la plataforma se ha utilizado un middleware; en este caso **passport** para NodeJS. Su uso ha permitido detectar accesos erróneos en el login, proteger el acceso a rutas de la plataforma, redirigir a determinadas rutas o diferenciar que acciones o pantallas mostrar según el rol del usuario. A su vez, se ha hecho uso del módulo **ngCookies**. Dicho módulo permite la creación de una cookie que mantiene la sesión del usuario durante su login y permite que durante la navegación de las diferentes páginas no se pierda su sesión.

En este punto, se procede a definir los perfiles al que va destinado el proyecto, y para ello se definen los grupos de usuarios que harán uso de la plataforma.

Por una parte, se establece un grupo de usuarios con un rango de edades comprendidas de entre los 7 a 11 años. Esto es así, debido a que establecer un target más amplio con edades superiores pudiera no generar interés para los alumnos, del mismo modo que un target inferior pudiera generar conflictos en la adaptación de la tecnología.

Por otra parte, el ámbito educativo engloba otros perfiles cómo profesores o progenitores, dónde no se define un target específico pero si la necesidad de tener unos conocimientos básicos de informática.

A continuación, una vez planteados los perfiles, se procede a explicar los tres roles que la aplicación puede tratar:

Usuario alumno: Este tipo de usuario tiene la capacidad de consultar datos y realizar la inserción en un caso muy concreto: guardar la nota de un test. No posee privilegios para realizar modificaciones ni borrar información en el sistema.

Usuario administrador/profesor: Este tipo de usuario posee un control total sobre los objetos de la plataforma, a excepción de la creación de nuevas materias. Del mismo modo, puede recuperar datos y ejecutar acciones cómo: borrar, modificar o insertar.

Usuario consultor/progenitor: Este tipo de usuario solamente puede consultar datos referentes a un alumno determinado, no puede realizar ninguna modificación ni acción sobre el sistema que ponga en peligro la integridad de este.

Estos perfiles coinciden con los actores y acciones definidas en el apartado **2.5 – Casos de uso.**

2.3 – Diseño BD

La información de diseño de la aplicación esta almacenada en una base de datos noSQL, por lo tanto se basa en un sistema de datos no relacionales y que no necesita seguir un esquema. A su vez, la metodología está orientada a documentos y para facilitar su administración se ha recurrido a la utilización de una interfaz gráfica. Las colecciones que componen la base de datos son las siguientes:

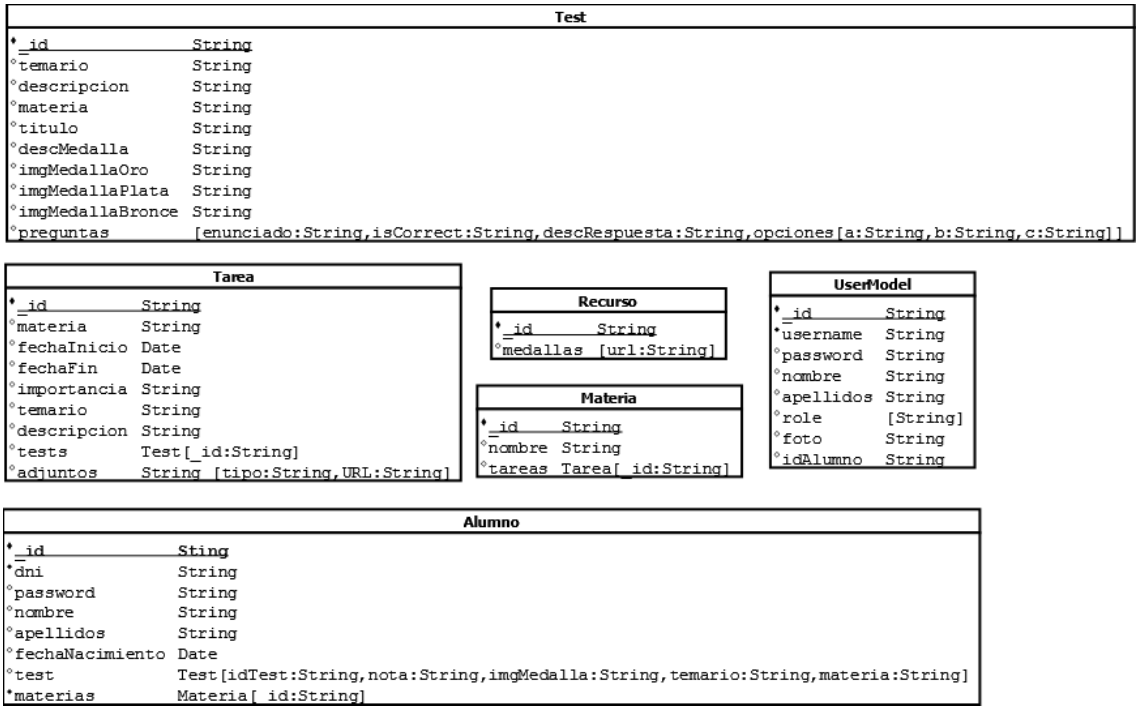


Figura 5 - Esquema BD

A continuación, se detallan las características de las colecciones:

Test: Se encarga de almacenar la información de los test asignados a las tareas. En esta colección se detallan las medallas que el alumno puede conseguir y la colección de preguntas que se le presentarán para cada test.

Tarea: Mantiene las tareas que se recopilan en el sistema y los contenidos complementarios al estudio que pueda tener asignados. Al mismo tiempo, se especifica (si dispone) del test pertinente a la tarea.

Alumno: Almacena los alumnos disponibles en el sistema con su información personal y la lista de materias que está cursando. También almacena el listado de los diferentes test que el alumno ha realizado de las diferentes tareas.

Recurso: Incluye la colección de medallas disponibles que se pueden asignar al test durante su creación.

Materia: Guarda las diferentes materias disponibles en el curso.

UserModel: En esta colección se definen los usuarios que pueden tener acceso al sistema y el rol asignado a este usuario en base a sus privilegios.

2.4 – Diseño UML

Para la preparación del diseño UML de **Arduino** se tienen en cuenta los diferentes archivos creados respecto a los módulos implementados y la referencia de ellos en el código principal. Se presentan archivos diferenciados en función de los módulos que

utiliza el proyecto. La representación de una composición es debido a que se trata de un tipo de relación más fuerte, es decir, los componentes del proyecto constituyen una parte del objeto compuesto y dependen de este para su funcionamiento.

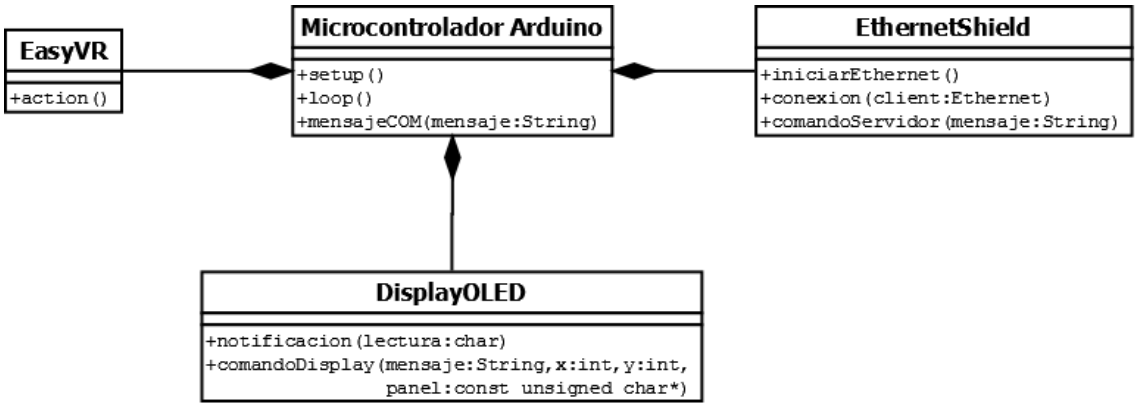


Figura 6 - Diagrama UML Arduino

La preparación del diagrama UML para el **servidor** clasifica las relaciones que se establecen entre los ficheros y métodos que en ellos se definen.

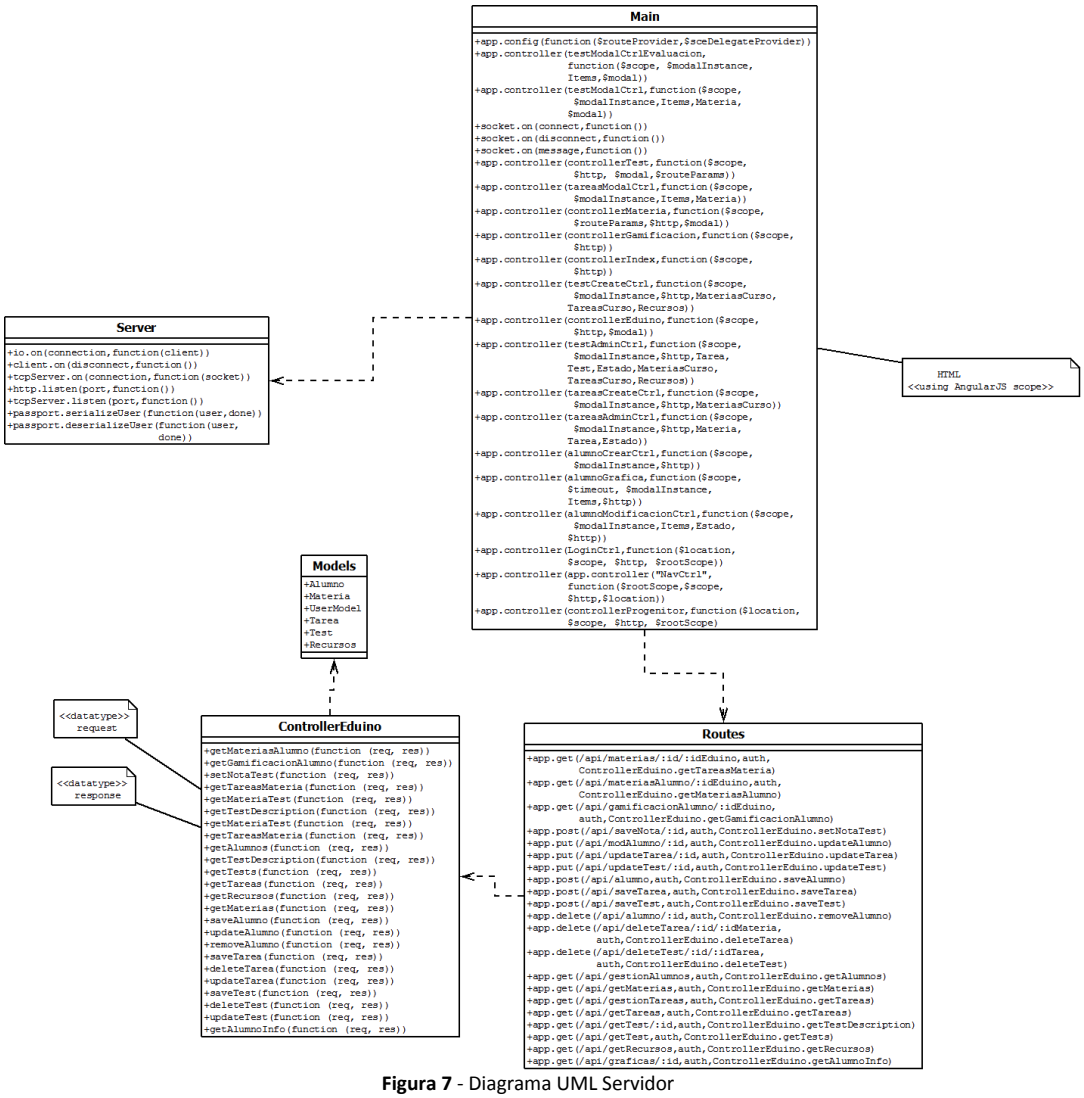


Figura 7 - Diagrama UML Servidor

Para entender mejor el funcionamiento de ambos diagramas se procede a realizar un diagrama de secuencia que ilustre el funcionamiento de la aplicación en su estado final con el alumno. Con este ejemplo se muestra el recorrido que se produce en cada interacción con el sistema y sus componentes.

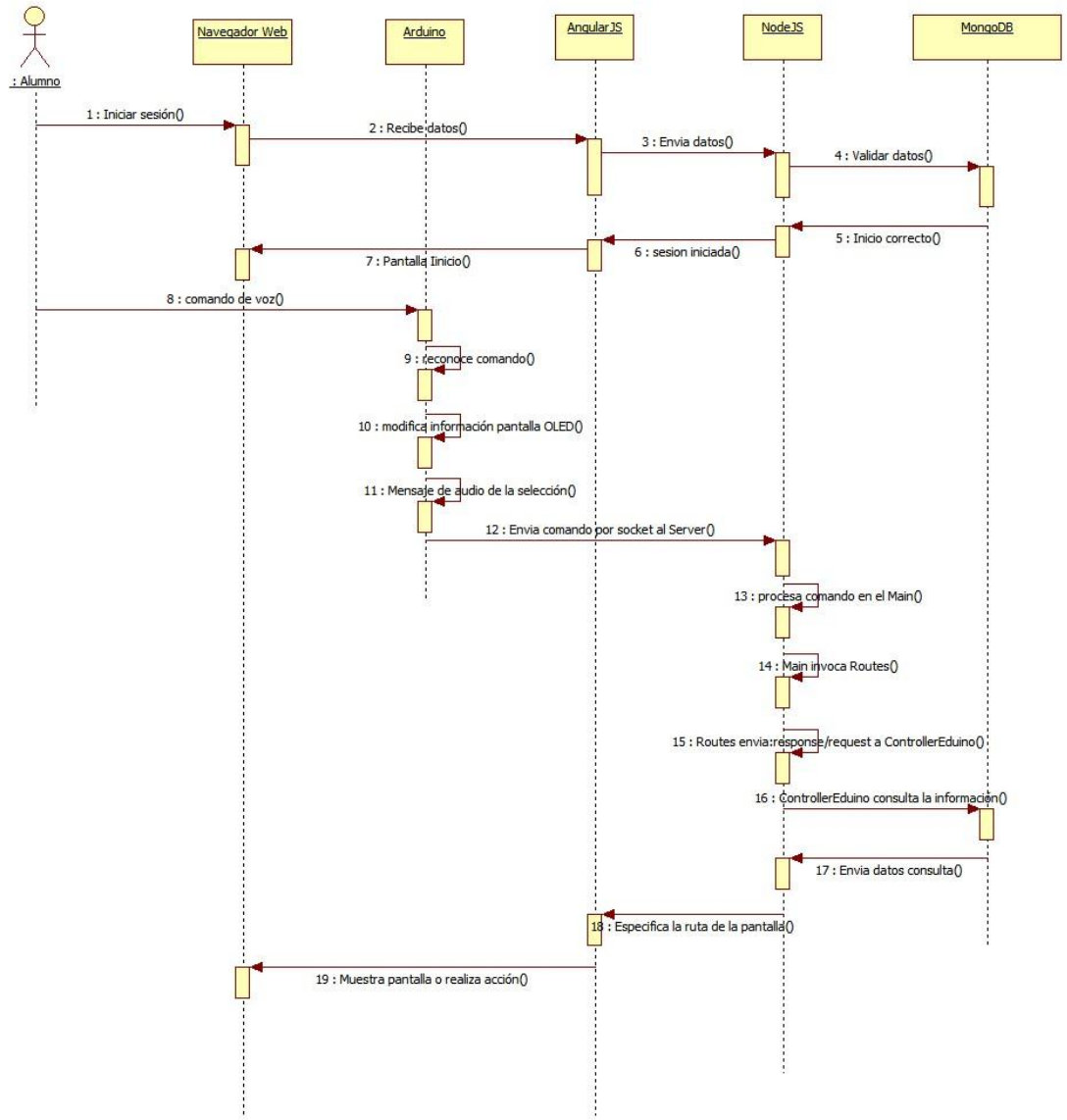


Figura 8 - Diagrama de secuencia de la Arquitectura UML

2.5– Casos de uso

Debido a que no se quiere aumentar la complejidad del proyecto en base al tiempo del que se dispone, se especifica que las gestiones de un posible actor super administrador de la aplicación las realiza el actor **profesor**. Del mismo modo, se asume que únicamente se define un profesor que gestione las diferentes materias que ya se establecerán fijadas en la BD. Quedando a futuras ampliaciones la implementación de estos requerimientos.

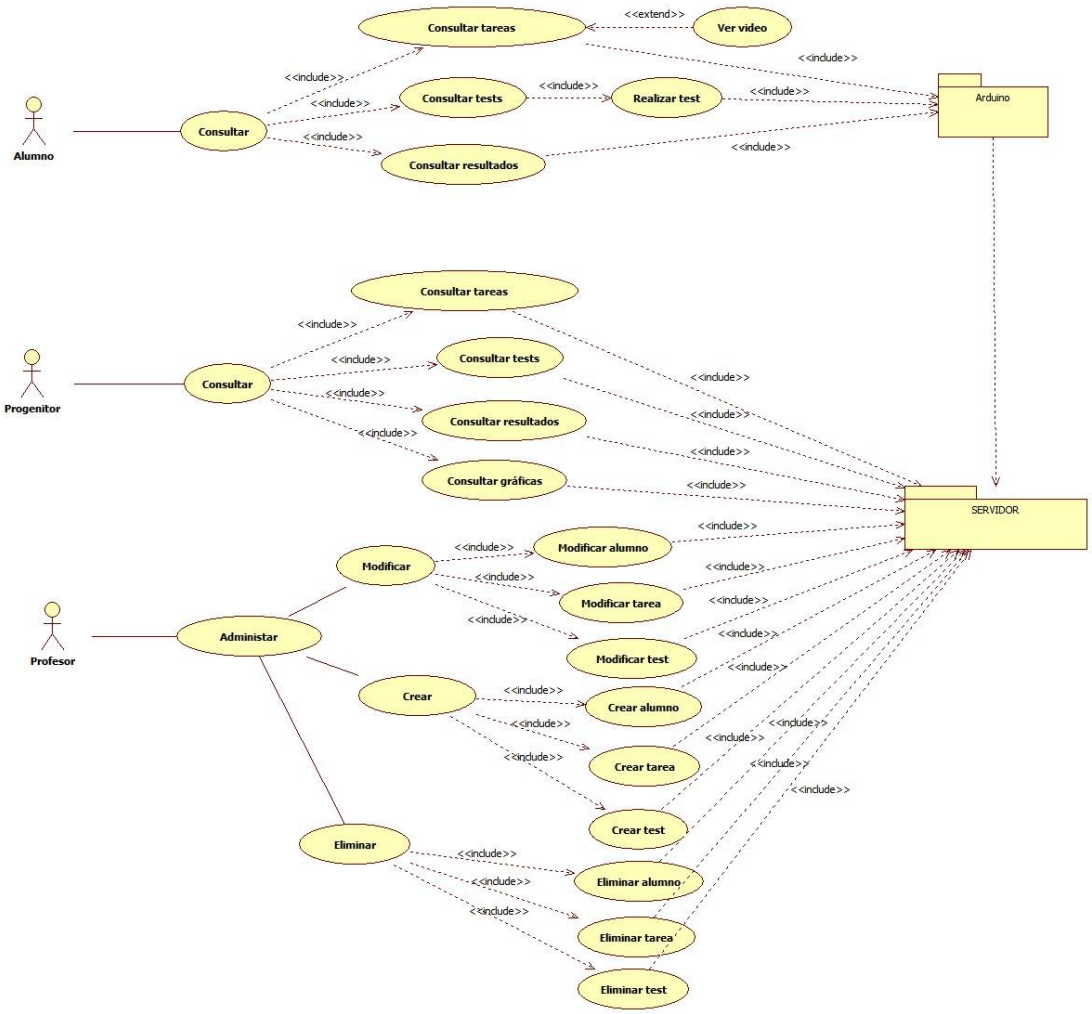


Figura 9 - Casos de uso

ID: CU-01	Crear alumno	
Descripción	El profesor quiere crear un nuevo alumno	
Actores	Profesor	
Pre-condición	El profesor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El profesor clica el icono de alumnos y crear un nuevo alumno.
	2	El sistema muestra un formulario con los diferentes campos disponibles a rellenar del usuario.
	3	El profesor rellena los datos y hace clic en Guardar.
	4	El sistema verifica que los datos introducidos son válidos
	5	Si los datos son válidos el sistema guarda en la base de datos el nuevo alumno y muestra el resultado en pantalla.

Post-condición	-El profesor crea el nuevo alumno en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	3	El profesor cancela la operación y no guarda el nuevo alumno.
	4	Si los datos no son válidos el sistema muestra un mensaje de error y vuelve al paso 2.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-02	**Crear tarea**	
Descripción	El profesor quiere crear una nueva tarea	
Actores	Profesor	
Pre-condición	El profesor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El profesor clica el icono de tareas y crear una nueva tarea.
	2	El sistema muestra un formulario con los diferentes campos disponibles a rellenar para la tarea.
	3	El profesor rellena los datos y hace clic en Guardar.
	4	El sistema verifica que los datos introducidos son válidos.
	5	Si los datos son válidos el sistema guarda en la base de datos la nueva tarea y muestra el resultado en pantalla.
Post-condición	-El profesor crea una nueva tarea en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	3	El profesor cancela la operación y no guarda la nueva tarea.
	4	Si los datos no son válidos el sistema muestra un mensaje de error y vuelve al paso 2.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-03	**Crear test**	
Descripción	El profesor quiere crear un nuevo test	
Actores	Profesor	
Pre-condición	El profesor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El profesor clica el icono de test y añadir un nuevo test.
	2	El sistema muestra un formulario con los diferentes campos disponibles a rellenar para el test.
	3	El profesor rellena los datos y hace clic en Guardar.
	4	El sistema verifica que los datos introducidos son

		válidos.
	5	Si los datos son válidos el sistema guarda en la base de datos el nuevo test y muestra el resultado en pantalla.
Post-condición	-El profesor crea un nuevo test en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	3	El profesor cancela la operación y no guarda el nuevo test.
	4	Si los datos no son válidos el sistema muestra un mensaje de error y vuelve al paso 2.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-04	**Eliminar alumno**	
Descripción	El profesor quiere eliminar un alumno existente	
Actores	Profesor	
Pre-condición	El profesor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El profesor clica el icono de alumnos.
	2	El sistema muestra una con los diferentes alumnos registrados.
	3	El profesor selecciona el alumno a borrar
	4	El sistema pide confirmación de la eliminación
	5	Si la confirmación es correcta el sistema muestra la lista de alumnos sin el alumno eliminado.
Post-condición	-El profesor elimina un alumno en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	4	El profesor cancela la operación y no elimina al alumno.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-05	**Eliminar tarea**	
Descripción	El profesor quiere eliminar una tarea	
Actores	Profesor	
Pre-condición	El profesor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El profesor clica el icono de tareas.
	2	El sistema muestra una lista con las diferentes tareas.
	3	El profesor selecciona la tarea a borrar
	4	El sistema pide confirmación de la eliminación
	5	Si la confirmación es correcta el sistema muestra la lista de tareas sin la tarea eliminada.
Post-condición	-El profesor elimina una tarea en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	4	El profesor cancela la operación y no elimina la tarea.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-06	Eliminar test	
Descripción	El profesor quiere eliminar un test	
Actores	Profesor	
Pre-condición	El profesor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El profesor clica el icono de test.
	2	El sistema muestra una lista con los diferentes test.
	3	El profesor selecciona el test a borrar
	4	El sistema pide confirmación de la eliminación
	5	Si la confirmación es correcta el sistema muestra la lista de test sin el test eliminado.
Post-condición	-El profesor elimina un test en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	4	El profesor cancela la operación y no elimina el test.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-07	Modificar alumno	
Descripción	El profesor quiere modificar un alumno	
Actores	Profesor	
Pre-condición	El profesor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El profesor clica el icono de alumnos.
	2	El sistema muestra una lista con los diferentes alumnos registrados.
	3	El profesor selecciona el alumno a modificar
	4	El sistema muestra la información del alumno seleccionado.
	5	El profesor realiza la modificación y selecciona Guardar.
Post-condición	-El profesor modifica un alumno en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	5	El profesor cancela la operación y no modifica el alumno.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-08	Modificar tarea	
Descripción	El profesor quiere modificar una tarea	
Actores	Profesor	
Pre-condición	El profesor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El profesor clica el icono de tareas.
	2	El sistema muestra una lista con las diferentes tareas.

	3	El profesor selecciona la tarea a modificar
	4	El sistema muestra la información de la tarea seleccionada
	5	El profesor realiza la modificación y selecciona Guardar.
Post-condición	-El profesor modifica una tarea en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	5	El profesor cancela la operación y no modifica la tarea.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-09	**Modificar test**	
Descripción	El profesor quiere modificar un test	
Actores	Profesor	
Pre-condición	El profesor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El profesor clica el icono de test.
	2	El sistema muestra una lista con los diferentes test.
	3	El profesor selecciona el test a modificar
	4	El sistema muestra la información del test seleccionado
	5	El profesor realiza la modificación y selecciona Guardar.
Post-condición	-El profesor modifica un test en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	5	El profesor cancela la operación y no modifica el test.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-10	**Consultar test**	
Descripción	El progenitor quiere consultar la lista de test	
Actores	Progenitor	
Pre-condición	El progenitor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El progenitor clica el icono de test.
	2	El sistema muestra una lista con los diferentes test.
Post-condición	No se requiere.	
Flujo alternativo	No se requiere.	
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-11	**Consultar tareas**	
Descripción	El progenitor quiere consultar la lista de tareas	
Actores	Progenitor	
Pre-condición	El progenitor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**

	1	El progenitor clica el icono de Inicio.
	2	El sistema muestra una lista de materias del alumno
	3	El progenitor escoge una materia
	4	El sistema muestra la lista de tareas de esa materia.
Post-condición	No se requiere.	
Flujo alternativo	No se requiere.	
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-12	**Consultar resultados**	
Descripción	El progenitor quiere consultar la lista de resultados	
Actores	Progenitor	
Pre-condición	El progenitor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El progenitor clica el icono de resultados.
	2	El sistema muestra una lista con los resultados de los diferentes test.
Post-condición	No se requiere.	
Flujo alternativo	No se requiere.	
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-13	**Consultar gráficas**	
Descripción	El progenitor quiere consultar las gráficas del alumno en las materias.	
Actores	Progenitor	
Pre-condición	El progenitor se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El progenitor clica el icono de gráficas.
	2	El sistema muestra pestañas con las diferentes materias del alumno
	3	El progenitor escoge una materia para visualizar las estadísticas de los test de cada tarea.
	4	El sistema muestra un gráfico de barras en función de los resultados obtenidos del alumno.
Post-condición	No se requiere.	
Flujo alternativo	No se requiere.	
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-14	**Consultar tareas**	
Descripción	El alumno quiere consultar la lista de tareas de una materia.	
Actores	Alumno	
Pre-condición	El alumno se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**

	1	El alumno introduce el comando de voz “Inicio”.
	2	El sistema muestra las materias disponibles.
	3	El alumno escoge una materia.
	4	El sistema muestra la lista de tareas para esa materia.
	5	El alumno introduce el comando de voz “Ver tarea N”
	6	El sistema muestra al alumno la información de la tarea.
Post-condición	No se requiere.	
Flujo alternativo	No se requiere.	
Comentarios	-Los comandos de voz de las materias están determinados por las que disponga cada alumno específico. La lista de materias presentada es: “Historia”, “Ciencias naturales” y “Música”. -En el comando de voz “Consultar tarea N”, N representa el número de la tarea en la lista.	

ID: CU-15	**Ver video**	
Descripción	El alumno quiere reproducir el video de una tarea	
Actores	Alumno	
Pre-condición	-El alumno se ha identificado correctamente en el sistema. -El alumno se encuentra en el paso 4 del caso de uso “Consultar tareas”.	
Flujo principal	**Paso**	**Acción**
	1	El alumno introduce el comando de voz “Reproducir”.
	2	El sistema reproduce el video de la tarea.
Post-condición	No se requiere.	
Flujo alternativo	**Paso**	**Acción**
	2	El alumno introduce el comando de voz “Parar” y para la reproducción de este.
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-16	**Consultar resultados**	
Descripción	El alumno quiere consultar la lista de resultados	
Actores	Alumno	
Pre-condición	El alumno se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El alumno introduce el comando de voz “Resultados”.
	2	El sistema muestra una lista con los resultados de los diferentes test.
Post-condición	No se requiere.	
Flujo alternativo	No se requiere.	
Comentarios	No se declaran comentarios para este caso de uso.	

ID: CU-17	Consultar test	
Descripción	El alumno quiere consultar la lista de test	
Actores	Alumno	
Pre-condición	El alumno se ha identificado correctamente en el sistema.	
Flujo principal	**Paso**	**Acción**
	1	El alumno introduce el comando de voz “Test”.
	2	El sistema muestra una lista con los test disponibles.
	3	El alumno introduce el comando de voz “Ver test N”
	4	El sistema muestra al alumno la información del test.
Post-condición	No se requiere.	
Flujo alternativo	No se requiere.	
Comentarios	En el comando de voz “Test Materia N”, N representa el número del test en la lista.	

ID: CU-18	Realizar test	
Descripción	El alumno quiere realizar un test	
Actores	Alumno	
Pre-condición	-El alumno se ha identificado correctamente en el sistema. -El alumno se encuentra en el paso 4 del caso de uso “Consultar test”.	
Flujo principal	**Paso**	**Acción**
	1	El alumno introduce el comando de voz “Empezar test.”
	2	El alumno introduce el comando de voz “Pregunta N” y a continuación introduce el comando de voz “A”, “B” o “C” según la opción que considera correcta.
	3	El alumno introduce el comando de voz “Corregir”
	4	El sistema muestra los resultados, la nota obtenida y el feedback con las respuestas correctas.
Post-condición	-El alumno obtiene una medalla de acuerdo al resultado. -Se guarda el resultado de la evaluación del test en el sistema.	
Flujo alternativo	**Paso**	**Acción**
	1	El alumno decide cancelar el inicio del test
	3	El alumno cancela la comprobación de resultados
Comentarios	-En el comando de voz “Pregunta N”, N representa el número de la pregunta en el test.	

Para simplificar la representación de los diagramas de secuencia, se asume que todos los actores se encuentran en la pantalla requerida para llevar a cabo el caso de uso.

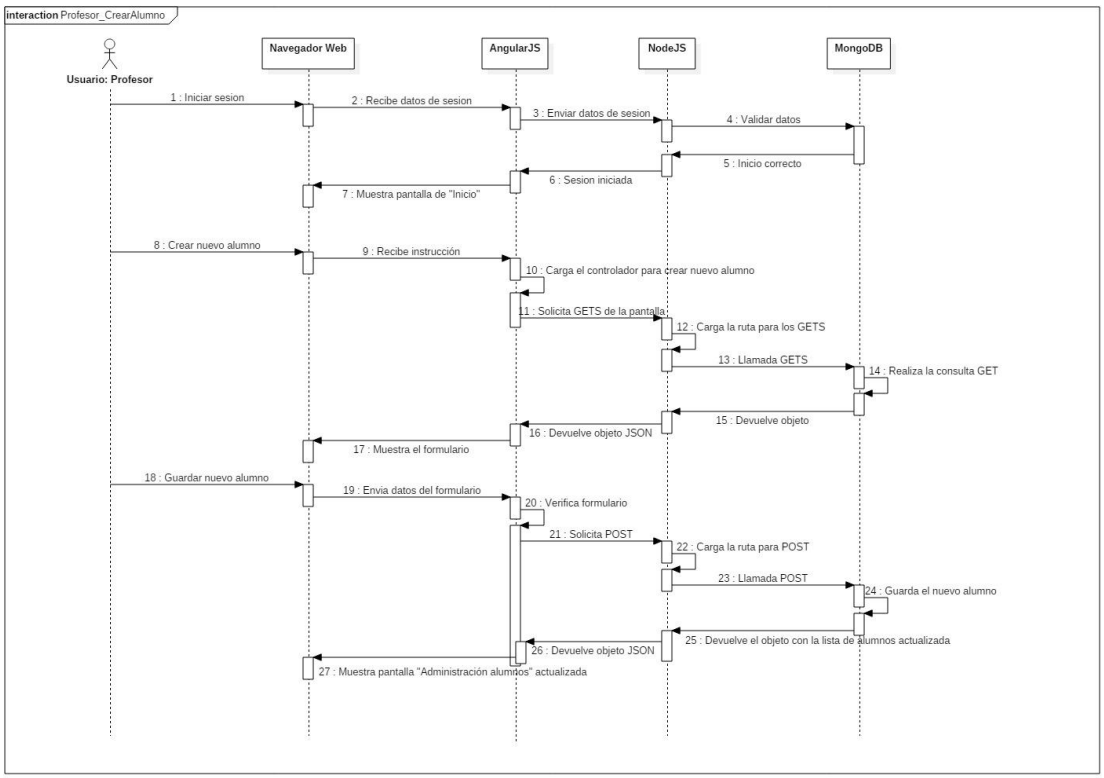


Figura 10 - Diagrama de secuencia: Profesor - Crear alumno

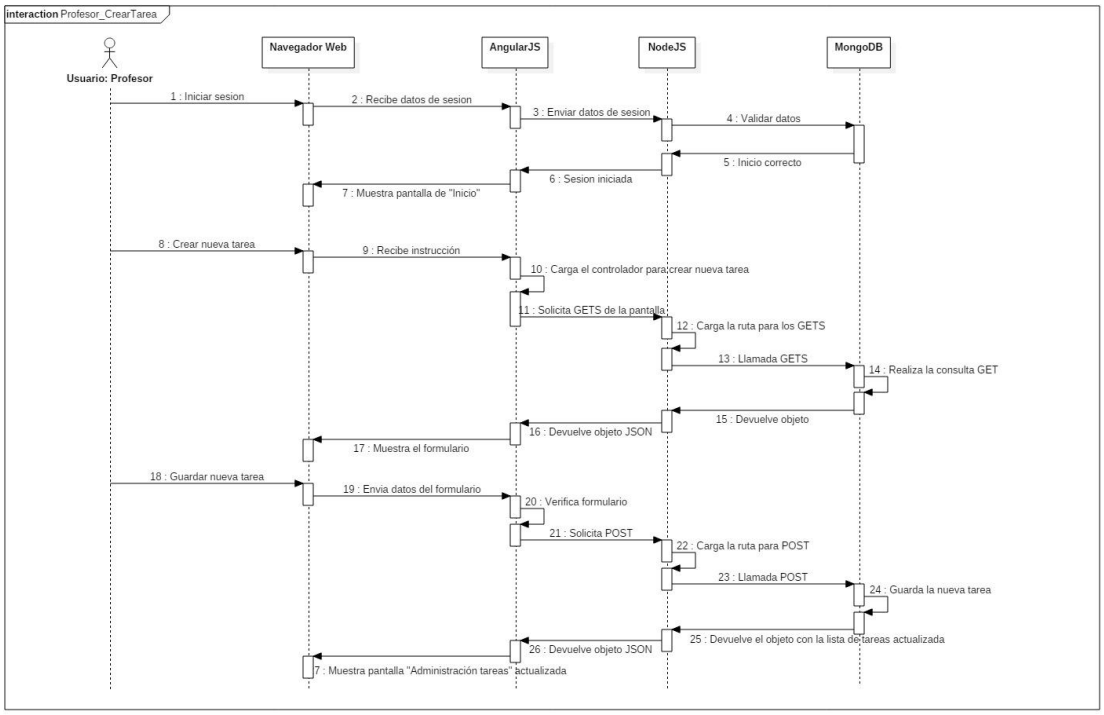


Figura 11 - Diagrama de secuencia: Profesor - Crear tarea

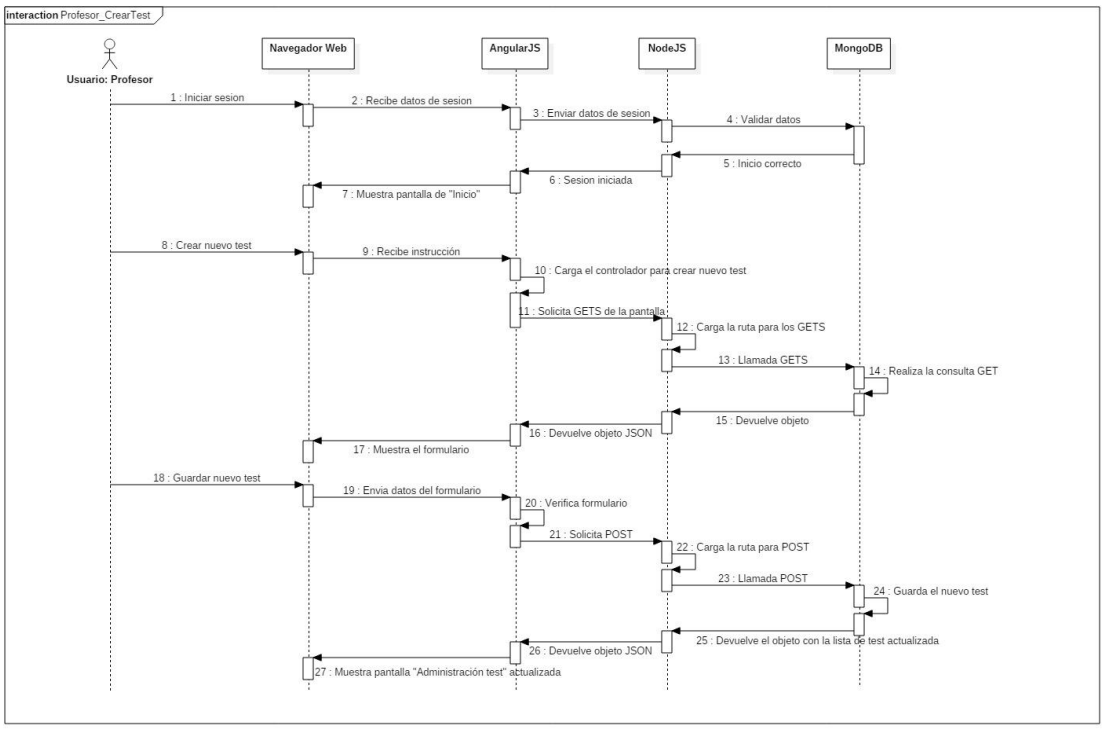


Figura 12 - Diagrama de secuencia: Profesor - Crear test

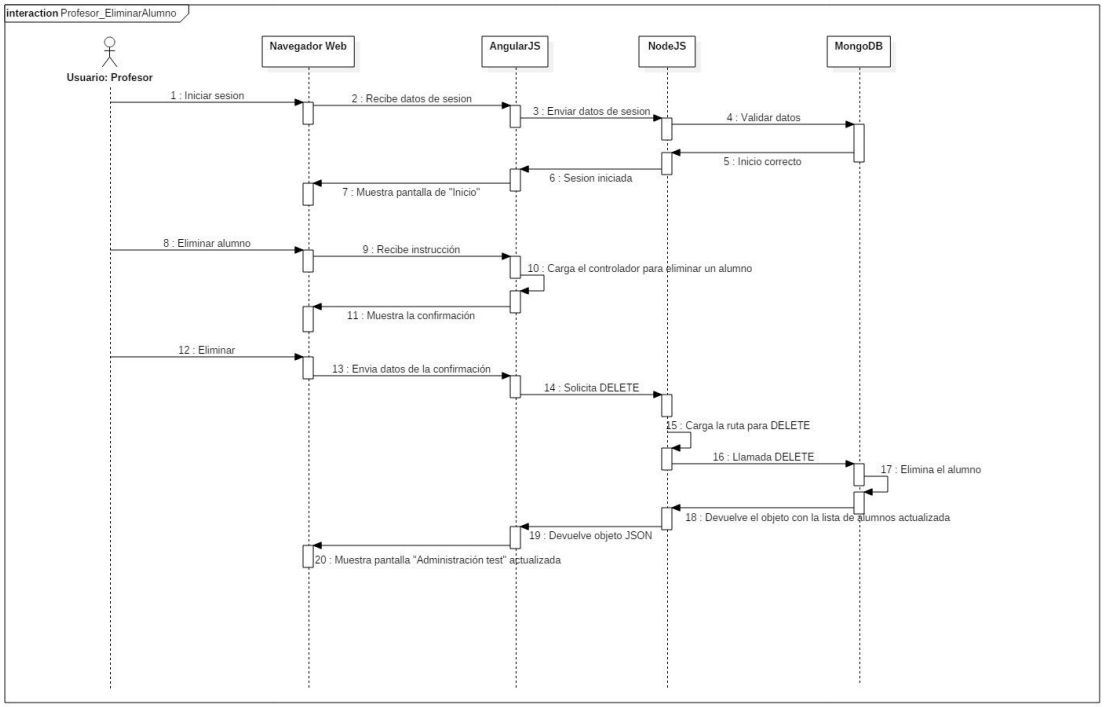


Figura 13 - Diagrama de secuencia: Profesor - Eliminar alumno

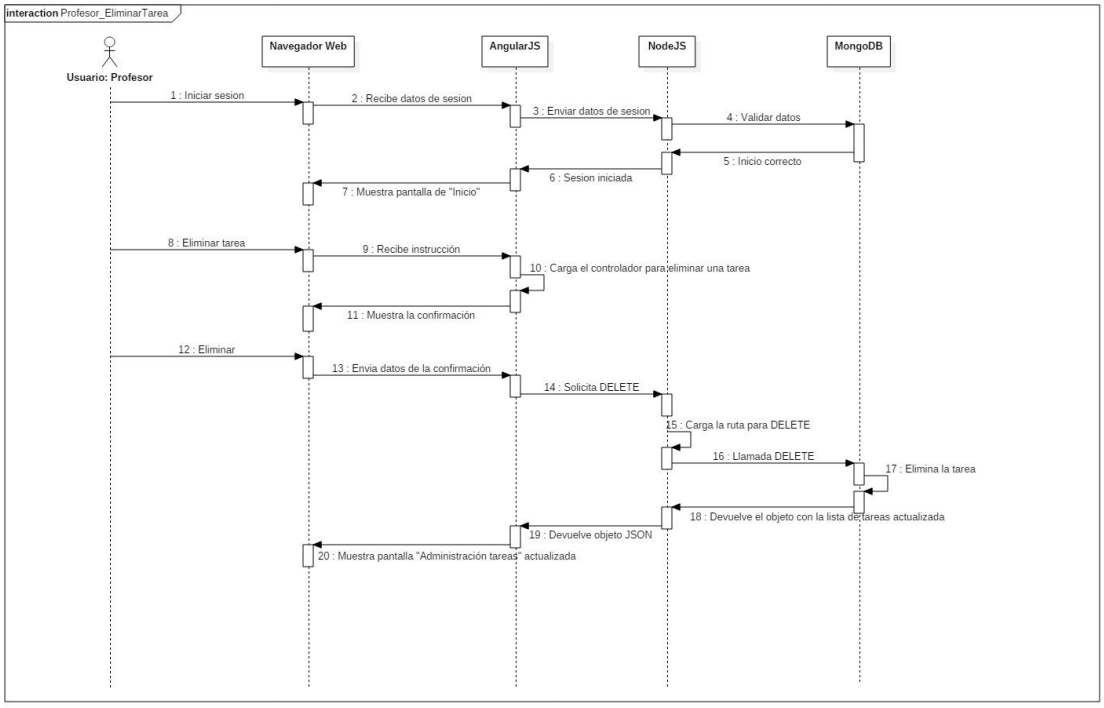


Figura 14 - Diagrama de secuencia: Profesor - Eliminar tarea

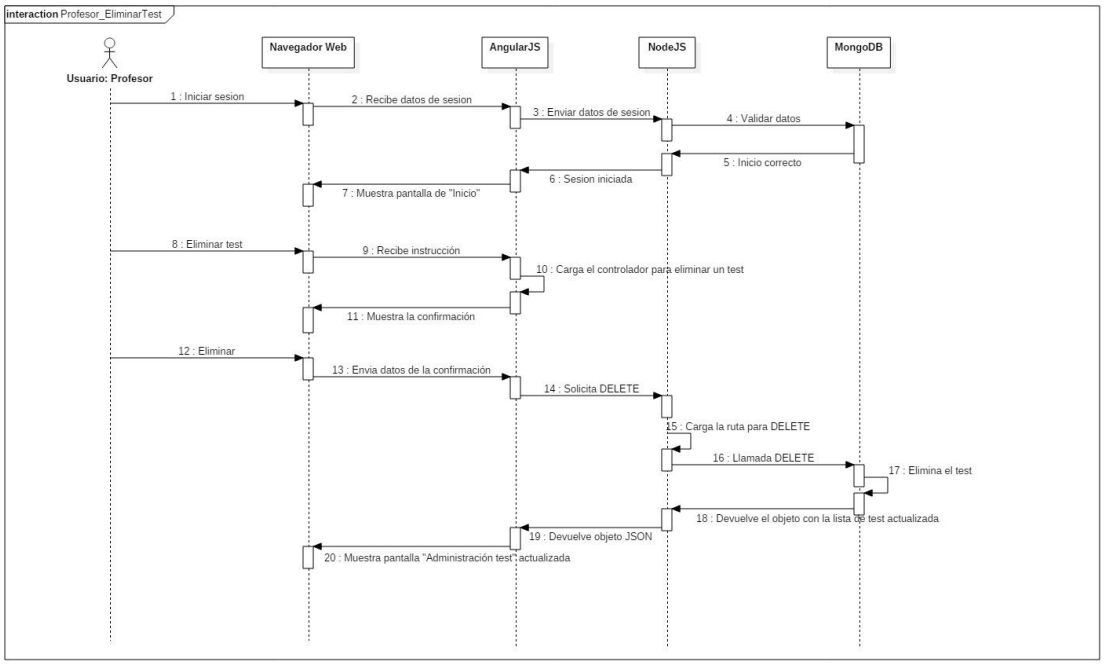


Figura 15 - Diagrama de secuencia: Profesor - Eliminar test

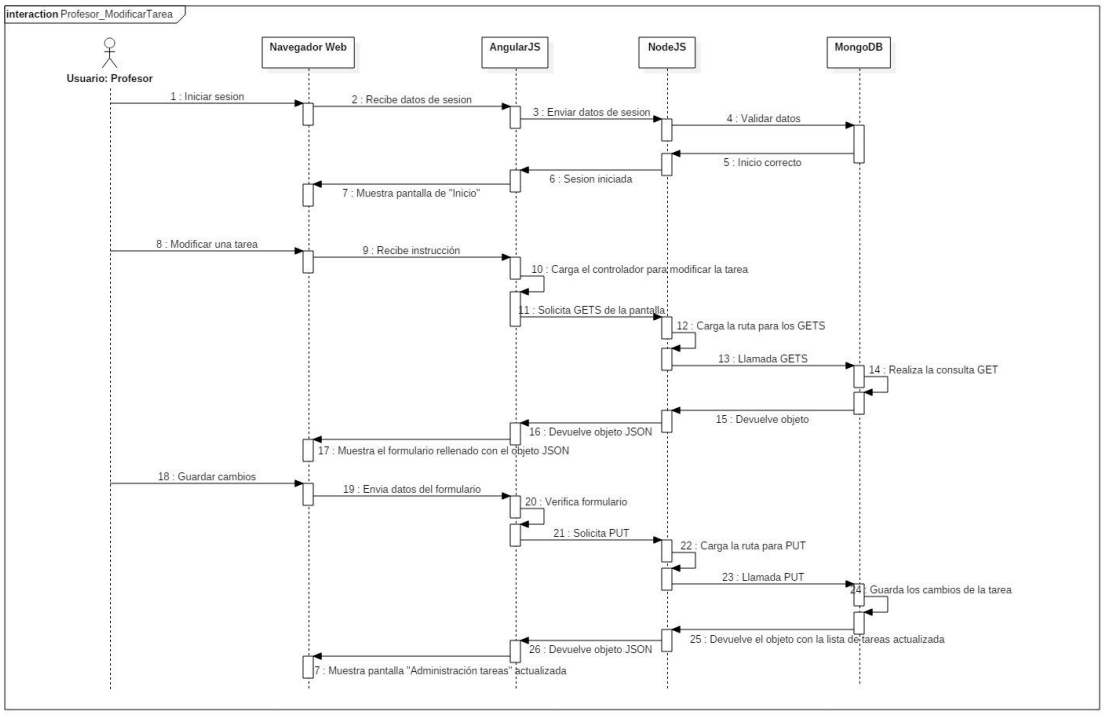


Figura 16 - Diagrama de secuencia: Profesor – Modificar tarea

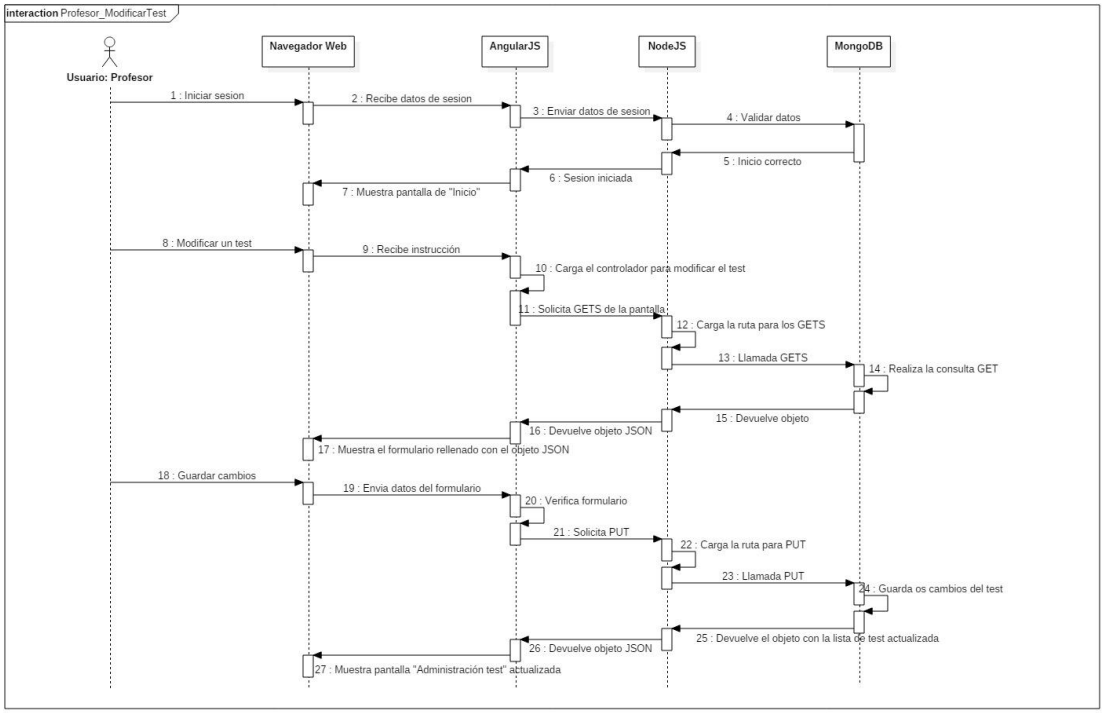


Figura 17 - Diagrama de secuencia: Profesor – Modificar test

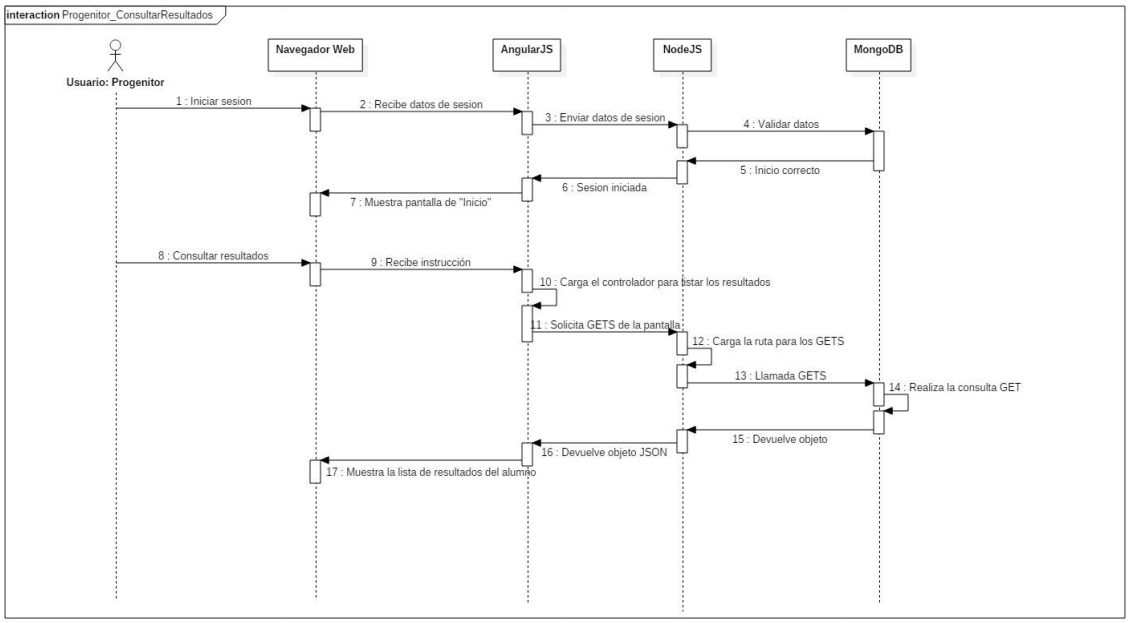


Figura 18 - Diagrama de secuencia: Progenitor – Consultar resultados

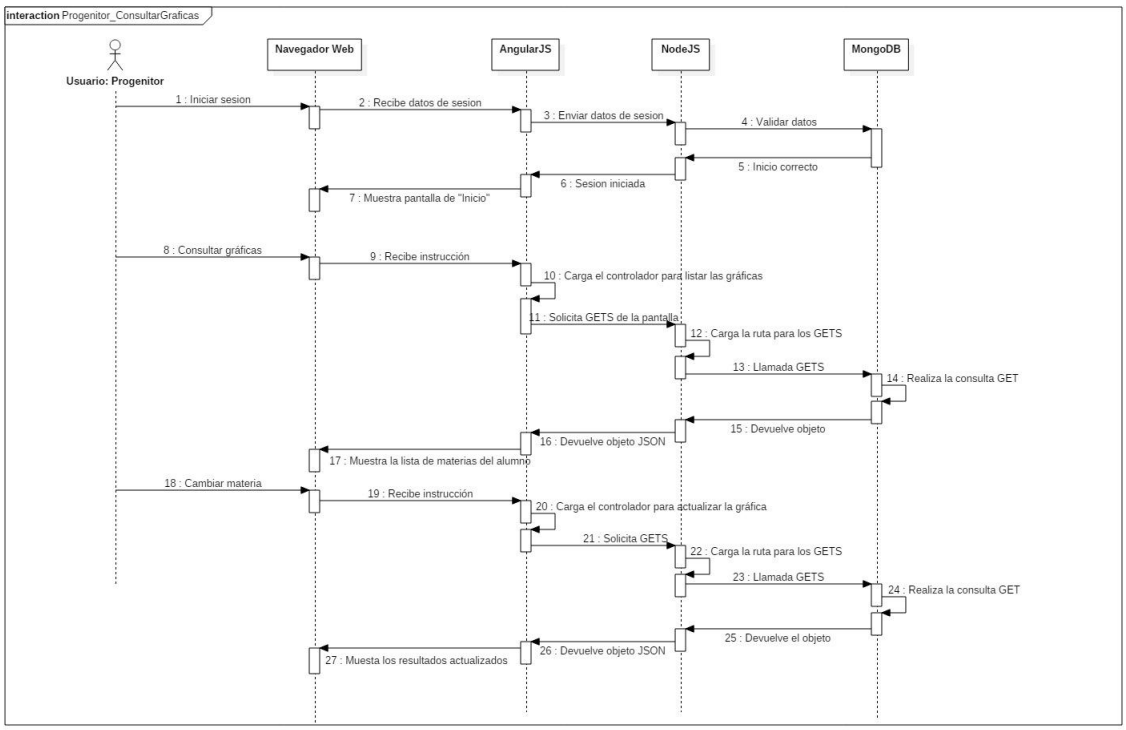


Figura 19 - Diagrama de secuencia: Progenitor – Consultar gráficas

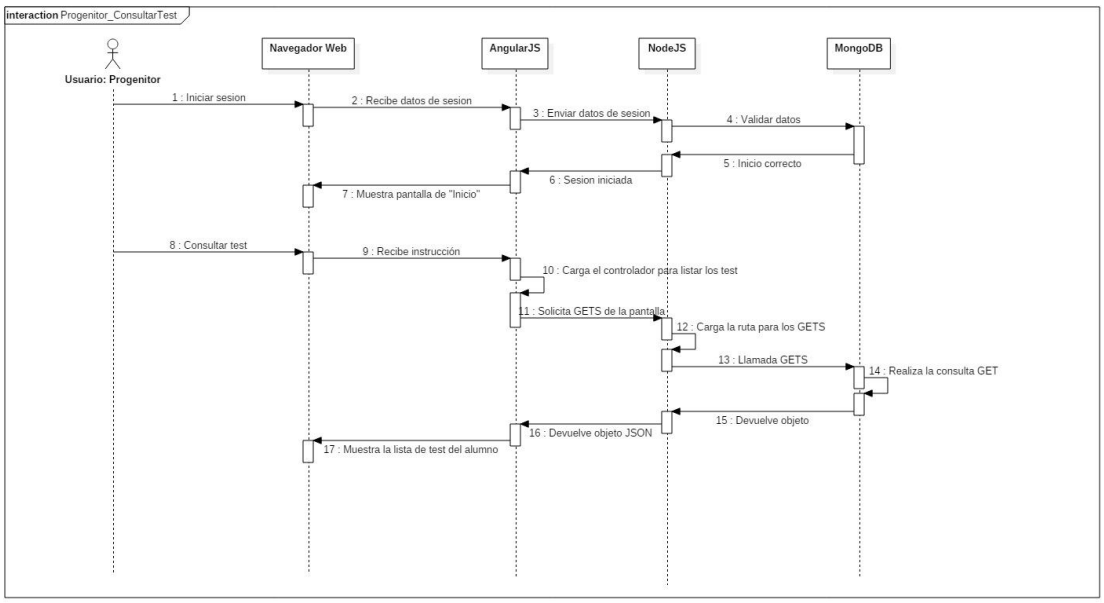


Figura 20 - Diagrama de secuencia: Progenitor – Consultar test

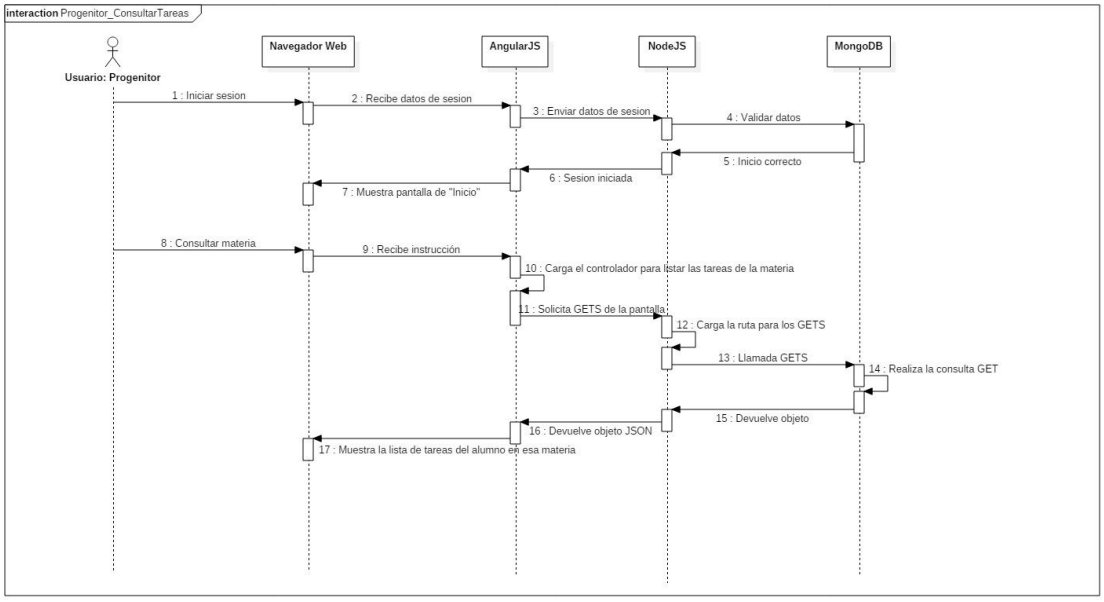


Figura 21- Diagrama de secuencia: Progenitor – Consultar tareas

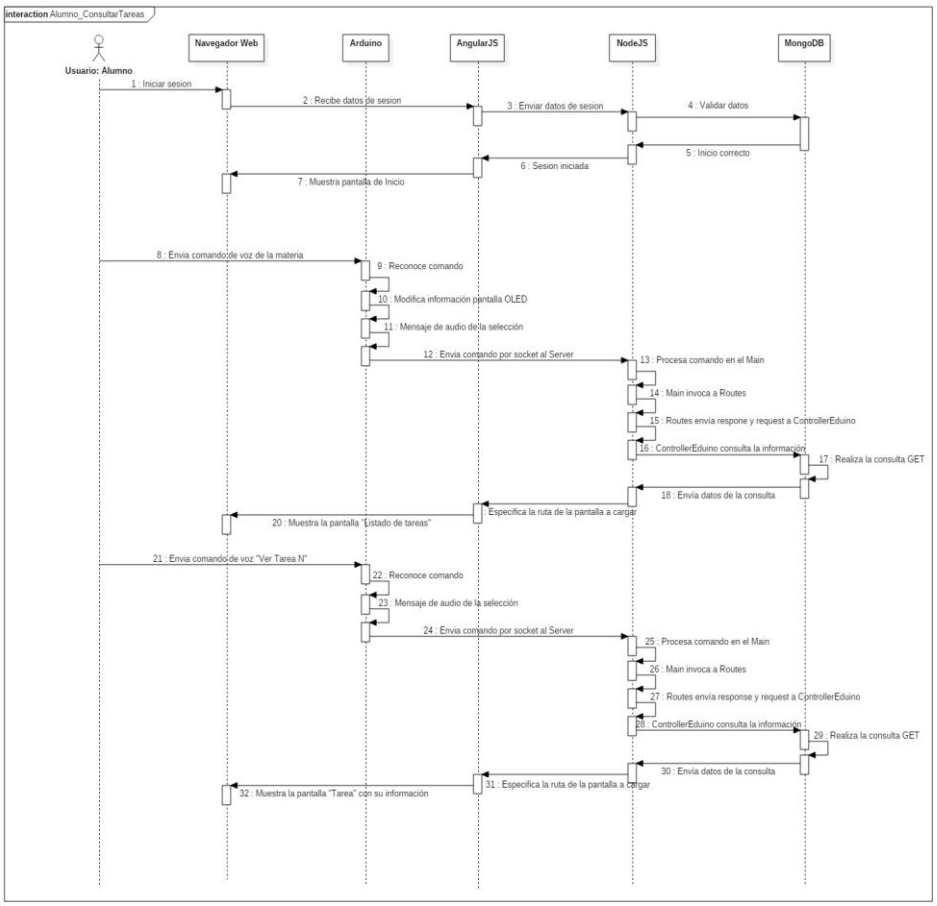


Figura 22 - Diagrama de secuencia: Alumno – Consultar tareas

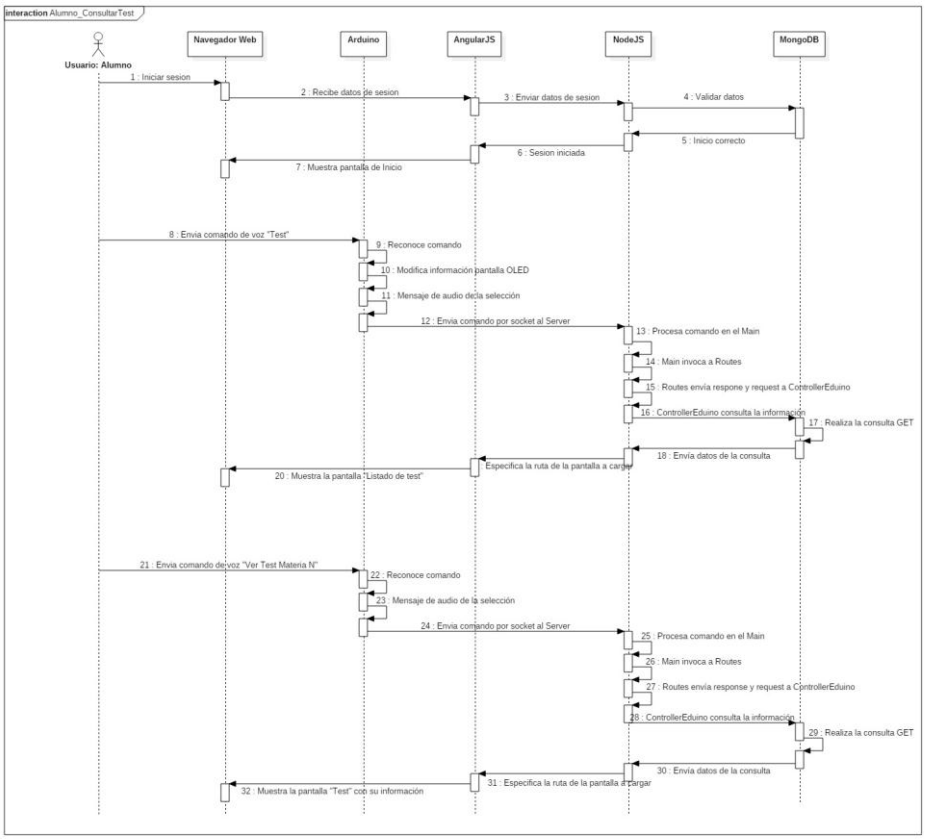


Figura 23 - Diagrama de secuencia: Alumno – Consultar test

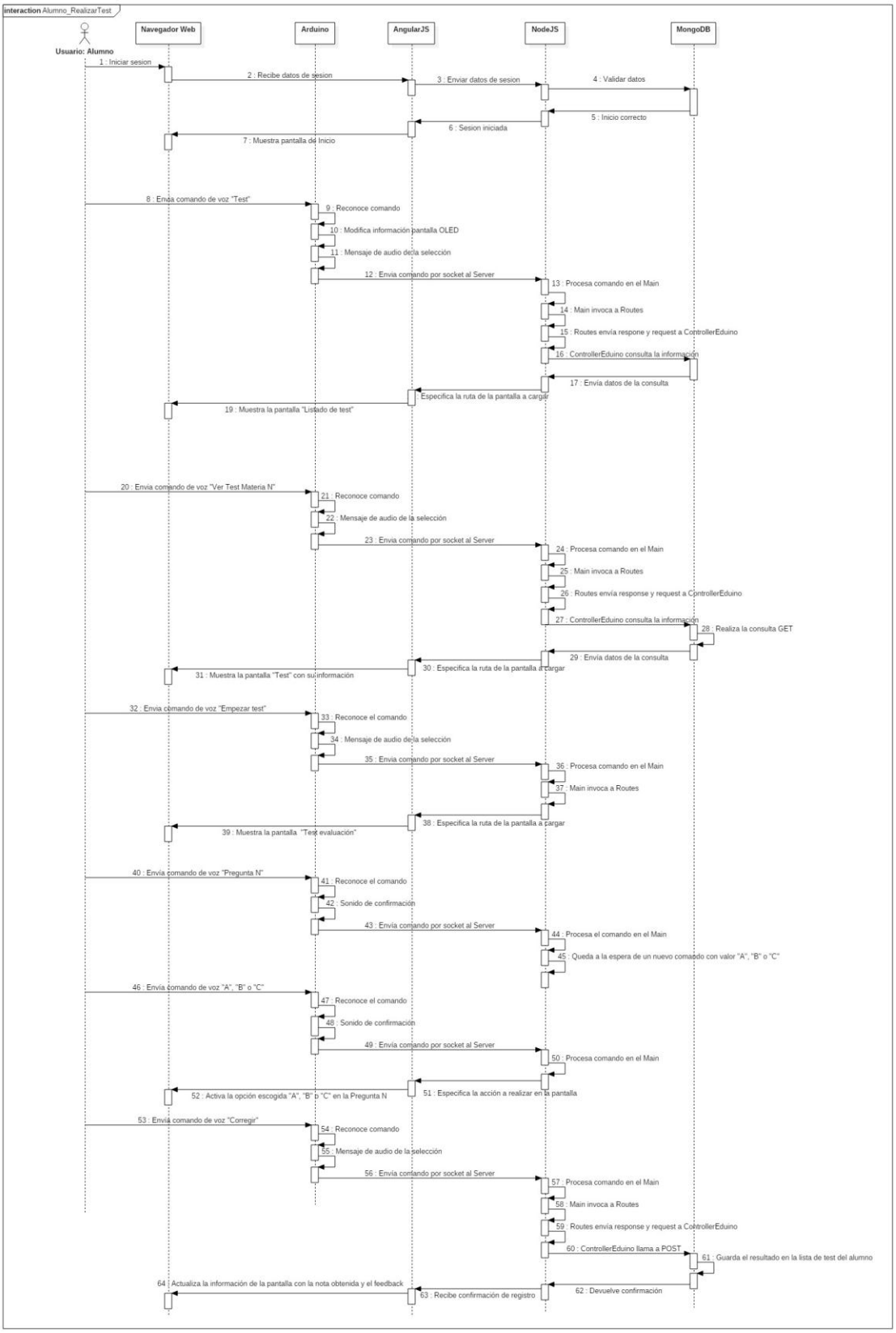


Figura 24 - Diagrama de secuencia: Alumno - Realizar test

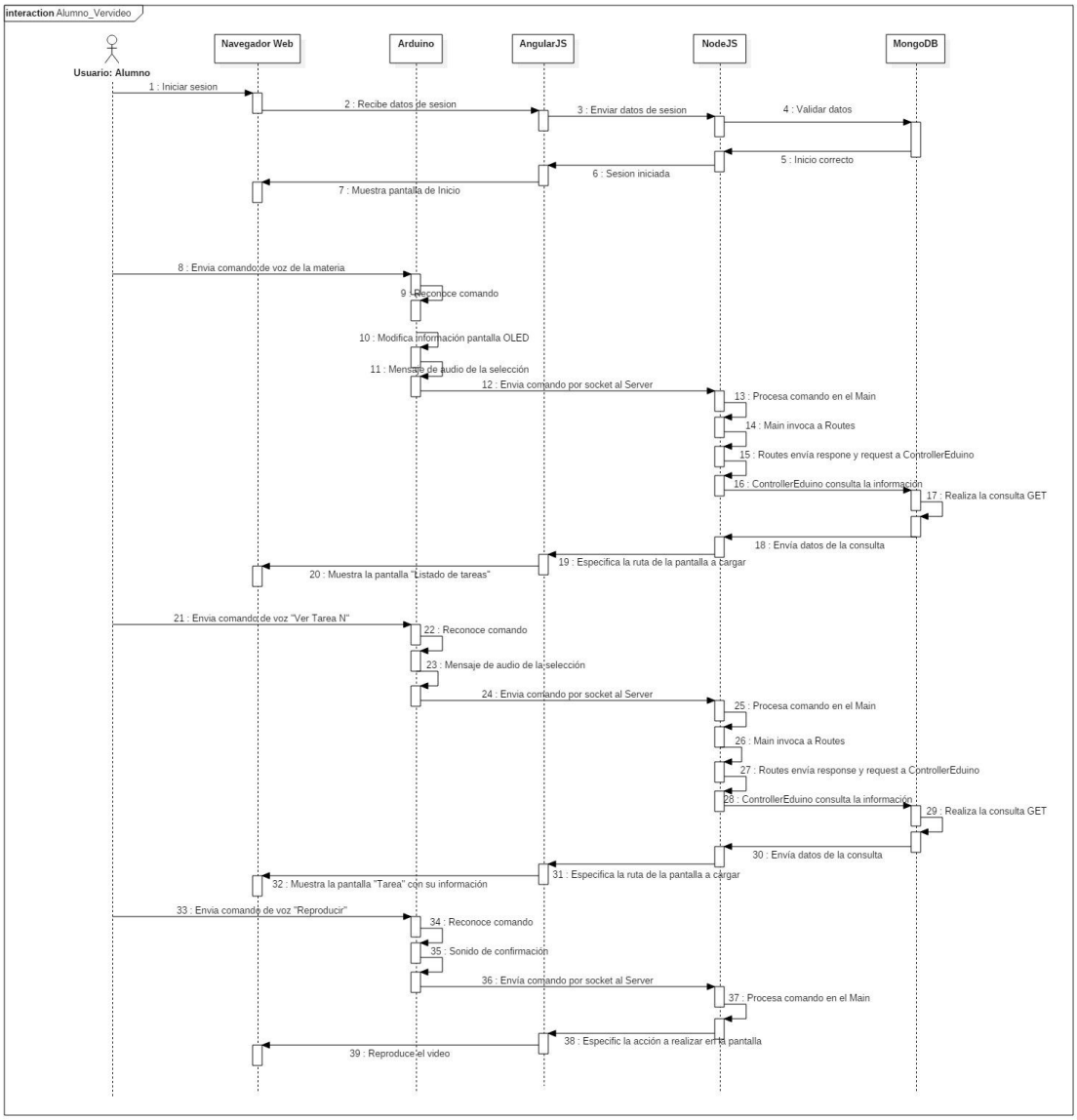


Figura 25 - Diagrama de secuencia: Alumno – Ver video

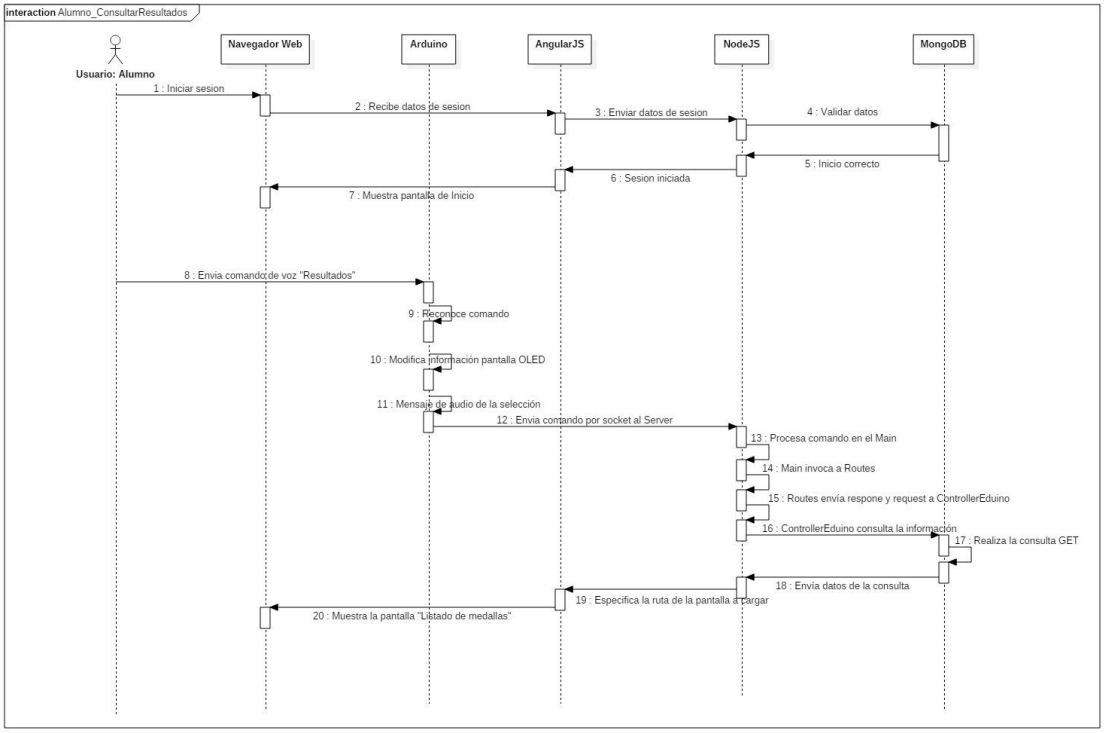


Figura 26 - Diagrama de secuencia: Alumno – Consultar resultados

2.6 - Instalación y preparación entorno trabajo

En esta fase se especifican los requerimientos que se deben tener en cuenta en la preparación previa al desarrollo del proyecto

2.6.1 - Cliente.

Esta capa utiliza:

- **jQuery:** Biblioteca de JavaScript que permite de una forma sencilla la interacción con documentos HTML**.**

- **AngularJS:** Framework utilizado para la presentación del tratamiento de la información representada por el servidor.

- **AngularJS routes**: Permite que el navegador pueda entender cómo cargar los enlaces de las diferentes rutas de la aplicación.

- **UI Bootstrap**: Una serie de componentes de bootstrap basados en AngularJS que permiten el uso de ventanas modales.

- **HTML5:** Lenguaje de marcado que permite la definición de la estructura de la página.

- **CSS3:** Utilizado para la presentación del documento HTML5 mediante hojas de estilo.

- **Bootstrap:** Framework o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web basados en diseño responsive.

- **FontAwesome:** Proporciona un conjunto de iconos vectoriales de sencillo uso.

- **Navegador web:** Necesario para la presentación y acceso a la Web de la aplicación.

- **Socket.io**: Librería que permite manejar eventos en tiempo real.

2.6.2 – Servidor

Se define por:

- **NodeJS:** Librería y entorno Javascript en tiempo de ejecución multiplataforma para la capa del servidor, basado en eventos.

- **MongoDB:** Sistema de base de datos NoSQL orientado a documentos JSON que nos facilita el Trabajo debido a su agilidad y escalabilidad.

- **Socket.io**: Librería que permite manejar eventos en tiempo real.

- **ExpressJS:** Infraestructura de aplicaciones web Node.js que proporciona un conjunto sólido de características para las aplicaciones web y móviles.

- **HTTP**: Librería que permite a NodeJS trabajar con el protocolo HTTP utilizado para la transferencia de datos a la Web.

- **Mongoose**: La utilización de esta librería permite la definición de esquemas para el modelado de datos de la aplicación, facilita la utilización de diversos tipos de consultas a la base de datos y enlaza NodeJS con MongoDB.

- **bodyParser**: Su uso permite parsear documentos JSON.

- **net:** Concede a NodeJS la posibilidad de creación de un servidor TCP que reciba los comandos enviados por Arduino.

- **Passport:** permite implementar un sistema de login con roles de usuario.

- **ngCookies:** mantiene en sesión una cookie que almacena la información del login durante la interacción con las diferentes pantallas.

- **Nvd3:** permite la representación de gráficos.

- **ngMessages:** control de errores de los datos en los formularios.

2.6.3 – Arduino

Los requerimientos de esta capa son los siguientes:

- **Microcontrolador:** Arduino MEGA.

- **Display Module 12864 0.96 Inch White SPI OLED**: Pantalla que permite la visualización de contenido bitmap enviado desde Arduino.

- **Ethernet Shield W5100**: Este módulo permite dotar a placas Arduino que no disponen de conexión a Internet de esta funcionalidad.

- **EasyVR Arduino Shield v3**: Módulo que permite implementar un reconocimiento de voz.

- **Altavoz de audio 8Ω:** Componente utilizado para la reproducción de las notificaciones o avisos del módulo EasyVR3.

- **Micrófono:** Permite la recepción de comandos de voz para su posterior procesamiento por el módulo EasyVR3.

- **Alimentador:** Alimentador de corriente de 12V DC - 1A.

- **Librerías para los diferentes módulos:**

    - **Arduino MEGA**
        - SPI.h
    - **Display Module 12864 0.96 Inch White SPI OLED**
        - Adafruit_GFX.h
        - Adafruit_SSD1306.h
        - Wire.h
    - **Ethernet Shield W5100**
        - Ethernet.h
    - **EasyVR Arduino Shield v3**
        - SoftwareSerial.h
        - EasyVR.h
        - Arduino.h

2.7 – Implementación Arduino

2.7.1 – Módulo Ethernet

El microcontrolador utilizado no dispone de conectividad a Internet, con lo que dadas las necesidades del proyecto se dota al sistema de un módulo que otorgue estas

capacidades de comunicación. Esto se consigue mediante el uso de Ethernet Shield, el cual se acopla perfectamente sobre Arduino debido a su funcionalidad Shield, y permite continuar con la utilización de los pines del microcontrolador al mismo tiempo que se hacen uso de las nuevas funcionalidades.

A su vez, se ha utilizado para transmitir al servidor las peticiones que recibe Arduino por parte del usuario mediante el uso de paquetes TCP, y recibir las respuestas que este le facilita, para posteriormente transmitirlas al microcontrolador. Así conseguir una comunicación bidireccional entre Arduino y Servidor.

A continuación, se pasan a definir las líneas de código que permiten la inicialización y configuración de este módulo. En primer lugar, es necesario incluir la librería del módulo para su correcto funcionamiento:

```
// Librería para Ethernet Shield
#include <Ethernet.h>
```

Con la librería incluida se puede empezar a especificar la configuración del dispositivo. Las siguientes líneas de código configuran la IP del dispositivo, la IP del servidor y la dirección MAC que ya viene definida en la shield.

```
byte mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0x6C, 0xFE };
IPAddress ip(192,168,1,20);
IPAddress server(81,184,160,214);
EthernetClient client;
```

El código begin que inicializa el módulo de la clase Ethernet, recibe como parámetros las dos variables anteriormente definidas.

```
Ethernet.begin(mac, ip);
```

2.7.2 – Módulo de reconocimiento de voz

El reconocimiento de voz está compuesto por el módulo EasyVR3 y su shield. La combinación de ambos da como resultado el módulo **EasyVR3 Shield** que permite al igual que Ethernet Shield un acoplamiento que mantiene las funcionalidades y conexiones del microcontrolador original Arduino. Cabe añadir, que este módulo y su shield se distribuyen sin las soldaduras y montaje, por lo tanto es necesaria la soldadura de la placa al módulo para su correcto funcionamiento.

Al mismo tiempo, merece la pena destacar ciertos aspectos previos que se deben tener en cuenta:

- El altavoz utilizado por el módulo debe ser de 8Ω.
- Las herramientas QuickSynthesis y EasyCommander deben ejecutarse en modo administrador.

Además, cómo primera toma de contacto, se debe conocer que el módulo implementa un conjunto de jumpers que permiten la realización de diferentes funcionalidades según la posición de este:

PC: Permite el reconocimiento por parte del programa Easy Commander del módulo y la definición de instrucciones vocales, además de la configuración de los comandos.
UP: En este estado se puede proceder a subir al módulo la tabla de contenido creada mediante la herramienta QuickSynthesis.
SW: Este modo permite el funcionamiento normal del microcontrolador, la utilización de la tabla de sonidos y comandos de voz previamente configurados.

Para su configuración, se empieza definiendo las librerías necesarias:

```
#include "Arduino.h"
#include "SoftwareSerial.h"
#include "EasyVR.h"
```

Realizada la importación se debe proceder a especificar el puerto por el que trabajará el módulo y a su inicialización:

```
SoftwareSerial port(12, 13);
EasyVR easyvr(port);
```

También se especifica mediante una enumeración los grupos y sus comandos disponibles creados cómo se verá más tarde en el apartado **2.9.2 – Configuración del reconocimiento de voz**.

```
enum Groups
{
  GROUP_0  = 0,
  GROUP_1  = 1,
};

enum Group0
{
  G0_EDUINO = 0,
};

enum Group1
{
  G1_HISTORIA = 0,
  G1_MUSICA = 1,
  G1_NATURALES = 2,
  G1_ARRIBA = 3,
  G1_INICIO = 4,
  G1_SALIR = 5,
……
};
```

Para ver el funcionamiento del módulo se puede consultar el **Anexo_EasyVR3Shield**.

2.7.3 – Módulo display OLED

El uso del display permite informar de la situación actual de la aplicación durante el uso de esta. La conexión con Arduino se realiza con los siguientes pines.

GND – GND
5V -- VCC
MOSI – SDA (Pin 5)
CLK – SCL (Pin 6)
DC – DC (Pin 7)
RESET – RES (Pin 4)

Realizada la conexión, se definen las librerías utilizadas por el módulo.

```
//Librerías para pantalla Adafruit OLED
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Habiendo previamente incluido las librerías necesarias, se deben definir e indicar los pines anteriormente mencionados, que serán usados para la comunicación:

```
#define OLED_DC 7
#define OLED_CS 1 //No se usa
#define OLED_CLK 6
#define OLED_MOSI 5
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC,
OLED_RESET, OLED_CS);
```

Las funcionalidades del display que se utilizarán en el proyecto, son la representación de imágenes y texto. El uso de imágenes en el módulo utiliza cómo formato la extensión bitmap (**.bmp**). Por ello, se requiere la conversión de imágenes a este formato, el cual se realiza mediante la herramienta **LCD Assistant**. Una vez generada la conversión se obtiene la imagen codificada. Para su uso se utiliza la siguiente definición:

```
//Es necesario añadir PROGMEM para la correcta
interpretación de la imagen
static const unsigned char PROGMEM Inicio[] =
{ B00000000, B11000000,
  B00000001, B11000000,
……
};
```

Una vez codificada la colección de imágenes, se implementa la función **comandoDisplay** que se encargará de recibir el mensaje de cada sección y mostrarlo en pantalla:

```
//Ejemplo de uso del envío al display
comandoDisplay("Inicio",32,48,Inicio);

void comandoDisplay(String mensaje,int x, int y,const
unsigned char* panel){
  display.clearDisplay();
  display.drawBitmap(40,0, panel, 48, 48, WHITE);
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(x,y);
  display.print(mensaje);
  display.display();
  delay(3000);
}
```

Acabada la definición del uso de texto e imágenes se puede proceder a la inicialización del display con el uso del método **begin**.

```
display.begin(SSD1306_SWITCHCAPVCC);
```

La demostración del funcionamiento de este módulo se puede consultar en formato visual en el **Anexo_DisplayOLED**.

2.8 – Implementación Servidor

2.8.1 – Implementación requerimientos

En primer lugar se debe tener presente la definición del árbol del proyecto con el cual se trabajara.

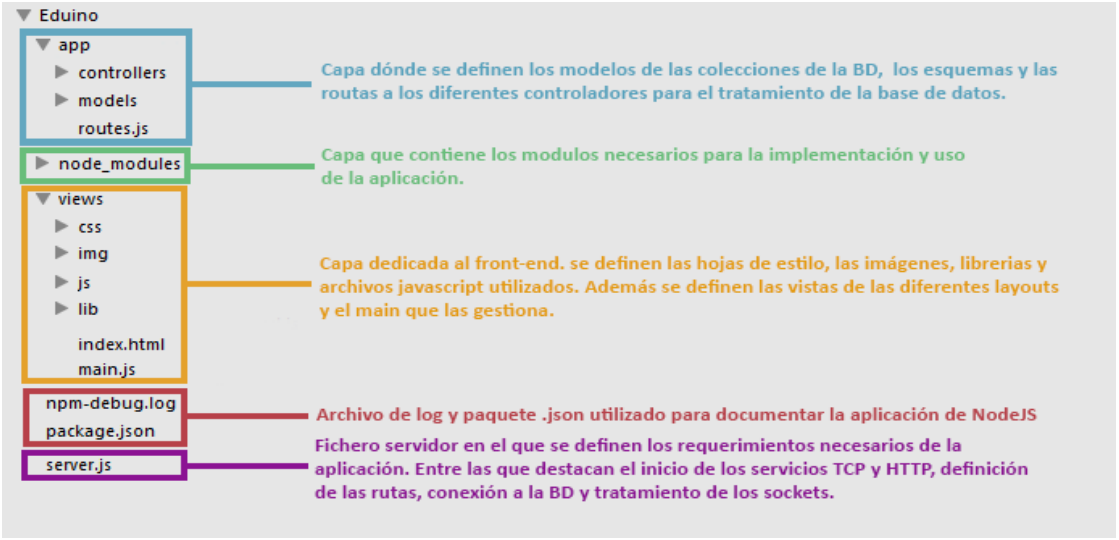


Figura 27 - Árbol del proyecto

Cómo se pudo ver en el apartado **2.6 - Instalación y preparación entorno trabajo**, el servidor requiere ciertos requerimientos para su funcionamiento.

Por lo tanto, se ha de tener en cuenta la incorporación al proyecto de los módulos que se requieren. Para su instalación se utiliza npm que viene incorporado con la instalación de NodeJS, cuya ejecución requiere se realice en el directorio de la aplicación. Dicho comando sigue el esquema:

```
npm install <nombre_del_paquete>
```

Realizadas las instalaciones de los diferentes módulos se procede a implementar en el código el requerimiento de estos. Por ello es necesario realizar el **require** de la siguiente manera:

```
var express = require('express'),
    io = require('socket.io'),
    mongoose = require('mongoose'),
    …
    bodyParser = require('body-parser');
```

Una vez se disponen de los módulos se debe proceder a especificar la conexión con la base de datos.

```
mongoose.connect('mongodb://localhost:27017/instituto');
```

Visto esto, se realiza la definición del puerto en el que se inicia la aplicación HTTP.

```
app.set('port', process.env.PORT || 8090);
```

La ruta de las vistas al front-end se especifica con la siguiente línea.

```
app.use(express.static(__dirname + '/views'));
```

También se especifica la carga de la ubicación de las rutas a los controladores:

```
require('./app/routes.js')(app);
```

2.8.2 – Implementación BD

Tomando como base la definición de la base de datos del servidor se comienza con la planificación que se desarrollará en dos fases: la generación de los documentos en la BD y la definición de la forma de estos documentos en el servidor mediante el uso de mongoose;

En la primera fase, se definen las colecciones ejemplo que servirán para la realización de las pruebas iniciales con la base de datos. Ciertos índices pueden estar obligados a contener valores únicos para ello se define la condición **unique** en el momento de su

creación. De este modo, en el caso de que se trate de insertar valores repetidos en algún campo con un índice único, MongoDB nos devolverá un error.

```
db.alumnos.ensureIndex( { "dni" : 1 }, {"unique":true} )
```

Visto esto, para insertar una nueva colección con atributos y referencias a otras colecciones se utiliza la sintaxis siguiente:

```
db.coleccion_a_insertar.insert({
    atributoN : “xxxx”,
    refColeccion: [{ _idColeccion : “xxxxxxxxxxx”}]
});
```

En la segunda fase, partiendo de las definiciones anteriores se procede a especificar los esquemas de las diferentes colecciones. Se ha de tener en cuenta que se utiliza una base de datos no relacional, por lo tanto no existe la utilización de joins; así que en casos donde se deban referenciar documentos de otras colecciones se hará uso del método **populate** de MongoDB, el cual ofrece un comportamiento parecido al típico relacional SQL.

Para la preparación de los modelos, en primer lugar se añaden las dependencias a mongoose:

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
```

A continuación, las posibles referencias a los modelos de las diferentes colecciones en las que se pueda hacer referencia en el Schema:

```
var Materia = mongoose.model('Materia');
```

Se ha de distinguir en la definición de los schema las posibles referencias a otras colecciones en base a la codificación de la base de datos MongoDB:

```
tests:[{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Test'
  }]
```

Finalmente definido el schema se realiza la exportación del modelo para el uso en la aplicación:

```
module.exports = mongoose.model('Alumno', alumnoSchema);
```

2.9 – Prototipo

Durante el desarrollo se ha detectado que el conjunto de módulos y funciones propuestos sobrepasaban las capacidades del microcontrolador Arduino UNO. Esto se ha detectado debido a mensajes de alerta por parte del IDE de Arduino basados en que la ocupación de la memoria del dispositivo excedida los niveles óptimos pudiendo producir errores de memoria:

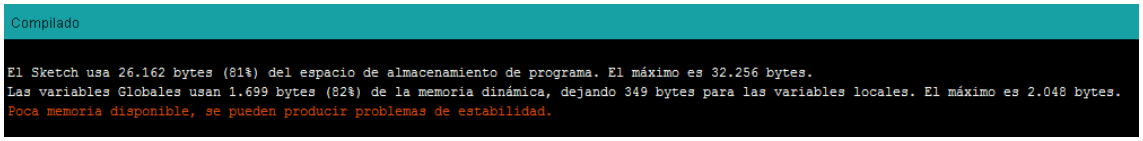


Figura 28 - Error de estabilidad

Para solventar esta situación se decide utilizar un microcontrolador que ofrezca unas características superiores, por lo tanto se decide continuar el proyecto con el uso de **Arduino MEGA**. A continuación se puede observar la comparativa de los aspectos más diferenciadores entre estos controladores:

Modelo	Arduino UNO	Arduino MEGA
Microcontrolador	ATmega328	ATmega2560
Pines E/S digitales	14	54
Pines E/S digitales PWM de salida	6	15
Entradas analógicas	6	16
Memoria Flash	32KB – 0.5KB para el bootloader	256KB – 8KB para el bootloader
SRAM	2KB	8KB
EEPROM	1KB	4KB

2.9.1 – Comunicación Arduino – Servidor

Para el éxito de la comunicación del microcontrolador y el servidor se tienen en cuenta los siguientes aspectos:

Servidor: La implementación de un servidor TCP que interactúe con Arduino.
Arduino: La definición del puerto en el que se trabajará con TCP mediante el cliente definido por el módulo EthernetShield.

Teniendo estos puntos en cuenta se procede a explicar la implementación del servidor TCP. En primer lugar, se define el inicio del servidor TCP que permite la identificación de conexiones de clientes mediante la función:

```
tcpServer.on('connection',function(socket){
  .....
 });
```

Además, en esta función se realizan las comunicaciones de recepción y envío con Arduino. Para el envío de mensajes se utiliza la instrucción:

```
socket.write('Arduino - Conectado con socket.\r\n');
```

Y para la recepción de eventos se utiliza la función:

```
//Si se recibe un mensaje de Arduino se procede al tratamiento
socket.on('data',function(data){
  console.log('NodeJS - Mensaje recibido de Arduino en server.js: ' + data + '.');
}
```

Creada la estructura, el paso final se centra en la escucha del puerto por parte del servidor.

```
tcpServer.listen(1337, function(){
  console.log('Socket funcionando en el puerto: 1337.');
});
```

Hecho todo esto, el servidor está capacitado para **recibir** y **enviar** eventos a Arduino. Acto seguido, se proceden a comentar las acciones que se deben tener en cuenta en Arduino para el envío y recepción de mensajes con el servidor.

Se empieza pues utilizando el método connect para establecer la conexión con el servidor especificado en pasos anteriores mediante el puerto 1337(TCP).

```
client.connect(server, 1337)
```

La realización de envíos de eventos al servidor se efectúa mediante la siguiente línea de código, en la que se define el contenido que recibirá el servidor.

```
client.print("Historia");
```

La recepción se realiza mediante la escucha por parte del client de posibles envíos del servidor. En el caso de que el cliente se encuentre preparado, se lee el nombre de bytes disponibles con el objetivo de tratar la información que se recibe del servidor, la cual se almacena en una variable para su control posterior.

```
if(client.available()){
      ….
      char c = client.read();
      ….
}
```

La demostración del funcionamiento de este módulo se puede consultar en formato visual en el **Anexo_EthernetShield**.

2.9.2 – Configuración del reconocimiento de voz

Para abordar este apartado se divide el trabajo en dos partes: la primera se centra en la elaboración de la tabla de sonidos que se importará a la placa; la segunda consiste en la declaración de los grupos e instrucciones de los comandos de voz que utilizará el módulo.

Se empieza pues, a preparar la tabla de sonidos mediante la conversión previa de los archivos de audio utilizando la herramienta Audacity, los cuales requieren el formato WAV con especificaciones de **16-bit - 22050Hz - mono**.

Una vez realizada la conversión, se procede a añadir estos ficheros a la tabla de sonidos para su posterior creación utilizando QuickSynthesis, la cual se realiza mediante los botones **Compres** y **Build**.

Con la tabla de sonidos creada, el siguiente paso consiste en la transferencia de dicha tabla al módulo. Para realizar este paso se debe cambiar el Jumper a UP, y de este modo proceder a utilizar el programa Easy Commander que permite importar la tabla de sonidos. Con esto se consigue disponer de las notificaciones de audio definidas en EasyVR3.

A continuación, se debe establecer otro cambio del Jumper a **PC** y así poder entablar la conexión con el puerto COM dónde se encuentra definido el microcontrolador. En los apartados de grupo se declaran los diferentes comandos que el módulo de voz va a implementar, para esto es necesario un entrenamiento del comando de voz mediante el uso del micrófono. Tras finalizar el esquema de comandos, se debe configurar vocalmente las diferentes instrucciones con motivo de que el reconocimiento sea específico y adaptado para cada usuario.

Hecho todo esto, el módulo EasyVR se encuentra cargado con la tabla de sonidos y definidos los comandos de voz que debe implementar, tan solo resta la generación del código. Para solventar este paso, la herramienta EasyCommander permite la generación de un código fuente preparado mediante las configuraciones previamente establecidas, es decir, incorporando los grupos de comandos y la tabla de sonidos que se realizará mediante el uso de la opción **generate code**.

A continuación, se pasa a trabajar con este código. De igual modo que en los pasos anteriores se debe modificar el Jumper a SW para el funcionamiento del conjunto EasyVR y Arduino.

Vistas las configuraciones iniciales, en el bucle **loop()** se interactúa mediante la repetición de la petición de introducción de un comando:

```
easyvr.recognizeCommand(group);
```

De este modo, el reconocimiento de un comando por parte del micrófono pasa por dos fases: Llamada de alerta a la escucha y escucha del comando.

Para verificar que se recibe un comando el módulo permanece en un estado continuo de escucha que active la introducción de la acción a ejecutar. En este caso se define, Robot o Eduino cómo los comandos principales. Al recibir un primer comando de voz con esta información el sistema responde con un aviso de que está pendiente de recibir una orden, y al recibir la segunda instrucción de voz, este la identificada y procede a realizar la operación pertinente.

Vista la implementación principal, se pasa al tratamiento de la reproducción de las notificaciones. Dicho aspecto, se implementa con la siguiente línea de código, dónde el primer parámetro especifica la entrada de la tabla de sonidos y el segundo la configuración de volumen deseada para el altavoz.

```
easyvr.playSound(0, EasyVR::VOL_FULL);
```

Finalmente se almacena una función que hace uso de una condición que diferencia que tipo de comando y grupo se ha detectado, y así reproducir el sonido de la tabla oportuno.

```
void action()
{
    switch (group)
    {
    case GROUP_0:
      switch (idx)
      {
      case G0_EDUINO:
        easyvr.playSound(4, EasyVR::VOL_FULL);
        group = GROUP_1; //Salto al grupo especificado
        break;
      }
      break;
    case GROUP_1:
      switch (idx)
      {
      case G1_HISTORIA:
      //Se especifican las acciones a realizar
      easyvr.playSound(2, EasyVR::VOL_FULL);
        group = GROUP_0;
     ……
}
```

La demostración del funcionamiento de este módulo se puede consultar en formato visual en el **Anexo_EasyVR3Shield**.

2.9.3 – Comunicación HTTP

En este apartado se explican los aspectos básicos que trata el prototipo en la comunicación, para más tarde en la **Etapa 2.11 – Implementación Interfaz Usuario** desarrollar la comunicación completa.

Así mismo, se procede a explicar la implementación del servidor HTTP. Para ello se definen en los requerimientos las siguientes líneas de código que crean el servidor y lo inician mediante el uso de sockets:

```
http = require('http').Server(app), //Creamos el Server HTTP
io = require('socket.io'),
io = io.listen(http), //Iniciamos el servidor con sockets.
```

Incorporada la función de sockets al servidor se definen las instrucciones que manejan los eventos que se realizan mediante la sentencia:

```
io.sockets.emit('mensaje', line);
```

Para la recepción y tratamiento de eventos en el back-end se establece una escucha permanente.

```
socket.on('mensaje',function(data){
....
}
```

Creada la estructura, el paso final se centra en la escucha del puerto por parte del servidor.

```
http.listen(app.get('port'), function(){
  console.log('NodeJS funcionando en el puerto: ' + app.get('port') + '.');
});
```

La demostración del funcionamiento del prototipo inicial se puede consultar en formato visual en el **Anexo_Prototipo**.

2.10 - Diseño

Durante la fase de desarrollo gráfico se esbozan las primeras pantallas de la aplicación y cómo se organiza el contenido en estas. Para ello se maquetan en wireframes de baja fidelidad, los cuáles darán lugar más tarde a los wireframes de alta fidelidad que son los que presentará la aplicación una vez finalizada. También se presenta el logo que distinguirá a la aplicación para la identificación del alumno y la usabilidad presentada durante el desarrollo del proyecto.

2.10.1 - Wireframe baja fidelidad

La realización de los wireframes de baja fidelidad se lleva a cabo mediante el uso de la herramienta **moqups** y permite una primera idea del resultado de la aplicación. Estos wireframes se utilizan como base para crear los wireframes de alta fidelidad explicados en el apartado **2.10.2 – Wireframe alta fidelidad**.

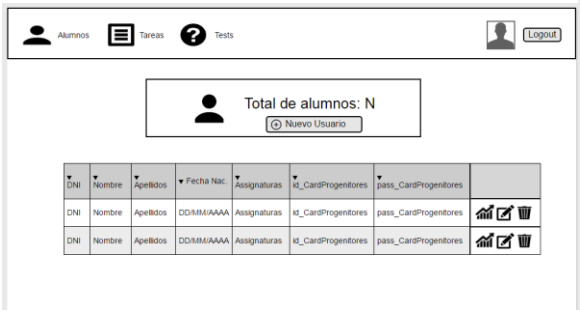


Figura 29 – Administración alumno

Figura 30 – Listado de tareas

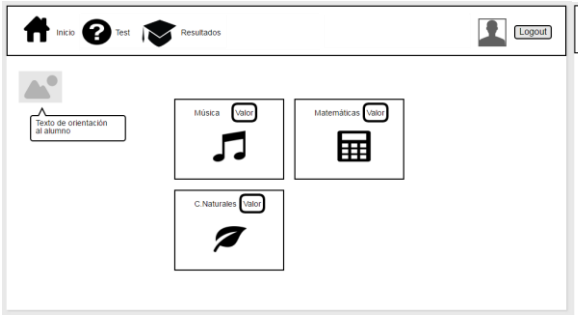


Figura 31 - Inicio

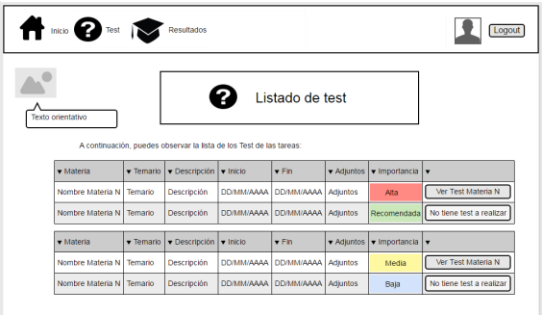


Figura 32 – Listado de test

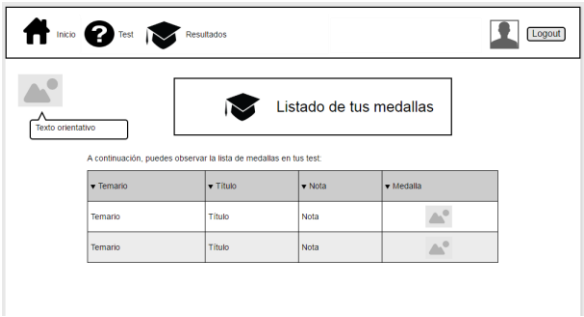


Figura 33 - Listado de medallas

Figura 34 - Listado de tareas

Figura 35 - Login - Error

Figura 36 – Login

Figura 37 - Login – Campo vacíos

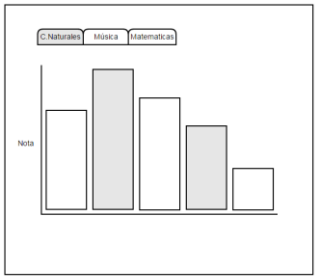


Figura 38 – Gráficas

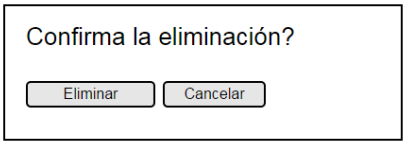


Figura 39 - Confirmación eliminar

Figura 40 - Registro Alumno – Error

Figura 41 - Registro alumno

Figura 42 - Modificación alumno: Error campos

Figura 43 - Registro alumno – Campos vacíos

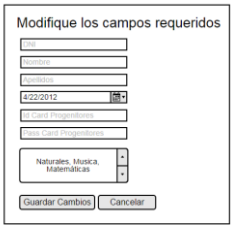


Figura 44 - Modificación tarea

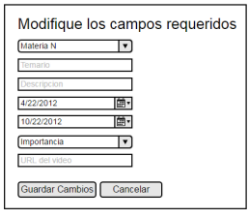


Figura 45 - Modificación alumno

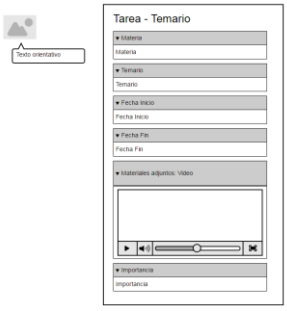


Figura 46 – Tarea

Figura 47 - Registro tarea

Figura 48 - Registro tarea – Campos vacíos

Figura 49 - Listado de test

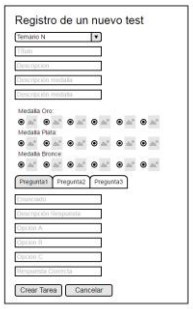


Figura 50 - Registro test

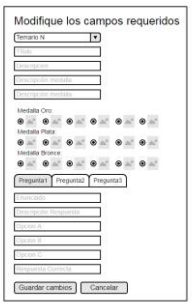


Figura 51 - Modificación test

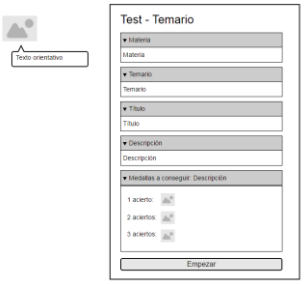


Figura 52 – Test

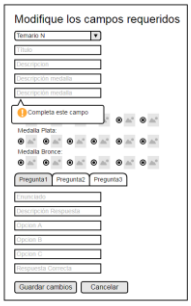


Figura 53 - Test – Campos vacíos

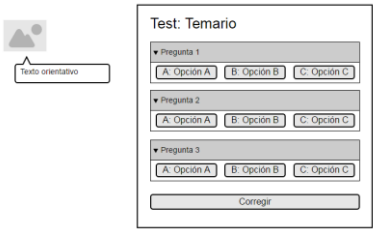


Figura 54 - Test Evaluación

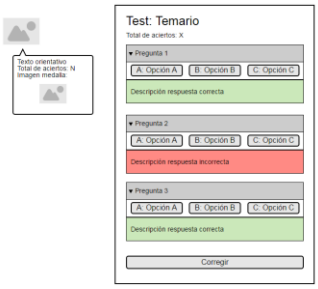


Figura 55 - Test validación

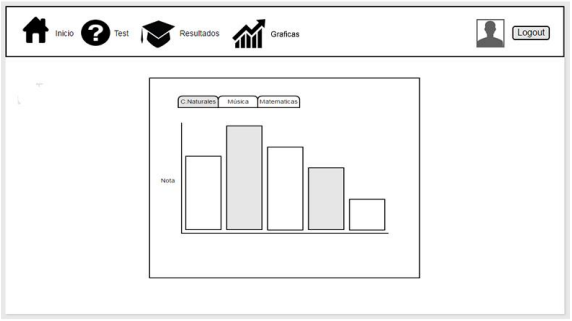


Figura 56 - Gráficas del alumno

2.10.2 - Wireframe alta fidelidad

Una vez diseñada la estructura básica de la aplicación se define la interfaz gráfica definitiva, implementando todos los estilos, tipografías, gráficos y colores a utilizar. Destacar que los wireframes de eliminar y campos vacíos son iguales en todos los casos, por lo tanto para evitar repetición se ejemplifica un único caso.

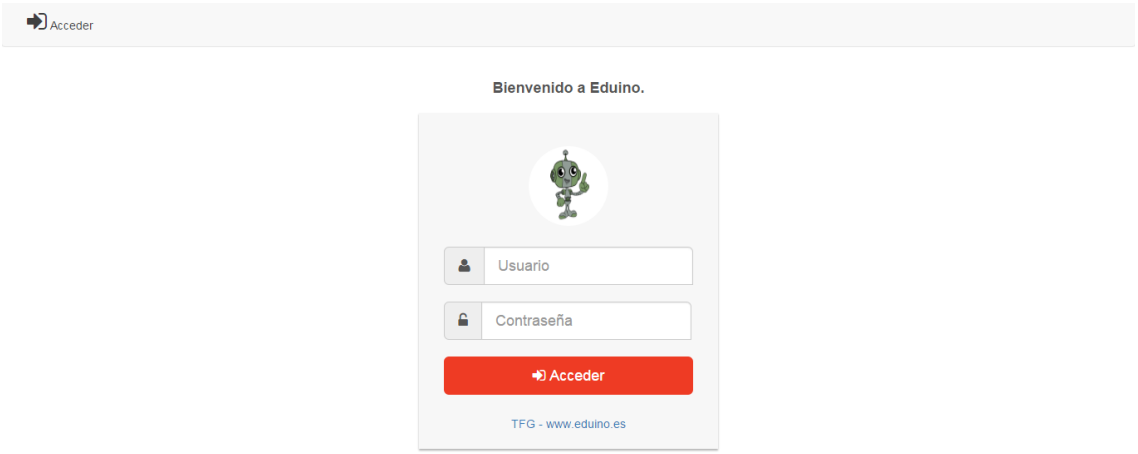


Figura 57 - Wireframe alta fidelidad: Login

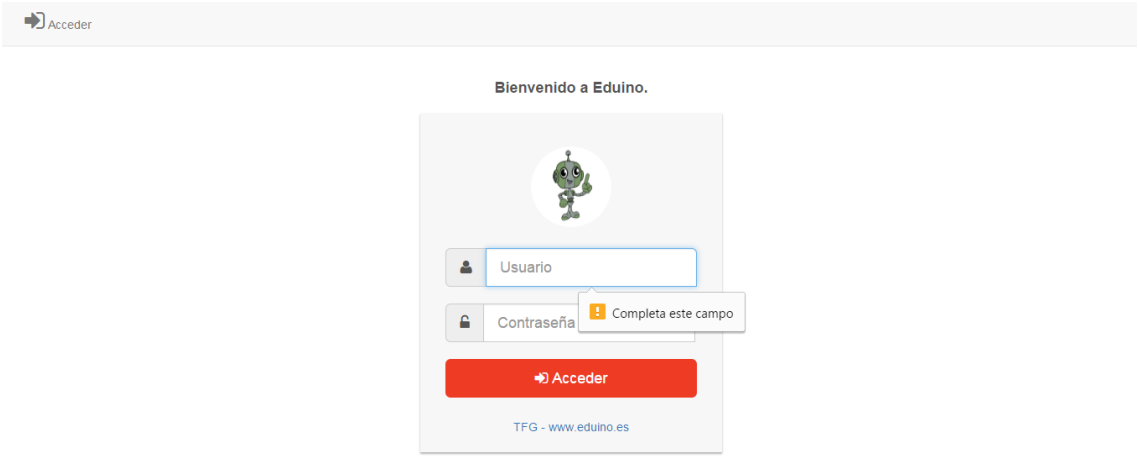


Figura 58 - Wireframe alta fidelidad: Login – Campo vacío

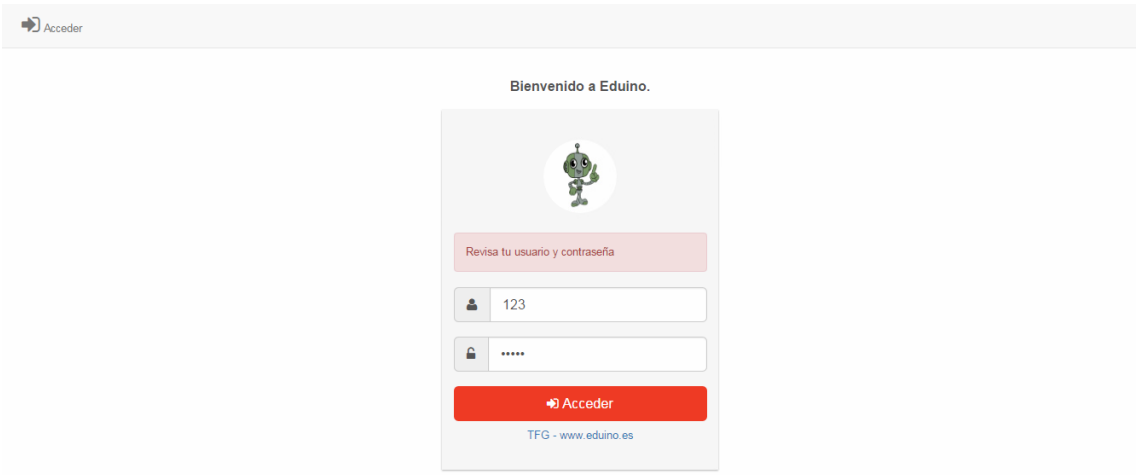


Figura 59 - Wireframe alta fidelidad: Login – Error

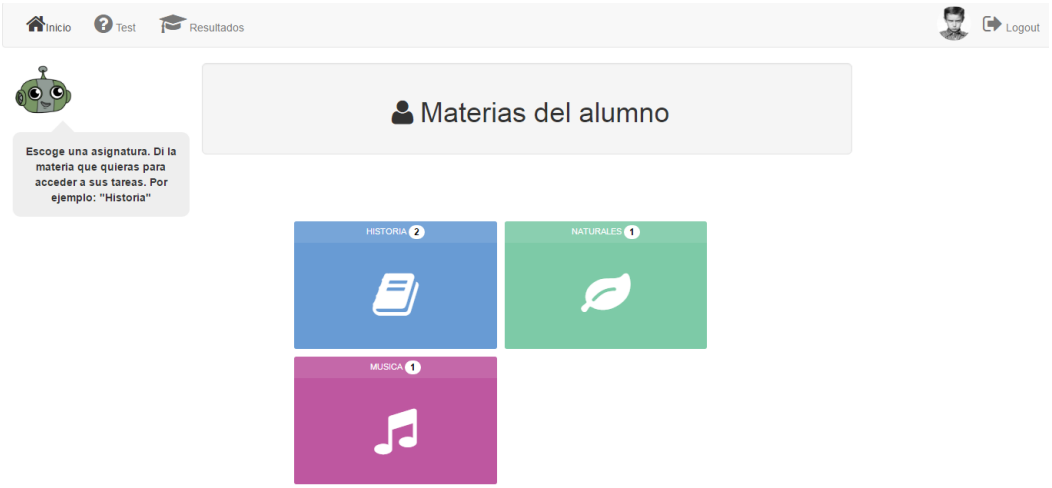


Figura 60 – Wireframe alta fidelidad: Inicio

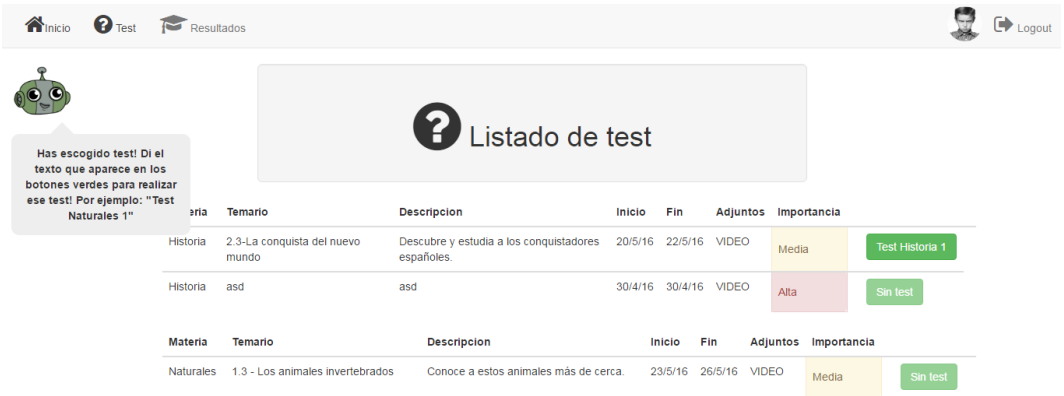


Figura 61 - Wireframe alta fidelidad: Listado de test

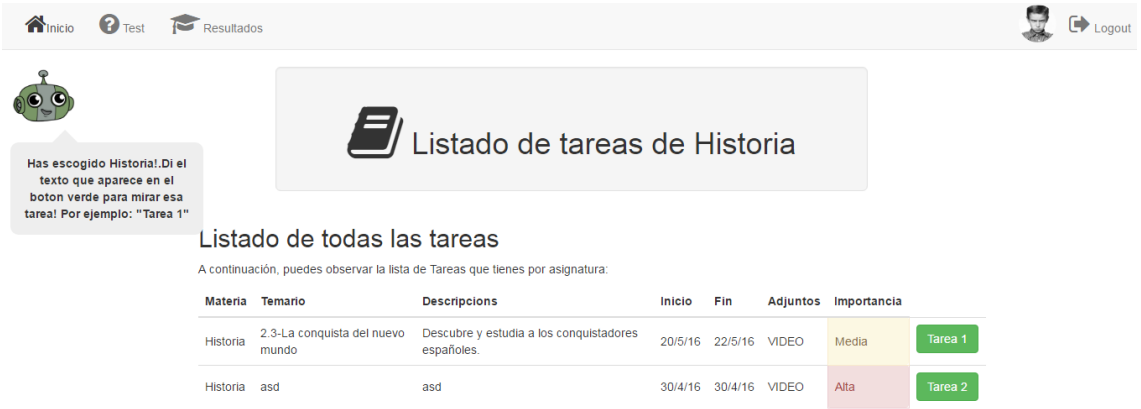


Figura 62 - Wireframe alta fidelidad: Listado de tareas

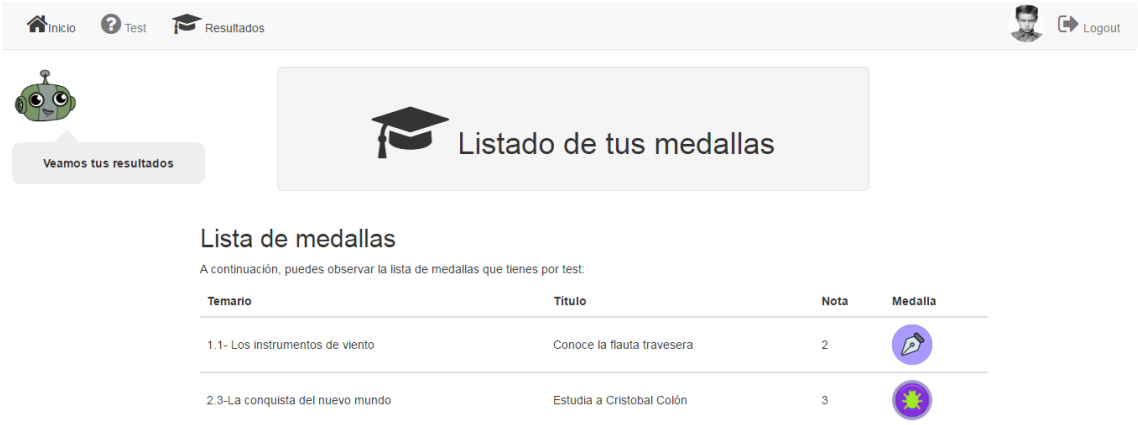


Figura 63 - Wireframe alta fidelidad: Listado de medalles

Figura 64 - Wireframe alta fidelidad: Test

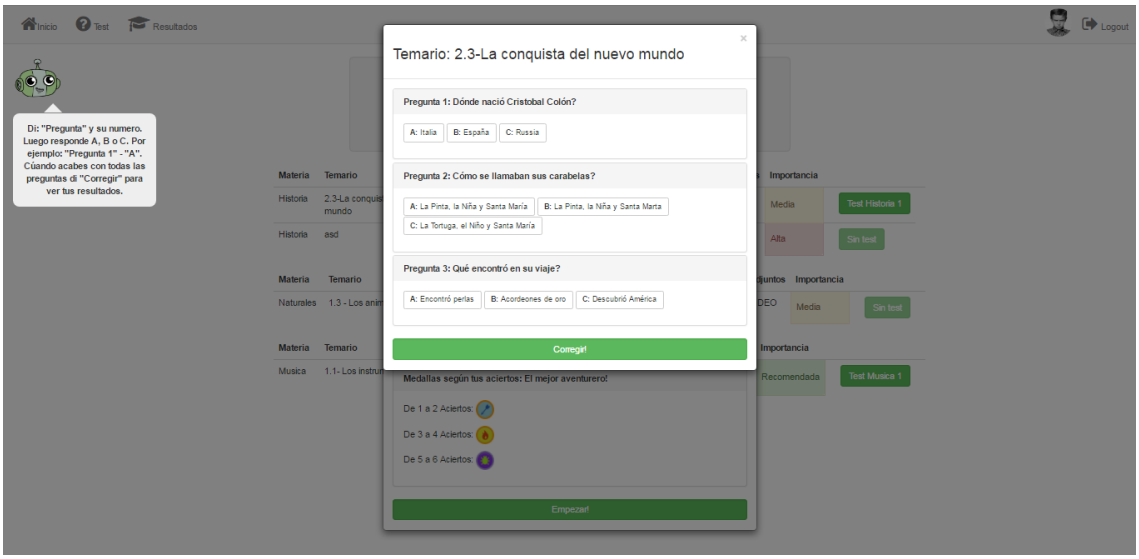


Figura 65 - Wireframe alta fidelidad: Test Evaluación

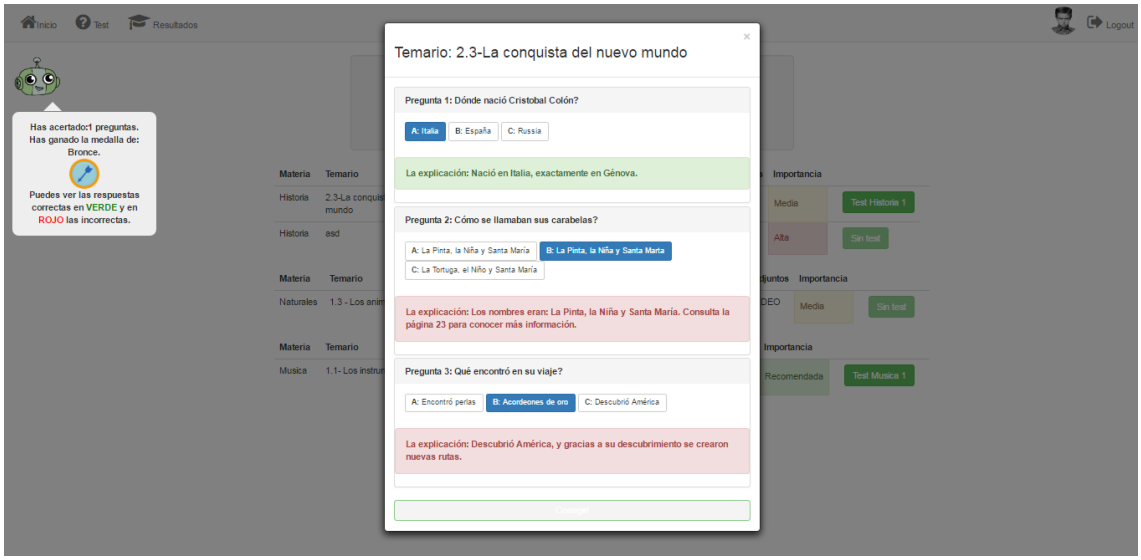


Figura 66 - Wireframe alta fidelidad: Correción test

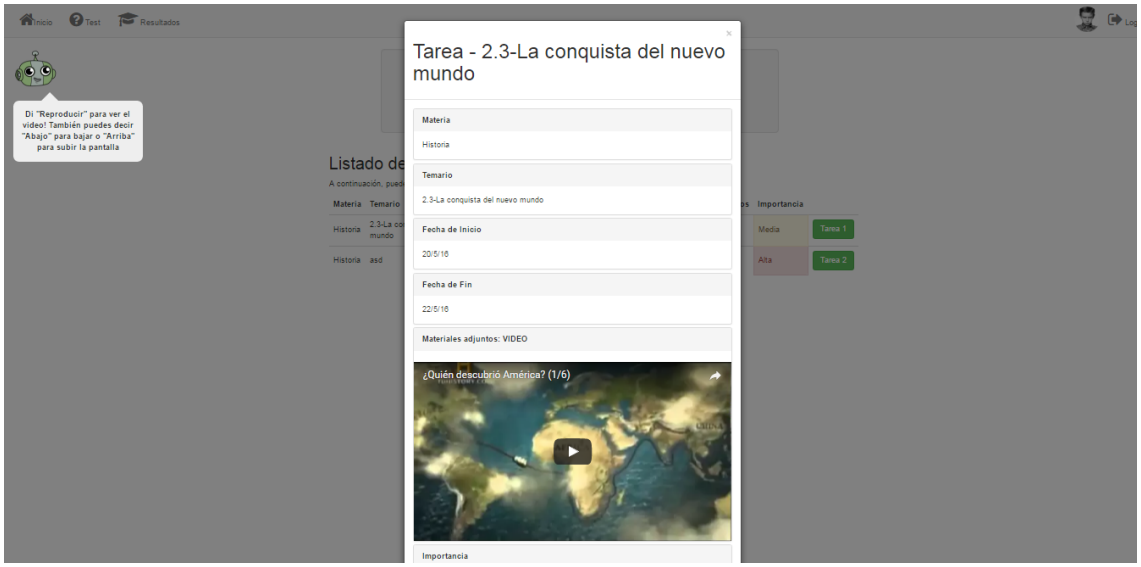


Figura 67 - Wireframe alta fidelidad: Tarea

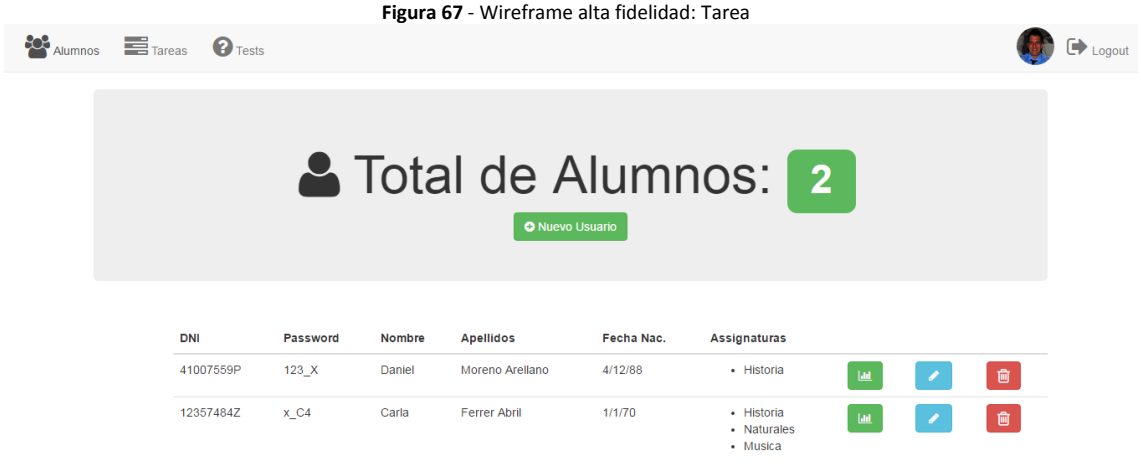


Figura 68 - Wireframe alta fidelidad: Administración alumnos

Figura 69 - Wireframe alta fidelidad: Confirmación eliminar alumno

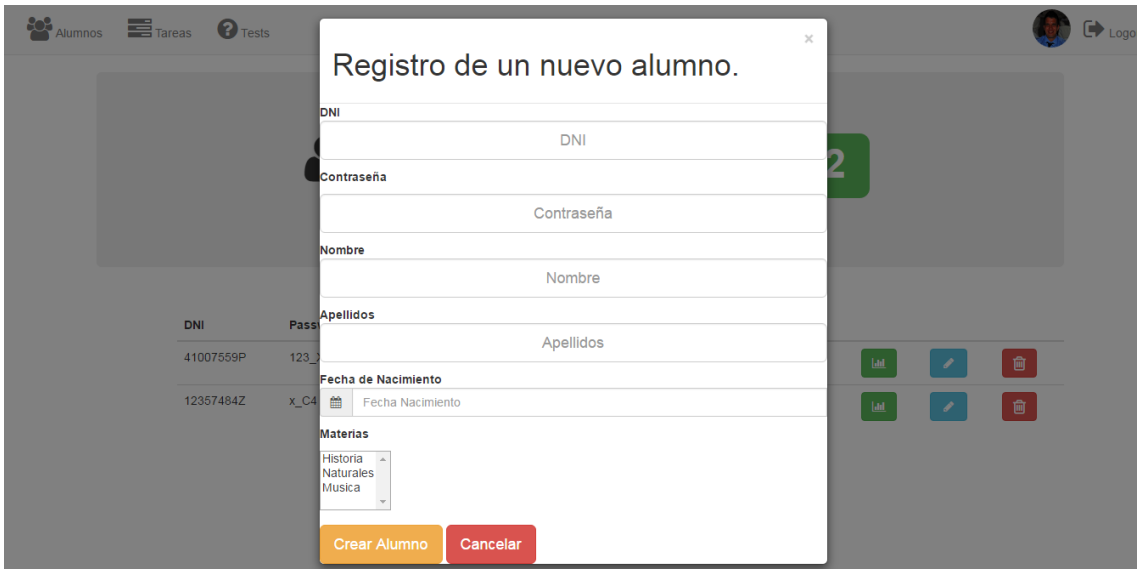


Figura 70 - Wireframe alta fidelidad: Registro alumno

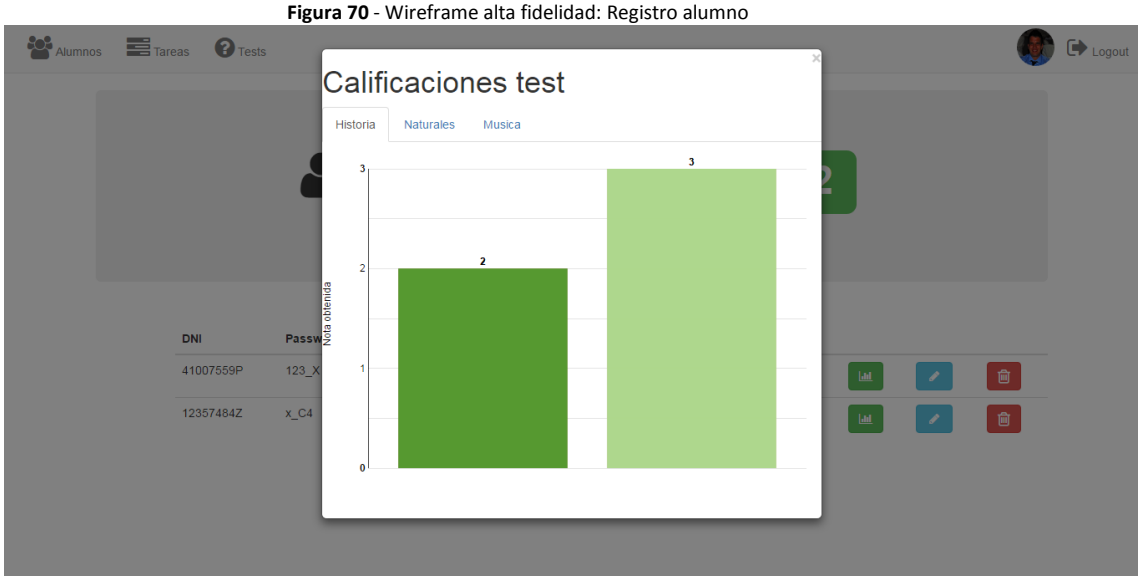


Figura 71 - Wireframe alta fidelidad: Gráficas

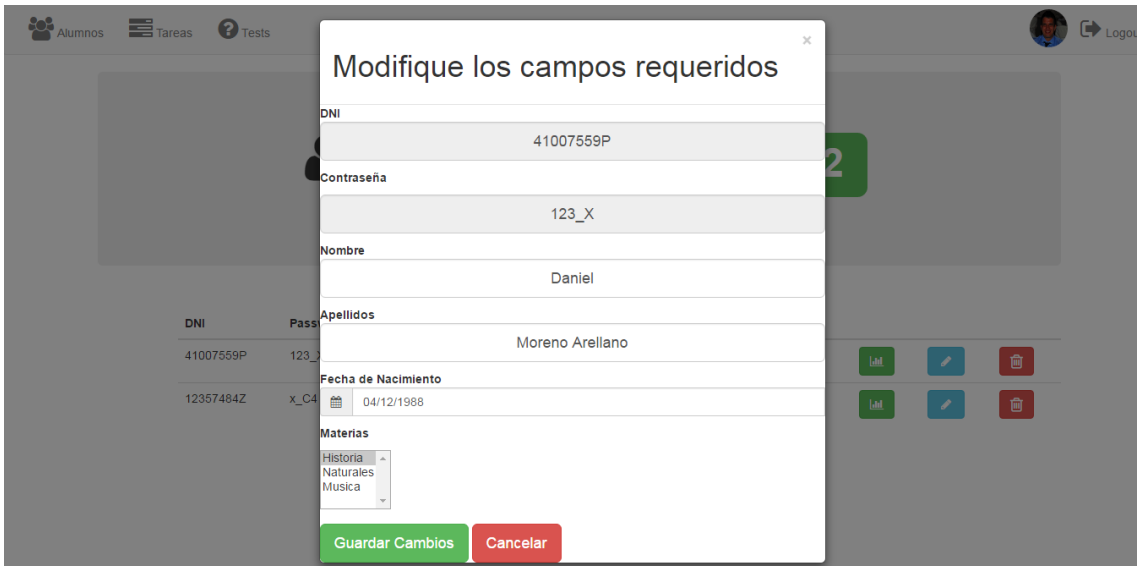


Figura 72 - Wireframe alta fidelidad: Modificación alumno

Figura 73 - Wireframe alta fidelidad: Administración tareas

Figura 74 - Wireframe alta fidelidad: Registro tarea

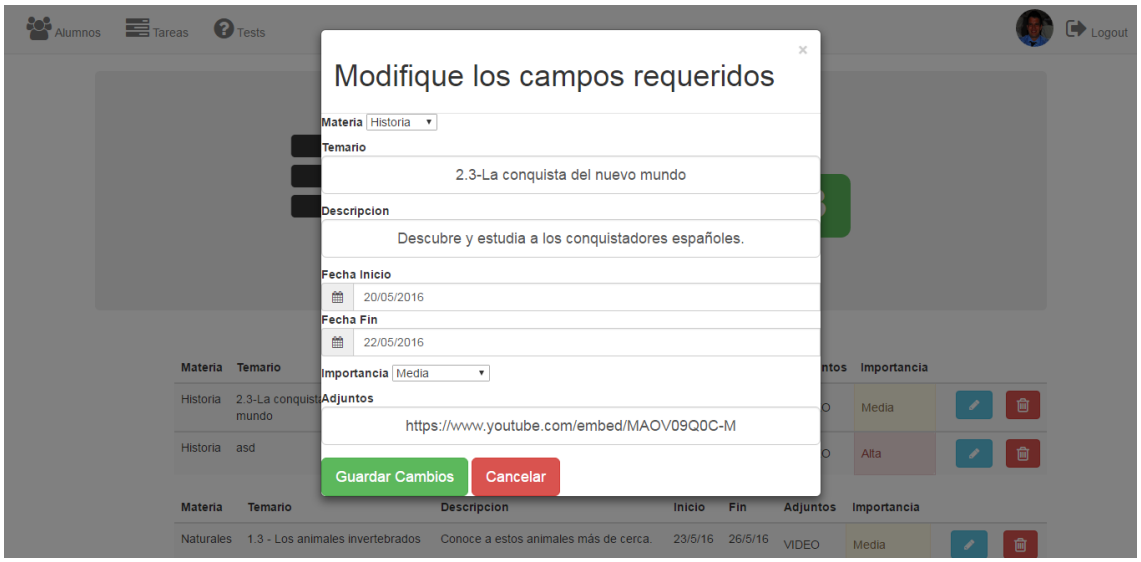


Figura 75 - Wireframe alta fidelidad: Modificación tarea

Figura 76 – Wireframe alta fidelidad: Administración test

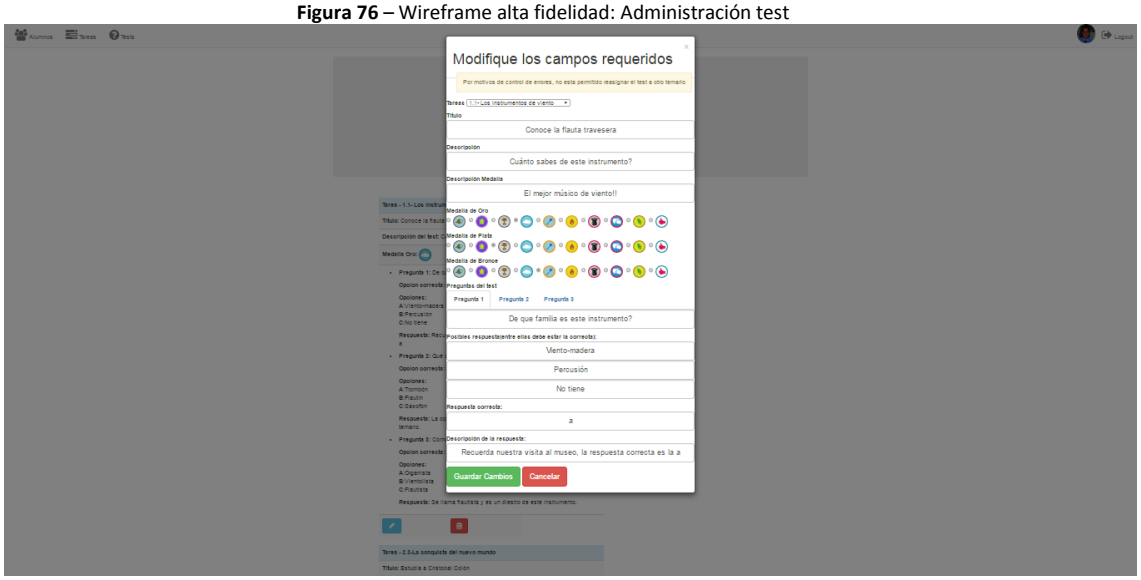


Figura 77 – Wireframe alta fidelidad: Modificación test

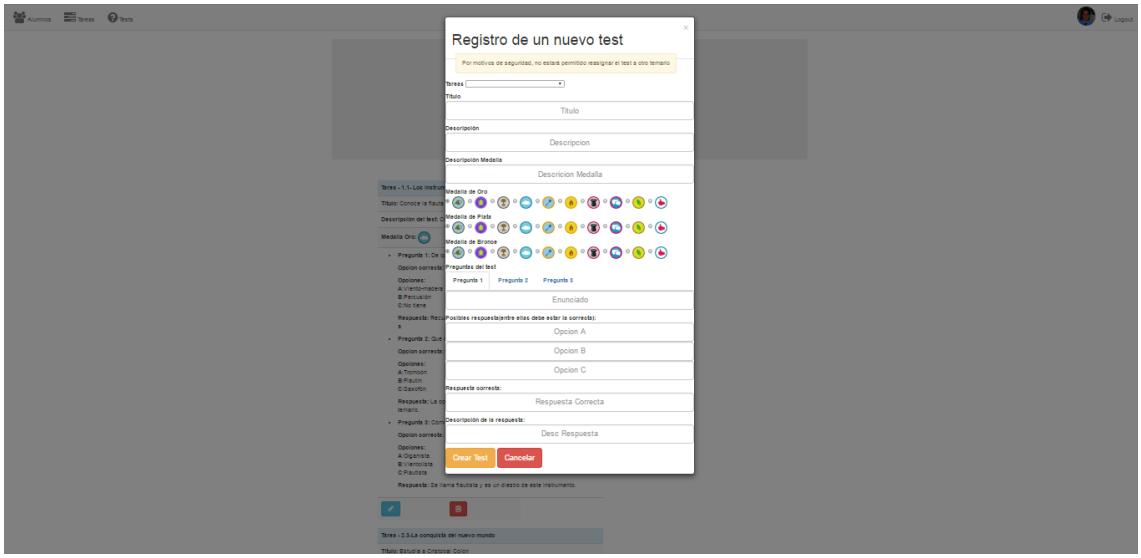


Figura 78 – Wireframe alta fidelidad: Registro del test

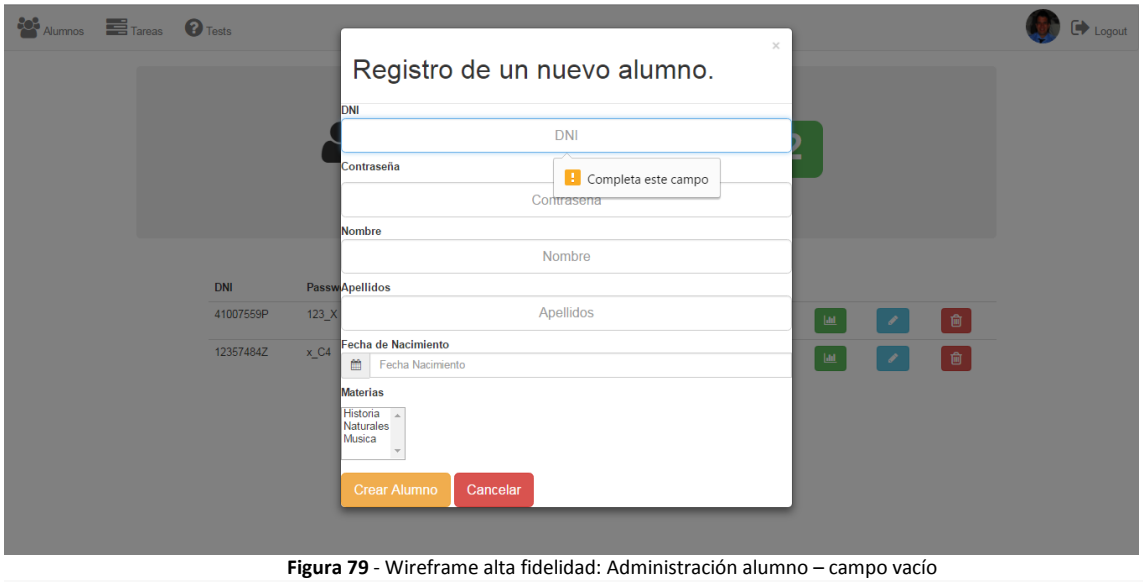


Figura 79 - Wireframe alta fidelidad: Administración alumno – campo vacío

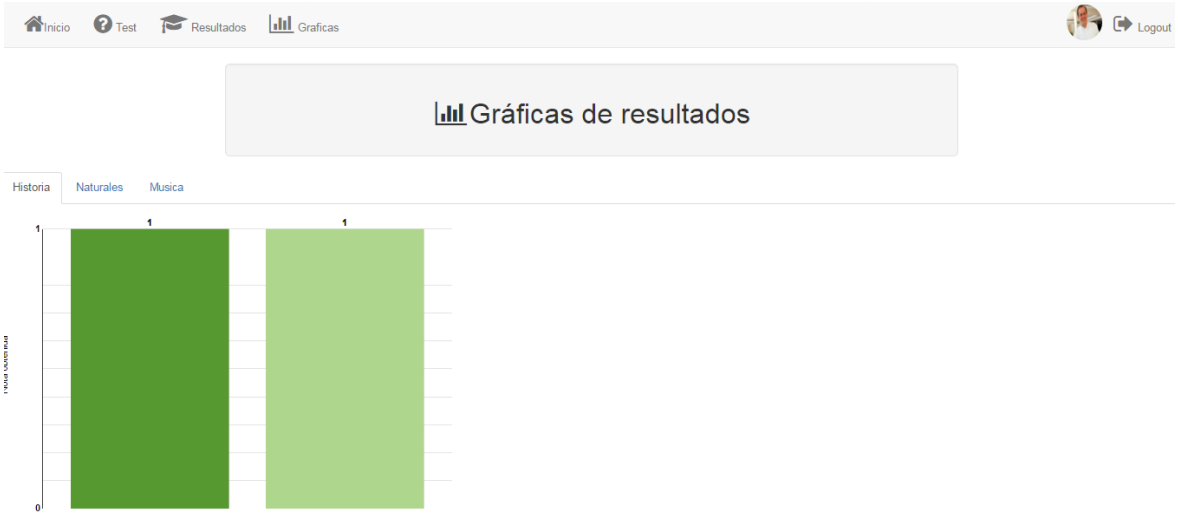


Figura 80 – Wireframe alta fidelidad: Gráficas del alumno

2.10.3 – Logotipo

Debido a que la aplicación y el uso de este proyecto está vinculado a la educación y más ciertamente a niños, se presenta un logo que identifique al robot, por eso se escoge representar un robot carismático y sonriente que resulte amigable para el alumno. A su vez se utilizará como icono para guiarlo en los pasos durante el uso de la aplicación.

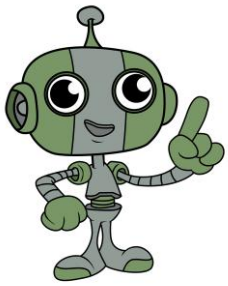


Figura 81 - Logotipo

2.10.4 - Usabilidad

En el proceso de diseño de la interfaz se ha mantenido en constante pensamiento que el uso pueda ser llevado fácilmente por un alumno que posee ciertos conocimientos de informática pero a su vez se está iniciando en el uso de las nuevas tecnologías. Por ello se destacan algunos valores que se han intentado respetar:

- La interfaz se ha diseñado con la intención de que resultes intuitiva y fácil de utilizar, minimizando la curva del aprendizaje.

- Los iconos se presentan con alta visibilidad y acompañan al texto para que resulte más sencilla su identificación.

- Se mantiene una coherencia entre las pantallas utilizando un diseño plano y colores visualmente agradables.

- Las fuentes son suficientemente grandes para aportar una fácil legibilidad.

- Se distinguen cabecera y cuerpo claramente en el diseño, utilizando la misma composición para las diferentes pantallas. De este modo, otorgando comodidad y sensación de conocer que información se encontrará debido a la coherencia de la estructura.

- Se utiliza el logotipo cómo representación de un sistema de ayuda que posiciona y transmite en todo momento la situación del alumno durante el proceso que esté realizando.

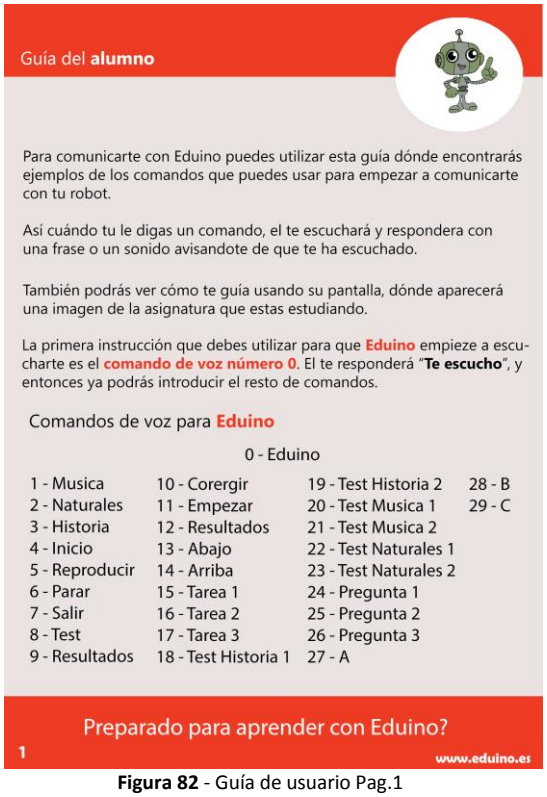
- Siempre que el alumno realiza un comando se le notifica que se ha recibido su petición y que esta se está tratando.

- Se notifica mediante comandos de voz la acción que se va a realizar facilitando la iteración y sensación de integridad con la aplicación. Al igual que las notificaciones visuales en la pantalla OLED posicionan al alumno en las diferentes materias disponibles con la intención de hacer que este encuentre todavía más interactivo el uso del robot.

- Se informa de posibles errores o fallos durante la introducción de datos o modificación en la administración.

2.10.5 – Instrucciones de uso

Debido a que el proyecto se enfoca a un ámbito y perfil que siente la necesidad de adaptarse a las nuevas tecnologías, se presenta para facilitar su comprensión, una guía de uso del robot que facilite su utilización y la identificación de los comandos de voz disponibles en cada pantalla.


Guía del **alumno**

Para comunicarte con Eduino puedes utilizar esta guía dónde encontrarás ejemplos de los comandos que puedes usar para empezar a comunicarte con tu robot.

Así cuándo tu le digas un comando, el te escuchará y respondera con una frase o un sonido avisandote de que te ha escuchado.

También podrás ver cómo te guía usando su pantalla, dónde aparecerá una imagen de la asignatura que estas estudiando.

La primera instrucción que debes utilizar para que **Eduino** empieze a escucharte es el **comando de voz número 0**. El te responderá "**Te escucho**", y entonces ya podrás introducir el resto de comandos.

Comandos de voz para **Eduino**

0 - Eduino

1 - Musica	10 - Corregir	19 - Test Historia 2	28 - B
2 - Naturales	11 - Empezar	20 - Test Musica 1	29 - C
3 - Historia	12 - Resultados	21 - Test Musica 2	
4 - Inicio	13 - Abajo	22 - Test Naturales 1	
5 - Reproducir	14 - Arriba	23 - Test Naturales 2	
6 - Parar	15 - Tarea 1	24 - Pregunta 1	
7 - Salir	16 - Tarea 2	25 - Pregunta 2	
8 - Test	17 - Tarea 3	26 - Pregunta 3	
9 - Resultados	18 - Test Historia 1	27 - A	

Preparado para aprender con Eduino?

1 www.eduino.es

Figura 82 - Guía de usuario Pag.1


Guía del **alumno**

Para poder empezar has de introducir tu nombre de usuario y tu contraseña en la pantalla de acceso:

Esta información la puedes encontrar en la tarjeta personal que incluía Eduino:

Preparado para aprender con Eduino?

2 www.eduino.es

Figura 83 – Guía de usuario Pag.2

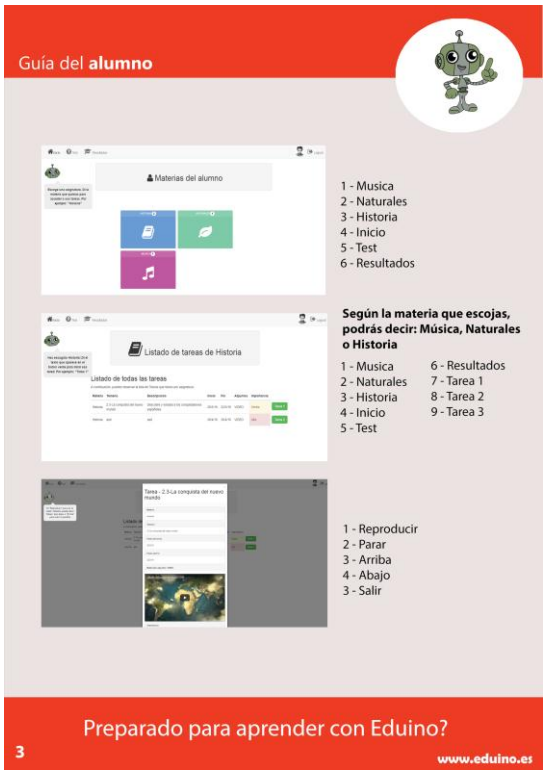


Figura 84 - Guía de usuario Pag.3

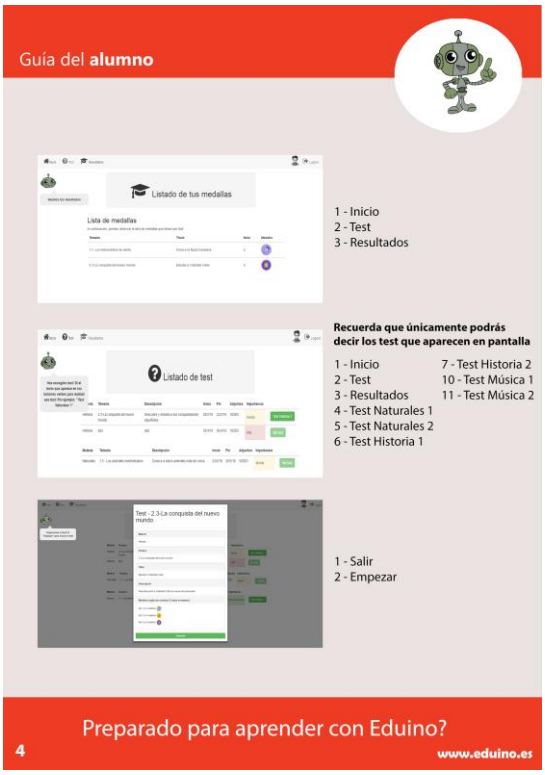


Figura 85 - Guía de usuario Pag.4

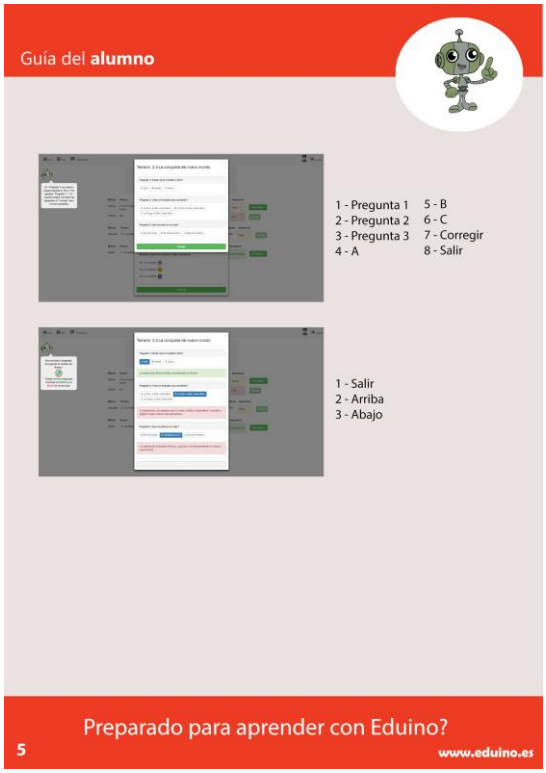


Figura 86 - Guía de usuario Pag.5

Del mismo modo, que se le entrega al alumno la guía de usuario con Eduino, se le presenta su tarjeta personal de acceso a la plataforma y la tarjeta de acceso para sus progenitores. En el primer caso, permitirá al alumno integrar conceptos implicados en

la tecnología cómo es el caso de iniciar una sesión personal, y en el segundo caso facilitará a los padres el control y seguimiento del alumno.

Figura 87 - Tarjeta acceso alumno

Figura 88 – Tarjeta acceso padres

2.11 – Implementación de la interfaz de usuario

Partiendo de los wireframes de alta fidelidad se procede a desarrollar la parte gráfica de la aplicación.

Para abordar esta fase se divide el trabajo en la parte gráfica y la parte funcional. De este modo se utilizan las herramientas AngularJS, JQuery, HTML, Bootstrap, CSS y Fontawesome para generar el contenido estático de cada pantalla. Una vez definida la

estructura visual se procederá mediante el uso de NodeJS a la generación de las funcionalidades que harán que se presente la información en la primera parte.

A partir de esta definición se crean los ficheros HTML de cada pantalla, incluyendo elementos cómo ventanas modales, selectores de fecha, botones, entre otros componentes. También se crea la barra de navegación horizontal dónde se definen las opciones del menú.

Con la estructura creada se procede a la programación de todos los métodos de la API que permiten el tratamiento de la información. De este modo, el código estático generado en la primera fase pasa a ser un código dinámico que se crea en base a los contenidos que mediante NodeJS se interaccionan con la base de datos, y a su vez se mandan para su presentación en el modelo con el uso del **scope** de AngularJS.

Finalizada la primera fase, se puede proceder a la identificación de los comandos enviados por el alumno mediante el uso de Arduino y sus componentes. Mediante el uso de sockets se proceden a realizar las configuraciones de reconocimiento en el servidor que procederán a hacer la llamada a los métodos de la API y que finalmente responderán a las instrucciones de voz.

La implementación de la interfaz sostiene los aspectos anteriormente mencionados en el apartado **2.10.4 – Usabilidad**.

2.11.1 – Desarrollo web

A continuación, se procede a explicar cómo se distribuye el aplicativo web y que pretende representar cada sección. Cabe recordar que debido a que el proyecto implementa un sistema CRUD, se permite la lectura, creación, modificación o eliminación en la administración de las tareas, test y alumnos.

Menú: Se especifican las categorías disponibles en la plataforma. En el caso del alumno, se presentan elementos estáticos cómo: Inicio, Test, Resultados. En cambio, otros elementos cómo las materias son variables en función de las materias que dicho alumno curse.

Materias: La representación de las materias trata de dividir en secciones las tareas que estas incluyen, diferenciando estas tareas para cada materia.

Tareas: Para la realización de las tareas, se han presentado una tabla que muestra el número una lista de tareas disponibles a consultar. La consulta de la información completa de cada tarea se representa en una ventana modal que se muestra al hacer clic sobre la tarea.

Test: La ejecución mantiene dependencia con la tarea. Esto es así, debido a que se determina que un test debe estar asociado a una tarea, en caso contrario el test no estará operativo. De este modo, se mantiene una relación entre las tareas y los test, consiguiendo que el alumno pueda probar los conocimientos adquiridos previamente

con la consulta de la tarea. Del mismo modo que las tareas, su contenido se muestra en una ventana modal.

Resultados: Se muestran las medallas conseguidas por el alumno. Organizadas en una tabla, su nota y el temario en la que ha conseguido el logro.

Para la **sección de administración** se especifican los siguientes apartados:

Administración tareas: La administración de tareas permite asociar una tarea a una materia determinada, asignarle un elemento adjunto (en este caso un video que refuerce la tarea) y elementos cómo título, descripción, fecha de inicio y final.

Administración test: Un test únicamente puede pertenecer a una tarea, debido a que no se ha contemplado otra posibilidad. Los test permiten definir aspectos que generan la motivación cómo son el uso de medallas y la elección de distintos diseños para estas, según los puntos conseguidos en el test. También se definen las preguntas y respuestas posibles, además de un feedback de la solución correcta para dicha pregunta.

Administración alumnos: En este apartado se permite añadir alumnos con sus campos característicos, y la asignación de 1-N materias. Además de observar un gráfico por materia de los resultados del alumno identificados por test.

Para la **sección de consulta** del alumno, se especifican los siguientes apartados:

Consultar gráficas: Para un correcto control del proceso de evolución del alumno se proporciona el acceso a las gráficas con los resultados de los test realizados.

Consultar tareas: Permite visualizar las tareas que tiene asignadas el alumno en las diferentes materias, además de la importancia de realizar la tarea, fecha de inicio y de finalización de esta.

Consultar test: La consulta de los test permite observar que tareas de la plataforma tienen asociadas un test para su evaluación.

Consultar resultados: Se muestra un listado de las medallas conseguidas y las puntuaciones en los diferentes temarios relacionados con las tareas que ha realizado el alumno en la aplicación.

2.11.2 – Conectividad Cliente – Servidor

En este punto, se procede a completar el apartado **2.9.3 – Comunicación HTTP**.

En primer lugar, se deben tener en cuenta las conexiones de nuevos clientes en el servidor. Para ello se utiliza la siguiente función, que de soporte a nuevas solicitudes de comunicación por el puerto HTTP:

```
io.on('connection', function(client){
});
```

Añadir que también se trata la desconexión del cliente con la función:

```
//Tratamiento de la desconexión de un cliente.
client.on('disconnect', function(){
  .....
});
```

Mediante el control de estas conexiones se puede informar, si es necesario, al resto de clientes conectados al servidor de que un nuevo cliente se ha conectado.

```
// Avisamos a los clientes web, que un nuevo cliente web se ha conectado.
 client.broadcast.send({ message: client.id + ' conectado.' });
```

Durante el proceso de comunicación entre el cliente y el servidor se pueden presentar estados dónde el servidor deba enviar información al cliente o viceversa. Para ello se utilizan dos funciones que permiten esta comunicación bidireccional. En el caso de enviar notificaciones a Eduino se utiliza:

```
// Envia un mensaje a Eduino
    listArduinos[a].write(message);
    console log('NodeJS - Mensaje enviado a Eduino: ' + message + '.');
```

Y como se vio en el apartado de comunicación HTTP para recibir mensajes:

```
socket.on('data',function(data){

   //Avisamos de que hemos recibido un mensaje de Eduino
    console.log('NodeJS - Mensaje de Eduino recibido en server.js: ' + data + '.');
};
```

De esta forma, en este punto del proyecto ya se puede establecer una comunicación completa entre Arduino – Cliente – Servidor mediante el uso de sockets y utilizar los beneficios de esta tecnología permitiendo una comunicación en tiempo real con el robot y la plataforma. Para visualizar un ejemplo en funcionamiento se puede consultar el **Anexo 1: Enlaces web**.

2.11.3 – API's utilizadas

Durante el desarrollo del proceso se han utilizado un conjunto de API's que interactúan con la aplicación para la obtención de los datos, modificaciones o inserción de estas en la BD. Al mismo tiempo se han definido rutinas que han permitido realizar acciones internas para la simplicidad de la programación. El conjunto de métodos utilizados se describe a continuación:

Método	Endpoint	Descripción
GET	/api/materias/:id/:idEduino	Devuelve las tareas de una materia.
GET	/api/materiasAlumno/:idEduino	Devuelve las materias de un alumno
GET	/api/gamificacionAlumno/:idEduino	Devuelve los resultados de los test de un alumno.
GET	/api/gestionAlumnos	Devuelve todos los alumnos registrados
GET	/api/getMaterias	Devuelve todas las materias disponibles.
GET	/api/gestionTareas	Devuelve todas las tareas disponibles
GET	/api/getTareasMateria	Devuelve todas las tareas disponibles de cada materia.
GET	/api/getTest/:id	Devuelve la información específica de un test en particular.
GET	/api/getTest	Devuelve la lista de todos los test disponibles.
GET	/api/getRecursos	Devuelve la lista de todas las medallas disponibles en los test.
GET	/api/graficas/:id	Devuelve las gráficas de un alumno específico.
POST	/api/testNota/:id	Registra una nueva nota de test a un alumno determinado.
POST	/api/alumno	Registra un nuevo alumno en el sistema.
POST	/api/saveTarea	Registra una nueva tarea en el sistema.
POST	/api/saveTest	Registra un nuevo test en el sistema
PUT	/api/modAlumno/:id	Trata la modificación de los datos de un alumno.
PUT	/api/updateTarea/:id	Modifica la información de una tarea.
PUT	/api/updateTest/:id	Actualiza la información de un test.
DELETE	/api/alumno/:id	Elimina un alumno del sistema.
DELETE	/api/deleteTarea/:id/:idMateria	Elimina una tarea de una materia específica.
DELETE	/api/deleteTest/:id/:idTarea	Elimina un test de una tarea específica.

2.12 – Juego de pruebas

2.12.1 – Evaluación de requisitos

Su objetivo es comprobar que la plataforma que se desarrolla cumpla todos los requisitos especificados en el proyecto y a su vez que estos funcionan de la forma esperada. Para evaluar estos se procede a realizar el siguiente juego de pruebas:

Comunicación con Arduino mediante el uso de sockets en tiempo real: Para realizar estas evaluaciones se procede a verificar que los comandos de voz se están recibiendo correctamente en el servidor mediante un sistema de log que muestra que cuando se introduce un comando de voz a Arduino. Con esta evaluación, podemos verificar que se realiza la comunicación TCP y HTTP, ya que obtenemos el registro de la recepción TCP en el log y la actualización de la pantalla en base al comando en el navegador.

Correcto registro de los documentos en la BD: Se procede a realizar una inserción, modificación y eliminación de una tarea del sistema y comprobar si se representa estos cambios en la base de datos.

Seguridad del sistema en el acceso de usuarios: Se realizan intentos de introducir una ruta en el navegador para saltarse el login. También, se procede a introducir un usuario que no existe.

Configuración de privilegios y roles: Se inicia una sesión con cada rol y se procede a verificar si cada perfil de usuario dispone de los correctos roles y privilegios descritos en el apartado **2.9.4 – Seguridad**.

Funcionalidades CRUD: Se inicia sesión con un perfil de usuario administrador y se realizan todos los casos de uso disponibles para el profesor.

Estadísticas gráficas por materia: Con el uso del rol de progenitor se accede a los resultados del alumno para verificar que los resultados de gamificación coinciden con los que se representan en las gráficas por materia. Del mismo modo, utilizando el perfil administrador se consultan las gráficas en la lista de alumnos.

Evaluación y registro de un test con Eduino: Utilizando a Eduino se recrea la realización de un test y posterior registro en la BD de la nota y medalla.

Configuración y reconocimiento de los comandos de voz por parte del servidor: Se realiza un test de los grupos de comandos utilizando la herramienta EasyVR Commander, en el que se procede a introducir diferentes comandos de la lista.

Correcto funcionamiento de sonidos y avisos del sistema: En primer lugar, utilizando la aplicación EasyVR Commander se verifica la tabla de sonidos del módulo. Una vez verificada, se procede a introducir comandos de voz a Eduino para obtener la respuesta del sonido en base al comando por el altavoz.

Finalmente, los análisis llevados a cabo determinan que estas funcionalidades se cumplen satisfactoriamente.

2.12.2 – Evaluación de usuarios

Para este apartado, se ha seleccionado un grupo de personas que cumplan los requisitos expuestos en la sección **2. 2 – Seguridad** con el objetivo de realizar pruebas sobre el sistema y el robot para analizar su experiencia de usuario y poder comprobar la eficacia del diseño y de la usabilidad. A su vez, estas evaluaciones proporcionarán la información necesaria para determinar aspectos que puedan generar situaciones de dificultad en el uso del sistema.

En el caso de los perfiles de usuario que hacen uso del navegador para consultar o administrar el sistema no se han identificado problemas para el acceso a la información o la administración de esta. Sin embargo, en la evaluación con el perfil de alumno se ha detectado que los comandos iniciales propuesto eran demasiado largos para la pronunciación y generaban confusión debido a tratarse de instrucciones con palabras quizá para un target de edades superiores. Al mismo tiempo, se detecta que en algunos casos el alumno no se ubica correctamente en relación a los comandos o pantallas que se le presentan. La propuesta de solución de estas dificultades, se detalla en el apartado **2.12.4 – Corrección de errores.**

Cómo añadido, se genera una guía de usuario específica para el alumno que detalla que comandos pueden utilizar en cada wireframe, y la lista completa de comandos para **Eduino**. Se puede consultar esta guía en el apartado **2.10.5 – Instrucciones de uso.**

2.12.3 – Identificación de errores

Se procede a identificar los errores encontrados durante el desarrollo del proyecto, a su vez se enumeran las distintas incidencias y se especifica el error detectado. La solución propuesta para cada una de las incidencias se encuentra en la fase **2.12.4 – Corrección de errores.**

Definición del tamaño de la pantalla OLED: En el momento de mostrar por la pantalla OLED la información de la materia y su nombre se detectó que estos datos aparecían ilegibles en la pantalla.

Conectividad entre Ethernet Shield, EasyVR y Arduino MEGA: Durante la fase de implementación de los componentes bajo el uso de Arduino MEGA se detectó la incompatibilidad de ambos en trabajar con la configuración implementada para **Arduino UNO**, en la que se definieron los pines 8 y 9 para la comunicación entre módulos.

Limitaciones en EasyVR: Una vez se inició la fase de juegos de pruebas y se procedió a detallar todos los comandos de voz, se observó que la placa de reconocimiento de voz no admitía toda la lista de comandos que se habían propuesto para el prototipo.

Detección de comandos con EasyVR: Esta incidencia, no se trata de un error por sí sólo, pero se considera importante tenerla en cuenta. Se observó que el robot no reconocía con facilidad los comandos de voz. Durante la evaluación de usuarios se detectó la dificultad de estos en trabajar con los comandos propuestos para la comunicación con el robot.

Dificultad por parte del alumno en la ubicación de la plataforma: Las fases de evaluación hicieron detectar la desubicación que presentaba el alumno en el uso de la aplicación.

2.12.4 – Corrección de errores

En esta fase se procede a explicar las acciones correctivas llevadas a cabo en base a los errores o dificultades detectadas durante las evaluaciones, así de esta forma conseguir una versión completamente operativa del sistema.

Definición del tamaño de la pantalla OLED: Para dar solución a esta situación se ha de modificar una línea en la librería Adafruit_SSD1306.h. Para obtener una correcta definición del tamaño de la pantalla utilizada en el proyecto la librería debe quedar de la siguiente manera:

```
   #define SSD1306_128_64
 //#define SSD1306_128_32
 //#define SSD1306_96_16
```

Conectividad entre Ethernet Shield, EasyVR y Arduino MEGA: Para solucionar este error se debió soldar en la placa de EasyVR la comunicación TX/RX en los canales 12 y 13, estableciendo así el funcionamiento predeterminado. Este error es debido a que **Arduino MEGA** utiliza determinados pines para la comunicación **RX**. Estos pines son: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15 (69). Por lo tanto, la configuración para **Arduino Uno** en EasyVR de utilizar los pines 8 y 9 no es compatible con la nueva placa, ya que el pin RX 9 no está soportado en esta para ser RX. Al mismo tiempo, se detecta que Ethernet Shield utiliza los pines 50, 51 y 52 para su funcionamiento con MEGA. De este modo, estableciendo la comunicación para RX en el pin 13 para EasyVR se consigue un funcionamiento global correcto y se solventan los conflictos entre los diferentes módulos.

Limitaciones en EasyVR: Debido a que la placa no soportaba más de 31 comandos de voz se debió resumir las instrucciones que el robot podía soportar, haciendo que el test pasara de 6 preguntas a 3 preguntas, reduciendo las tareas a un máximo de 3 por asignatura y los test a un total de 2 disponibles por materia.

Detección de comandos con EasyVR: En la fase de configuraciones del reconocimiento de voz, es importante recordar que los comandos se graban en la placa con la entonación y pronunciación de ese momento. Por lo tanto, si no se utiliza el mismo uso de la voz o la prueba la realiza otra persona que no haya hecho previamente las configuraciones de voz el sistema no responderá. Con estos requisitos se redujo la complejidad de las instrucciones de voz que el alumno debía utilizar, consiguiendo que él porcentaje de aciertos aumentara considerablemente.

Dificultad por parte del alumno en la ubicación de la plataforma: Para dar solución a esta situación se generaron nuevas respuestas por parte del robot con frases que situasen mejor al alumno y también se implementó el logotipo de Eduino con un bocadillo que exprese con claridad las opciones de las que dispone, consiguiendo así una mayor familiaridad con el sistema. A su vez, se diseñó una guía de uso dónde encontrar todas las posibles opciones en cada pantalla del sistema.

2.13 Versiones de la plataforma

2.13.1 – Versión Alpha

Esta versión corresponde a la etapa inicial de conexión del microcontrolador Arduino con el Servidor. Se detallan:

- **Reestructuración del código Arduino**: Configuración ordenada del código en clases diferenciadas para su fácil comprensión.

- **Primera infraestructura de pruebas del servidor:** Preparación del entorno de trabajo, instalación de NodeJS con sus respectivos módulos, codificación de la configuración inicial para el tratamiento de las peticiones, tratamiento de sockets e implementación de la base de datos mongoDB.

- **Primera versión de la base de datos:** codificación de las colecciones y documentos ejemplo para las pruebas de conexión y consulta de datos.

- **Prototipo de comunicación:** Página simple para la recepción de los comandos recibidos de Arduino y métodos de solicitudes a la base de datos mediante el formateo de las respuestas JSON.

- **Primera configuración de comandos y sonidos:** preparación de comandos iniciales y sonidos para el control de las instrucciones vocales.

- **Primeras notificaciones al display:** codificación de los primeros mensajes al display para la orientación en el uso del prototipo.

2.13.2 – Versión Beta

Esta versión corresponde a la primera entrega del prototipo una vez implementada la comunicación entre Arduino - Cliente - Servidor. Los aspectos presentados son:

- **Prototipo funcional**: Preparación del primer prototipo de sistema que reconoce los comandos del alumno e interactúa en función de estos.

- **Display OLED**: Diseño de las pantallas que se mostrarán en la pantalla OLED para los diferentes comandos o procesos del sistema.

- **Administración**: Se presenta la administración de la plataforma implementando las funcionalidades CRUD.

- **Estadísticas**: Preparación de las gráficas del alumno en función de los resultados de los test identificados por materia.

- **Diseño físico del prototipo**: Primer diseño del prototipo en cartón. Se puede consultar en el **Anexo 1: Enlaces web**.

- **Fase 2 página web**: Se amplía la información del proyecto en la página web. Se incluyen las imágenes del prototipo, vídeos de su funcionamiento y uso de la administración durante esta fase. También se incluye un vídeo promocional orientado a los alumnos sobre Eduino.

- **Instrucciones y sonidos**: Se terminan de definir los sonidos que utilizará el robot en la comunicación y se perfilan los comandos finales que usará el robot.

- **Versión de la base de datos**: Se genera la primera BD con contenido introducido desde la administración.

- **Logotipo**: Se incluye el logotipo que representará al proyecto.

2.13.2 – Versión 1.0 – Release

- **Detección y corrección de errores:** Se añaden las correcciones de todos los errores detectados.

- **Fase definitiva de la página web:** Se amplía la información con la descripción de esta fase. Se añaden recursos en formato vídeo del funcionamiento de Eduino.

- **Definición e implementación de los roles**: Se integran los roles y las restricciones en base a los privilegios de cada usuario.

- **Seguridad**: Se restringen posibles malos usos de la plataforma.

- **Tarjetas de acceso**: Se presentan las tarjetas que facilitaran el usuario y contraseña que dará acceso a la plataforma.

- **Gráficas para el progenitor**: Se añade la pantalla de gráficas para el rol progenitor dónde consultar la evolución del alumno en las materias.

- **Diseño definitivo del prototipo**: Diseño del prototipo en madera. Se puede consultar en el **Anexo_Eduino**.

2.14 – Proyección a futuro

2.14.1 – Objetivos secundarios

Entre las diferentes opciones que el proyecto presenta para su uso a futuro, cabe destacar los objetivos secundarios planteados que han quedado pendientes. Cómo se comenta en estos, destacar que sería interesante añadir funcionalidades **wifi** al robot, lo que permitiría equipararlo a la tendencia actual del mercado tecnológico de desarrollar productos sin cables.

También, teniendo en cuenta el uso acentuado de aplicaciones en los terminales móviles, sería muy interesante la creación de una aplicación que permita a los padres o profesor la gestión de la plataforma de igual manera a cómo la realizan con el navegador.

2.14.2 – Ampliación de comandos de voz

Otros de los aspectos importantes que se pueden considerar, recae en las limitaciones de instrucciones que presenta el módulo de reconocimiento de voz. Estas limitaciones del módulo pueden poner en peligro la escalabilidad del proyecto. Es por ello que uno de los aspectos a mirar a futuro sería ampliar el funcionamiento del proyecto a un reconocimiento de voz utilizando un terminal móvil o tableta. No obstante, se debe recordar que el proyecto se orienta a un target específico, por lo tanto, se debería evaluar el uso de estos dispositivos en este target y detallar si es conveniente su uso o si por el contrario se debe ampliar el público al que va orientado el proyecto.

2.14.3 – Sistema de notificaciones

Un aspecto interesante a desarrollar, sería que dado que la plataforma que gestiona Eduino incorpora el uso de roles, se pudiese aprovechar esta situación para implementar un sistema de mensajes y notificaciones entre el profesor y los progenitores.

Con este sistema de mensajería, el progenitor dispondría en todo momento de un conocimiento pleno sobre actividades, comportamiento, reuniones o posibles aspectos que sean importantes a tener en cuenta sobre su hijo, simplemente accediendo a la plataforma de Eduino.

2.14.4 – Multi-idioma

Otro punto a mejorar que debería llevarse a cabo en la plataforma, especialmente importante en expandir su uso, es incorporar la posibilidad de escoger un idioma con el que presentar la información, ya que todos los textos se han codificado directamente en castellano.

2.14.5 – Copias de la base de datos

Al mismo tiempo, se podría considerar la utilización de herramientas que realizarán copias de seguridad de la información de la BD en el servidor, permitiendo así poder recuperar, en caso de pérdida, el mayor volumen posible de datos almacenados. Para esto se podrían utilizar herramientas cómo **Acronis**, pudiendo gestionar copias incrementales o copias completas.

2.14.6 – Introducir un nuevo rol

Cómo ya se vio en la administración, sería interesante añadir el rol instituto cómo un **super administrador** que se encargue de gestionar la aplicación a un nivel superior que el actor profesor. De esta forma, permitir la realización de nuevos privilegios cómo poder crear nuevas medallas o crear nuevas materias, entre otras.

2.14.7 – Mejorar la interfaz gráfica

Uno de los aspectos fundamentales para que la aplicación tenga una buena aceptación, es su interfaz gráfica. Si esta resulta cómoda y atractiva para el alumno, se garantizará su éxito. Por este motivo, uno de los puntos a revisar sería su apartado gráfico, dónde un diseñador podría aportar sus conocimientos en la mejora visual de la plataforma.

2.14.8 – Sractch

Dado que el robot persigue el introducir al alumno en las nuevas tecnologías considero que la oportunidad de valorar que el robot pueda ser programado por parte del estudiante, lo puede convertir todavía más en un perfecto compañero para su introducción. Las posibilidades de utilizar Arduino con Scratch completarían la oportunidad de introducir el pensamiento computacional en el aula favoreciendo al alumno en la posible resolución de problemas cotidianos.

3. Conclusiones

Una vez concluido el proyecto es el momento de realizar un análisis de todo el trabajo llevado a cabo durante su desarrollo.

Para empezar, este proyecto me ha resultado apasionante y enriquecedor tanto en la parte tecnológica cómo en la personal. Si bien es cierto que en cada fase se han dado problemas que han requerido mi total atención, con esfuerzo y constancia se han conseguido ir resolviendo satisfactoriamente hasta dar como resultado a Eduino.

El planteamiento inicial de utilizar las tecnologías MEAN era totalmente desconocido por mi parte, del mismo modo que lo era la programación con Arduino. Aunque ambas han requerido muchas horas de aprendizaje y dedicación he disfrutado gratamente. Construir este proyecto desde cero e ir programando la comunicación con la plataforma utilizando sockets y profundizando el uso de MEAN, ha sido una ardua tarea, sin embargo ver cómo poco a poco el robot funcionaba con los diferentes módulos y estos se comunicaban con la plataforma en tiempo real, ha resultado reconfortante y desde luego ha merecido la pena.

La fase de desarrollo del prototipo y el funcionamiento entre los diferentes módulos, me resulto más compleja de lo que pensaba, ya que al no haber tenido contacto con la programación de Arduino, tuve complicaciones en la comunicación entre estos. A su vez, aprendí a soldar y ciertos conocimientos técnicos sobre el uso de los pines en las placas. Gracias a esta situación y con todo lo aprendido, cuando tuve que cambiar de Arduino UNO a Arduino MEGA por requerimientos del proyecto, me resulto mucho más intuitivo volver a hacer funcionar los módulos entre sí con el nuevo microcontrolador.

Durante el desarrollo de la parte gráfica he podido aprender sobre las técnicas que se utilizan hoy en día para el desarrollo web y ver la evolución que ha tenido desde mis inicios en la informática. Son muchos los aspectos a mejorar, sin duda, pero creo que he conseguido generar una plataforma visualmente agradable e intuitiva.

Ciertamente, el nivel de exigencia para este proyecto ha sido alto, aunque haber seguido las pautas estipuladas en el diagrama de Gantt y los consejos por parte del consultor, me han ayudado a poder seguir de una forma un poco más cómoda la planificación. Del mismo modo, haber seguido correctamente la planificación me ha permitido dar solución a errores inesperados cómo la comunicación entre módulos y el microcontrolador, y de una forma ágil poder volver a reubicarme hacia la planificación inicial.

En este proyecto se puede observar que existen diversas líneas de futuro que Eduino podría seguir para mejorar la introducción del alumno en el aula. Aspectos cómo la programación del robot, pueden aportar nuevas líneas de aprendizaje que hasta el día de hoy no se habían contemplado, y de esta forma presentar al alumno nuevas metodologías. Otros aspectos como un sistema de notificaciones permitirían que los padres se mantuvieran constantemente informados sobre la actividad de sus hijos y

pudieran seguir más de cerca los eventos que requiriesen atención. Sin duda, son muchas las opciones y oportunidades de ampliación que Eduino presenta, motivo que hace que encuentre este proyecto tan interesante.

Personalmente, estoy muy satisfecho con el resultado obtenido, dado que considero que he podido dar respuesta a todos los objetivos que me marqué al inicio, y poder observar cómo mientras escribo estas líneas he sido capaz de conseguirlo, me llena de energía para afrontar nuevos proyectos y darme cuenta que el aprendizaje basado en robótica y la educación son mis metas de futuro. También, mencionar que esta es una primera versión del proyecto que podría crecer ampliamente, y siendo sincero me gustaría ver evolucionar todavía más a Eduino y buscar una salida comercial en un futuro.

En conclusión, me llena de orgullo haber sido capaz de afrontar las situaciones críticas y de tensión mostrándome resolutivo, y ver que con esfuerzo y dedicación uno puede conseguir cualquier reto que se proponga.

4. Glosario

Arduino: Es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios.
Android: Es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets.
Wireframe: Es una representación esquemática de una página web sin elementos gráficos que muestran contenido y comportamiento de las páginas.
TIC: Tecnologías de la información y la comunicación.
MEAN: Acrónimo de *MongoDB - Express - AngularJS - Node.JS*.
CRUD: Acrónimo de Crear, Leer, Actualizar y Borrar, en inglés (Create, Read, Update, Delete).
Feedback: método de control de sistemas, en el cual los resultados obtenidos de una tarea o actividad son reintroducidos nuevamente en el sistema.
BD: Abreviatura de Base de datos.
Front-end: Corresponde a la parte del software que interactúa con el o los usuarios.
Back-end: Corresponde a la parte que procesa la entrada de peticiones por parte del usuario.
Bootstrap: es un framework o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web
AngularJS: Es un framework MVC de JavaScript para el desarrollo del front-end.
Sockets: concepto abstracto por el cual dos programas pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.
HTML: Siglas en inglés de *HyperText Markup Language*, hace referencia al lenguaje de marcado para la elaboración de páginas web.
JSON: Acrónimo de *JavaScript Object Notation*, es un formato de texto ligero para el intercambio de datos.
APP: *Abreviatura* de la palabra inglesa application, suele referirse a aplicaciones para teléfonos móviles inteligentes.
CSS: Siglas en inglés de *Cascading Style Sheets* es un lenguaje usado para definir y crear la presentación de un documento estructurado.
Responsive: se refiere al uso de la técnica de diseño web que busca la correcta visualización de una misma página en distintos dispositivos.
Parsear: analizar sintácticamente una secuencia de un documento o sentencia escritos en un lenguaje en particular.
Npm: gestor de instalación de paquetes javaScript para el entorno NodeJS.
Target: público objetivo al que se dirige el producto.

5. Bibliografía

En este apartado se incluyen los recursos utilizados durante el transcurso del proyecto:

Bootstrap: Diseño gráfico [5/05 – 13/05]
http://getbootstrap.com/getting-started/

Bootstrap DatePicker: Diseño para fechas [5/05 – 13/05]
https://bootstrap-datepicker.readthedocs.io/en/latest/

Bootstrap Modals: Diseño gráfico para el modal [5/05 – 13/05]
http://getbootstrap.com/javascript/#modals

Arduino: Microcontrolador [17/03 – 25/03]
https://www.arduino.cc/en/Reference/HomePage

FontAwesome: Diseño gráfico de iconos [5/05 – 13/05]
http://fontawesome.io/get-started/

W3Schools CSS3: Diseño gráfico de estilos [5/05 – 13/05]
http://www.w3schools.com/css/css3_intro.asp

W3Schools HTML5: Diseño gráfico web [5/05 – 13/05]
http://www.w3schools.com/html/html5_intro.asp

Socket.io: Uso de sockets en MEAN [4/04 – 15/04]
http://socket.io/

Express: Paquete MEAN [28/03 – 31/03]
http://expressjs.com/

AngularJS: Paquete MEAN [5/05 – 13/05]
https://angularjs.org/

AngularJS ng-grid: Tablas [5/05 – 13/05]
http://angular-ui.github.io/ui-grid/

AngularJS ng-click: Invocación al click [5/05 – 13/05]
https://docs.angularjs.org/api/ng/directive/ngClick

AngularJS ng-pattern: Control del formulario [20/05 – 23/05]
https://docs.angularjs.org/api/ng/directive/ngPattern

AngularJS ng-messages: Control del formulario [20/05 – 23/05]
https://docs.angularjs.org/api/ngMessages/directive/ngMessages

AngularJS ng-repeat: Tratamiento JSON [5/05 – 13/05]
https://docs.angularjs.org/api/ng/directive/ngRepeat

AngularJS ng-show/ng-hide: Controlador de campos [5/05 – 13/05]
https://docs.angularjs.org/api/ng/directive/ngShow

Passport: Seguridad [20/05 – 23/05]
http://passportjs.org/

NgCookies: Sesiones [20/05 – 23/05]
https://docs.angularjs.org/api/ngCookies

NodeJS: Paquete MEAN [28/03 – 31/03]
https://nodejs.org/en/

MongoDB: Paquete MEAN [1/04]
https://www.mongodb.com/es

EasyVR: Reconocimiento de voz [21/03 – 23/03]
http://www.veear.eu/products/easyvr3/

JQuery Onclick: Invocación del click [5/05 – 23/05]
https://api.jquery.com/click/

JQuery scroll: Scroll en modales [5/05 – 23/05]
https://api.jquery.com/scroll/

Powtoon: Creación del video publicidad [3/05 – 5/05]
https://www.powtoon.com/home/g/es/

Nvd3 Charts: Gráficas [5/05 – 23/05]
http://nvd3.org/

Adafruit display OLED: Pantalla [23/03 – 25/03]
https://learn.adafruit.com/category/lcds-and-displays

Ethernet Shield: Módulo para Internet [18/03]
https://www.arduino.cc/en/Reference/Ethernet

5.1. Comunidades

Durante el desarrollo del proyecto se han utilizado documentaciones y comunidades de usuarios de las diferentes tecnologías. A continuación se enumeran las más representativas:

Arduino: http://forum.arduino.cc/
Stackoverflow: http://es.stackoverflow.com/
Github : https://github.com/
Adafruit: https://forums.adafruit.com/

6. Lista de figuras

Anexo 1: Enlaces web

En este apartado se incluyen las rutas que enlazan a recursos disponibles en la web de presentación del proyecto y permiten la ampliación de la información mediante videos explicativos.

Anexo_EasyVR3Shield: http://www.eduino.es/easyVR3Shield.html
Anexo_EthernetShield: http://www.eduino.es/EthernetShield.html
Anexo_DisplayOLED: http://www.eduino.es/displayOLED.html
Anexo_ Prototipo: http://www.eduino.es/ArduinoMEGA.html
Anexo_Administracion: http://www.eduino.es/administracionTutor.html
Anexo_FaseUno: http://www.eduino.es/prototipoFaseUno.html
Anexo_Eduino: http://www.eduino.es/Eduino.html