



TRABAJO FIN DE GRADO  
INGENIERÍA INFORMÁTICA

# Adaptación de la plataforma VIRTRA-EL a nuevas tecnologías

---

**Autor**

David Infante Casas

**Directores**

Carlos Rodríguez Domínguez María  
José Rodríguez Fórtiz



Escuela Técnica Superior de Ingenierías Informática y de  
Telecomunicación

—  
Granada, Mayo de 2020









## **Adaptación de la plataforma VIRTRA-EL a nuevas tecnologías**

David Infante Casas

**Palabras clave:** aplicación web, nuevas tecnologías web

### **Resumen**

El proyecto consiste en la adaptación de los ejercicios, páginas y componentes básicos de la Plataforma Virtual de Evaluación e Intervención Cognitiva en Mayores (VIRTRA-EL), desarrollada inicialmente en PHP y JavaScript, a los frameworks Angular e Ionic.

La plataforma se desarrolló inicialmente para la evaluación y estimulación cognitiva de personas mayores, ofreciendo ejercicios a modo de juegos serios, a través de Internet. Está destinada a la prevención e intervención sobre el deterioro cognitivo, accesible a través de internet por cualquier persona en el caso de no tener recursos asistenciales o con dificultades para acceder a ellos.

El proyecto nace con el propósito de adaptar la plataforma VIRTRA-EL a tecnologías más modernas tales como Angular e Ionic. Angular ofrece la posibilidad de crear interfaces muy interactivas, además, es posible la modularización de cada ejercicio en componentes y la información requerida en servicios Angular. Por otro lado, Ionic presenta componentes estándar de interfaces de usuario, que, complementados a Angular, permiten la creación de interfaces muy intuitivas.

Con el fin de que la aplicación sea lo más extensible posible, se ha implementado un sistema de adición de nuevos ejercicios y carga dinámica. Angular nos permite cargar nuevos ejercicios dinámicamente cuando sea necesario y mediante eventos se posibilita el paso de información entre los ejercicios y la página que los carga.



## **VIRTRA-EL's adaptation to newer web technologies**

David Infante Casas

**Keywords:** Web application, new web technologies

### **Abstract**

The project consists in the adaptation of the exercises, pages and basic components from "Plataforma Virtual de Evaluación e Intervención Cognitiva en Mayores" (VIRTRA-EL) previously developed in PHP and JavaScript to Angular and Ionic frameworks.

The platform was initially developed for the cognitive evaluation and stimulation of the elderly via Internet. Its purpose is the prevention and intervention of cognitive impairment, accessible via Internet by everyone in case of not having healthcare resources or difficulty accessing them.

The project was born with the purpose of adapting VIRTRA-EL to modern technologies such as Angular and Ionic. Angular offers the possibility of creating deeply interactive interfaces. It is also possible to modularize each exercise in components and the required information in Angular services. On the other hand, Ionic presents user interface standard components that, complemented with Angular, allows the creation of really intuitive interfaces.

In order to make the application as extensible as possible, there has been implemented an exercise addition system and dynamic loading. With this system it's really simple to add new exercises; Angular allows dynamic loading of the exercises when it's needed and through events the information can transit between the exercises and the page that loads them.





Yo, **David Infante Casas**, alumno de la titulación **TITULACIÓN de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75165296P, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: David Infante Casas

Granada a 23 de Mayo de 20



**D. Carlos Rodríguez Domínguez,**  
Profesor del Área Lenguajes y Sistemas  
Informáticos del Departamento  
Lenguajes y Sistemas Informáticos de la  
Universidad de Granada.

**Dña. María José Rodríguez Fórtiz,**  
Profesora del Área de Lenguajes y  
Sistemas Informáticos del Departamento  
Lenguajes y Sistemas Informáticos de la  
Universidad de Granada.

**Informan:**

Que el presente trabajo, titulado ***Adaptación de los ejercicios de la plataforma VIRTRA-EL a nuevas tecnologías***, ha sido realizado bajo su supervisión por **David Infante Casas**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 23 de Mayo de 2020.

**Los directores:**

**Carlos Rodríguez Domínguez**

**María José Rodríguez Fórtiz**

# Índice

1. [Introducción](#)
  - 1.1. [Motivación o justificación del proyecto](#)
  - 1.2. [Objetivos](#)
  - 1.3. [Planificación temporal](#)
  - 1.4. [Presupuesto](#)
2. [Estado del arte](#)
  - 2.1. [Dominio del problema a resolver](#)
  - 2.2. [Metodologías y tecnologías base que podrían usarse](#)
  - 2.3. [Trabajos relacionados](#)
3. [Propuesta](#)
  - 3.1. [Metodología de trabajo](#)
  - 3.2. [Fase 0](#)
  - 3.3. [Fase 1](#)
    - 3.3.1. [Arquitectura y navegabilidad del sistema](#)
    - 3.3.2. [Páginas del sistema](#)
    - 3.3.3. [Componentes básicos del sistema](#)
    - 3.3.4. [Ejercicios y sesiones](#)
      - 3.3.4.1. [Página Session y componente Assistant](#)
    - 3.3.5. [Sistemas de comunicación](#)
      - 3.3.5.1. [Comunicaciones frontend-frontend](#)
      - 3.3.5.2. [Comunicaciones frontend-backend](#)
        - 3.3.5.2.1. [Lista de endpoints y modelos de datos](#)
  - 3.4. [Fase 2](#)
    - 3.4.1. [WordListExercise](#)
    - 3.4.2. [LogicalSeriesExercise](#)
    - 3.4.3. [SemanticSeriesExercise](#)
    - 3.4.4. [PositionsExercise](#)
    - 3.4.5. [DirectNumbersExercise](#)
    - 3.4.6. [NumbersAndVowelsExercise](#)
    - 3.4.7. [PyramidsExercise](#)
    - 3.4.8. [WordListExerciseAlternative](#)
    - 3.4.9. [MouseDialogExercise](#)
    - 3.4.10. [IntroductionExercise](#)
    - 3.4.11. [InstrumentalQuestionnaireExercise](#)
    - 3.4.12. [ClassifyObjectsExercise](#)
4. [Conclusiones y trabajos futuros](#)
  - 4.1. [Conclusiones](#)
  - 4.2. [Trabajos futuros](#)
5. [Bibliografía](#)
  - 5.1. [Referencias](#)
  - 5.2. [Figuras y tablas](#)
6. [Anexo](#)
  - 6.1. [Cómo ejecutar la aplicación](#)
  - 6.2. [Cómo añadir nuevos ejercicios y sesiones](#)

# 1. Introducción

## 1.1. Motivación y justificación del proyecto

El proyecto nace con el propósito de adaptar los ejercicios, páginas y componentes básicos de la plataforma VIRTRA-EL, cuyo desarrollo finalizó en 2018, a tecnologías más modernas. La versión original de VIRTRA-EL utilizaba principalmente los lenguajes PHP y JavaScript.

Recientemente han surgido nuevas tecnologías web, mucho más usables, las cuales se han convertido en estándar.

Un claro ejemplo es Angular. Lanzado en 2016, desarrollado y mantenido por Google, es un framework de código abierto utilizado en el frontend para el desarrollo de aplicaciones web [1]. Angular ofrece la posibilidad de crear interfaces muy interactivas, además, es posible la modularización de cada ejercicio en componentes y la información requerida en servicios Angular. Además Angular está soportado por Google, lo cual es una gran fuente de confianza en lo que a actualizaciones y apoyo al desarrollador se refiere, esto implica que esta tecnología va a seguir siendo apoyada por lo menos, durante un futuro próximo.

Por otro lado, Ionic es un SDK cuya primera versión fue lanzada en 2013. Implementa componentes web estándar “*cross-platform*”, utilizados para la creación de interfaces de usuario [2].

Con el fin de evitar la obsolescencia y mejorar el rendimiento de la plataforma se decide comenzar el desarrollo de la plataforma VIRTRA-EL usando estas tecnologías. Al complementar Angular e Ionic, se permite la creación de interfaces muy intuitivas.

Angular también ofrece “la máxima velocidad y rendimiento en plataformas web a día de hoy”. [1] Un ejemplo de esto es que Angular solo carga módulos cuando son necesarios, por tanto podemos asegurar que la migración a Angular de la plataforma conlleva una mejora en el rendimiento de la misma.

En cuanto a Ionic, los componentes que ofrece son estándar en lo que a dispositivos móviles se refiere, con lo cual se aumenta el rango de compatibilidad soportando dispositivos tanto Android como iOS, incluyendo sus versiones más modernas [3][4]. Por tanto, son de una gran utilidad en nuestro caso, ya que sus pantallas táctiles facilitan la interacción, siendo ideales para el público objetivo de esta aplicación, personas de edad avanzada.

Con el objetivo en mente de que el sistema sea lo más flexible y extensible posible, se desea implementar un sistema de carga de ejercicios utilizando las funcionalidades “*componentFactory*” y emisores de eventos en Angular. Gracias a este sistema, será posible, de manera muy sencilla, desarrollar nuevos ejercicios en forma de componentes Angular. Al desarrollar los ejercicios de forma independiente, se pueden añadir, eliminar o modificar ejercicios sin influir en el resto, por tanto, se simplifica el mantenimiento de la plataforma en el futuro.

## 1.2. Objetivos

El principal objetivo de este proyecto ha sido adaptar la aplicación web “VIRTRA-EL” destinada a la estimulación y evaluación cognitiva de personas de edad avanzada a tecnologías más modernas. Con esta propuesta se pretende mejorar el rendimiento y la compatibilidad de la aplicación, además de facilitar su mantenimiento y futuras extensiones mediante la reutilización de componentes.

Adaptar la aplicación a nuevas tecnologías aporta distintas ventajas que pueden fomentar su uso, por ejemplo:

- Se aumenta el rango de compatibilidad de dispositivos y navegadores incluyendo sus versiones más modernas, por tanto, es más fácil para los usuarios acceder a la plataforma desde cualquier smartphone, tablet u ordenador. [\[3\]](#)[\[4\]](#)
- Los componentes que Ionic ofrece son sencillos de entender y con los que interactuar, siendo ideales para el público objetivo de esta aplicación.

Para lograr este fin, se han desarrollado una serie de objetivos más específicos, subdivididos en dos categorías:

- Objetivos de investigación, estudio y práctica:
  - Investigar y estudiar de las tecnologías con las que se va a trabajar, Angular, Ionic, Node.js y MongoDB, para estar más familiarizado con su uso y conseguir una mayor productividad.
  - Estudiar y conocer la estructura e implementación original de VIRTRA-EL, para tener una visión general de la plataforma y poder desarrollar una nueva versión acorde a la funcionalidad de la original.
  - Practicar con esas tecnologías para reforzar los conceptos aprendidos sobre componentes, Ionic pages, servicios o TypeScript.
  - Estudiar aspectos específicos del desarrollo del frontend de una aplicación, el uso de componentes en una sola página y su sincronización.
  - Estudiar las funcionalidades “*componentFactory*” y emisores de eventos en Angular para construir un sistema flexible, extensible y para proporcionar la mayor libertad posible al programador a la hora de desarrollar nuevos ejercicios de cara al futuro.
- Objetivos de diseño, desarrollo e implementación:
  - Diseñar y desarrollar los diagramas del sistema y documentación necesaria para implementar sesiones y ejercicios.
  - Desarrollar las distintas páginas que forman la aplicación para

estructurar la navegación en el sistema.

- Desarrollar el sistema de carga de ejercicios para poder utilizar cualquier ejercicio desarrollado en un componente Angular.
- Desarrollar e implementar elementos básicos del sistema en componentes para su reutilización en la aplicación.
- Diseñar e implementar algunos de los ejercicios y sus servicios, para dejar una base de ejercicios que pueda ser ampliada en un futuro.
- Desarrollo y configuración del backend con el que el que el frontend se pueda comunicar para enviar y recibir información.



### 1.3. Planificación temporal

El proyecto se ha subdividido en tres fases de desarrollo.

En la primera, denominada “Fase 0”, se han realizado tareas de estudio y práctica con las tecnologías con las que se iba a trabajar. En la segunda, se ha planteado y desarrollado la arquitectura y navegación del sistema. Por último, en la tercera fase de desarrollo se han implementado algunos de los ejercicios de la plataforma. La misión de estos ejercicios es la estimulación cognitiva del usuario que los realiza.

La duración del proyecto ha sido de un total de tres meses.

A continuación, en la [figura 1](#), se muestra el diagrama de Gantt del proyecto.

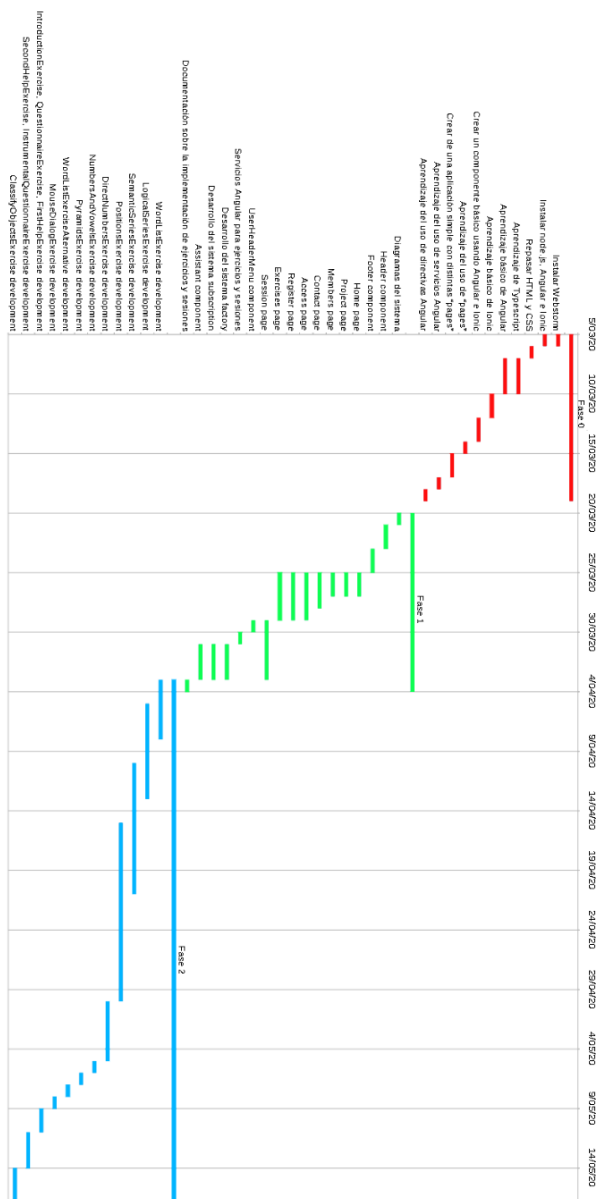


Figura 1. Diagrama de Gantt del proyecto

## 1.4. Presupuesto

El cálculo del presupuesto se ha realizado suponiendo que la persona que va a realizar el proyecto es ingeniero informático y tiene ciertos conocimientos de diseño y desarrollo de software. El coste de contratación aproximado de un informático recién licenciado se ha supuesto de unos 1250 euros al mes. La duración del proyecto se ha aproximado a unos seis meses. Por otro lado, se incluyen los costes de hardware y software necesarios para el desarrollo del proyecto.

Cabe destacar que tanto Angular [5] como Ionic [6] cuentan con licencias “MIT” [7], lo cual indica que el uso de ambos frameworks es gratuito y por tanto no se han añadido gastos adicionales de estas licencias en el presupuesto.

El presupuesto final aproximado se puede observar en la [figura 2](#).

Gastos Elegibles	Importe
<b>Gastos de Personal – Contratación de personal</b>	
Ingeniero / Desarrollador 1250€/mes * 6 meses de proyecto x 1	7.500,00 €
<b>Gastos de Ejecución – Material destinado al proyecto</b>	
Ordenador portátil x 1	1.000,00 €
Licencia WebStorm IDE 12,90€/mes x 3 meses de proyecto	38,70 €
<b>Coste total aproximado:</b>	<b>8.538,70 €</b>

Figura 2. Tabla de presupuesto

## 2. Estado del arte

### 2.1. Dominio del problema a resolver

VIRTRA-EL es un proyecto inicialmente desarrollado por los miembros del grupo MYDASS de la UGR junto a otros grupos y entidades colaboradoras con el fin de crear una plataforma virtual para la evaluación y estimulación cognitiva de personas mayores a través de Internet.

Esta plataforma se pone a disposición de todas aquellas personas que requieran prevención o intervención sobre el deterioro cognitivo y que no dispongan de recursos asistenciales o dificultades de desplazamiento para acceder a ellos. La aplicación también toma datos sobre el paciente con el fin de mejorar la plataforma.

La aplicación cuenta con una serie de páginas: Inicio, Proyecto, Participantes, Contacto, Acceder, Registro y Ejercicios (ver apartado [Páginas del sistema](#)). Una vez registrado un usuario, puede acceder a los ejercicios que el sistema aporta, los cuales están pensados para que sean realizados con la ayuda de un terapeuta o una persona capaz de asistir al usuario.

Con cada uno de estos ejercicios se realiza una evaluación y una estimulación de habilidades cognitivas como memoria, atención, planificación y razonamiento. El usuario ha de realizar distintas tareas en función del ejercicio, por ejemplo, memorizar una lista de palabras y escribir aquellas que recuerde tras un periodo concreto de tiempo (ver [figura 3](#)), ser capaz de reconocer palabras cuyo campo semántico no coincide con el resto (ver [figura 4](#)), clasificar imágenes según su tipo (ver [figura 5](#)) o distinguir imágenes con patrones concretos (ver [figura 6](#)).

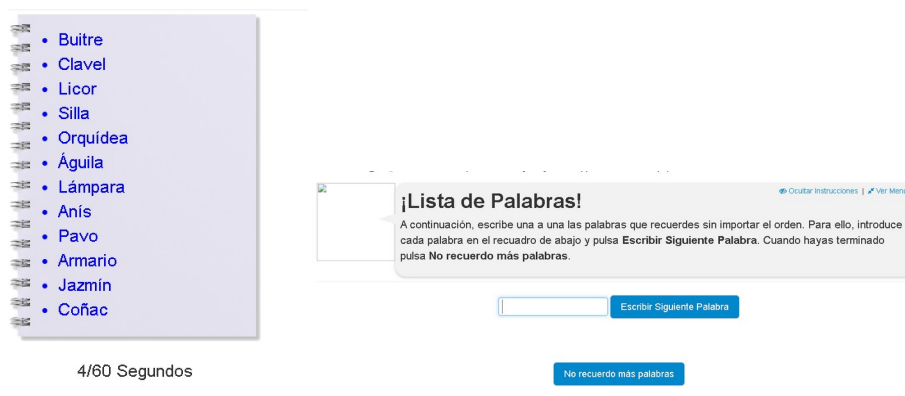


Figura 3. WordList Exercise



Figura 4. SemanticSeries Exercise

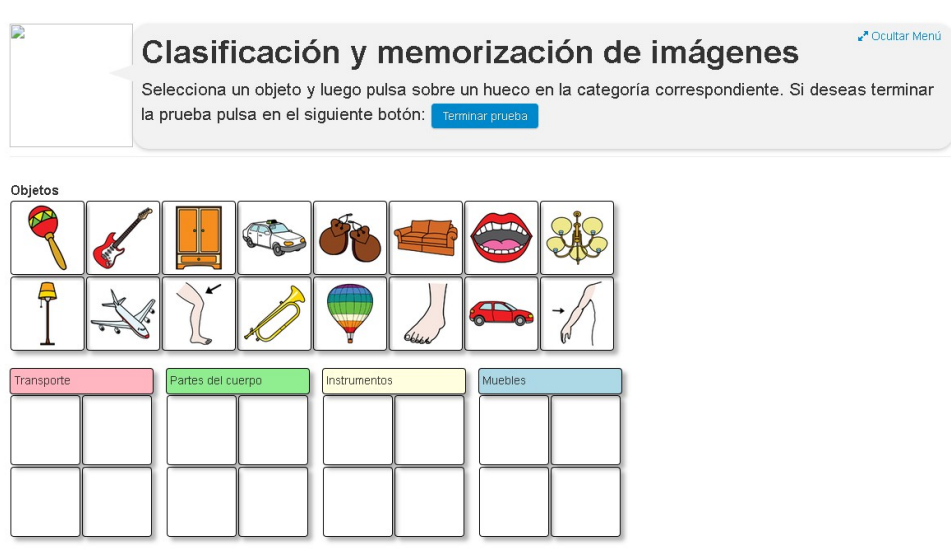


Figura 5. ClassifyObjects Exercise

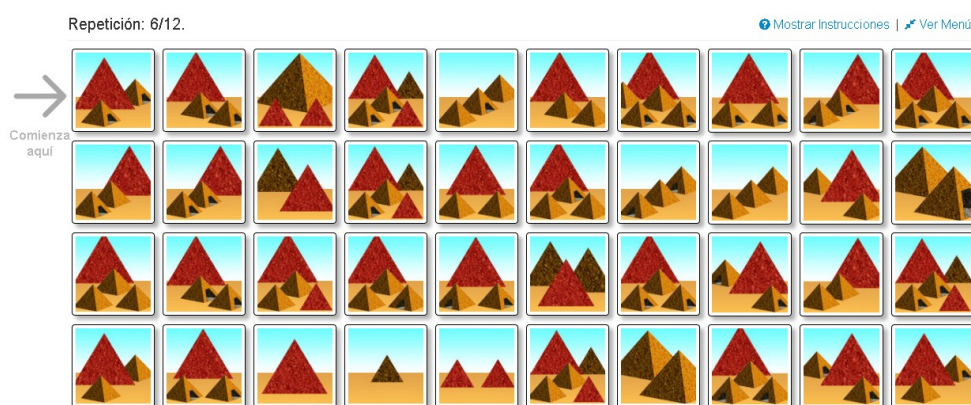


Figura 6. Pyramids Exercise

Estos ejercicios se agrupan en sesiones con una duración aproximada de una hora y una media de cinco ejercicios por sesión.

Los ejercicios están desarrollados principalmente en JavaScript con el ocasional uso de PHP para acceder a variables de sesión y para la conexión con la base de datos que recoge información de la interacción del usuario con los ejercicios y los

resultados de estos. Los datos de los ejercicios, incluyendo textos e imágenes se encuentran en ficheros XML. Permiten estructurar la aparición de los datos en el ejercicio y su relación de dependencia, por ejemplo, una serie de objetos con sus nombres y enlaces a ficheros de imagen (ver apartado [Ejercicios y Sesiones](#)). Estos ficheros XML se han sustituido por servicios Angular en la nueva versión de la plataforma.

La idea de migrar VIRTRA-EL a nuevas tecnologías propone resolver una serie de inconvenientes de la implementación original:

- Uno de los principales motivos con el que se toma la decisión de adaptar a nuevas tecnologías la plataforma es la de mejorar la compatibilidad de la aplicación. Gracias al uso de tecnologías como Angular, se garantiza la compatibilidad con navegadores modernos o con dispositivos móviles. Favorecer la compatibilidad con estos últimos es vital, dado que gran parte del público objetivo de la aplicación no dispone de ordenadores o portátiles, pero sí de smartphones o tablets.
- Un problema de la implementación original es la dificultad para extender la aplicación debido a estar desarrollada en PHP y JavaScript. Para facilitar el mantenimiento de la plataforma en el futuro, se ha optado por encapsular los ejercicios en componentes Angular, ideales para poder desarrollar nuevos ejercicios de forma sencilla. Estos componentes utilizan tecnologías como TypeScript que es un lenguaje compilado a diferencia de JavaScript, lo cual permite detectar errores en tiempo de compilación y por tanto, facilita el desarrollo. Al ser tipado y orientado a objetos, el código es más consistente y limpio. Por otro lado, el uso de componentes Angular garantiza al programador que el nuevo código que se añada al sistema no interferirá con el código base de la plataforma.
- Otro gran factor a la hora de comenzar el proyecto es buscar una mejora en el rendimiento de la aplicación. Angular solo carga componentes cuando son necesarios, incrementando la eficiencia y rendimiento de las aplicaciones desarrolladas en este framework. En las siguientes figuras (ver [figura 7](#)) se puede ver una comparativa en cuanto a tiempos de carga, scripting, renderizado y dibujado de un mismo ejercicio, en la versión original y en la nueva versión. La figura de la izquierda representa la versión antigua y la de la derecha la nueva.

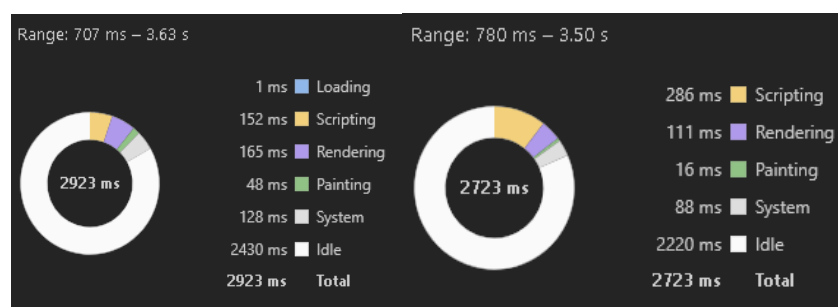


Figura 7. Rendimiento del ejercicio LogicalSeriesExercise en la versión original y en la nueva

- Utilizar Angular como framework, también soluciona otro problema presente en la implementación original, algunas llamadas al backend se encuentran en el frontend. Los servicios Angular permiten separar del frontend las llamadas al backend, dejando un código mucho más legible y estructurado.

- Adaptar el aspecto visual a nuevos estándares. La adaptación visual de la plataforma aumenta la usabilidad de la misma, es por esto que se ha optado por usar componentes Ionic con el fin de que la aplicación pueda ser usada en cualquier dispositivo, adoptando un aspecto visual acorde a este. El uso de Ionic también soluciona un problema del desarrollo original, reduce la cantidad de código CSS necesario para estilizar el código HTML.

Además de los ejercicios, la plataforma cuenta con otras funcionalidades, por ejemplo, cambiar el icono del asistente que explica los ejercicios, un sistema de asignación de terapeutas (ver [figura 8](#)) y cuidadores, ver la lista de medallas obtenidas durante la realización de ejercicios (ver [figura 9](#)) y la posibilidad de compartir los resultados obtenidos en los ejercicios a través de Facebook.

#### Aún no tiene un terapeuta asignado.

Si sabe el e-mail de su terapeuta, y éste ya se encuentra registrado en VIRTRA-EL, puede introducirlo a continuación para solicitarle ser su paciente.

Escriba el e-mail del terapeuta

Resultado de 3 + 2   
Esto evita que se haga un mal uso del formulario de solicitud.

[Solicitar](#)

De manera alternativa, puede contactar con nosotros a través del siguiente formulario para que le asignemos uno.

Texto

Resultado de 2 + 4   
Esto evita que se haga un mal uso del formulario de contacto.

[Enviar](#)

[Limpiar](#)

Figura 8. Sistema de asignación de terapeutas



Figura 9. Medallas obtenidas

Estas funcionalidades no se migrarán durante el proyecto dado que no está previsto su desarrollo, aunque su desarrollo podría realizarse en una fase futura (ver [Trabajos futuros](#)). Utilizando módulos como “*angularx-social-login*” que nos

permiten la autenticación con cuentas de Google, Facebook o Amazon, se podría replicar el sistema original de compartición de resultados en la nueva implementación.

## 2.2. Metodologías y tecnologías base que podrían usarse

Para la realización de este proyecto se han valorado distintas alternativas en cuanto a las metodologías y tecnologías que se podrían haber aplicado.

Por un lado, la metodología de desarrollo que se ha adoptado ha sido “desarrollo en cascada”. También se valoró en el inicio usar metodologías ágiles como Scrum [28], donde el cliente forma parte del equipo de desarrollo para aportar ideas y cambios durante el desarrollo del proyecto pero finalmente fue descartada. A continuación se analizan las ventajas e inconvenientes de ambas metodologías, concretando el por qué se decidió finalmente utilizar desarrollo en cascada.

- El producto a desarrollar se decide desde el inicio, al contrario que en Scrum, donde en cada iteración el cliente valora el producto exigiendo cambios o nuevas funcionalidades para el mismo. En este proyecto se ha tenido en cuenta desde el inicio el producto a desarrollar, siendo este un sistema de carga de ejercicios y la implementación de algunos de ellos basados en la plataforma VIRTRA-EL.
- Las etapas de desarrollo son herméticas, lo cual implica que hasta que no se acaba el desarrollo de una, no se puede avanzar a la siguiente. Por su lado, Scrum divide su desarrollo en “*Sprints*”, al final de cada uno se debe de haber creado un incremento terminado, utilizable y desplegable del producto final. El sistema de Sprints que Scrum plantea es muy interesante, pero dado que el proyecto iba a ser desarrollado por una sola persona y esta no estaba familiarizada con el desarrollo web en general, se decidió optar de nuevo por el desarrollo en cascada, fijando las fases de desarrollo desde un inicio, cada una con un objetivo concreto.
- La presencia del cliente no es obligatoria durante el desarrollo. Aunque durante el desarrollo del proyecto se consultaría paulatinamente a los tutores de este, no sería necesario su constante presencia durante el desarrollo, al contrario que en Scrum, donde un cliente forma parte del equipo de desarrollo para verificar que el producto desarrollado es el que se requiere.
- Dado que el diseño se realiza en una etapa temprana de desarrollo, se pueden estudiar las tecnologías con las que se va a trabajar desde el inicio, aplicando estos conocimientos al diseño del sistema. De esta forma, se sientan unas bases desde el principio, las cuales evitan problemas de diseño e implementación en futuras fases de desarrollo, al contrario que en Scrum, donde pueden surgir problemas debido a cambios continuos exigidos por parte del cliente o problemas de implementación por falta de diseño. Este tipo de problemas se almacenan en el “*Impediments Backlog*”.
- Un aspecto positivo de Scrum respecto al desarrollo en cascada es la capacidad de obtener requisitos de calidad, sin embargo, dado que el proyecto se basa en la migración de una aplicación a nuevas tecnologías, no ha sido necesario la toma de muchos requisitos, pues se ha tomado como base el proyecto original.

A continuación se muestra la [figura 10](#), la cual compara las características de ambas metodologías:



	Cascada	SCRUM
Cambios durante el desarrollo	El producto a desarrollar se decide desde el inicio y no cambia durante el desarrollo	Durante cada iteración el producto puede cambiar, añadiendo cambios o funcionalidades
Etapas de desarrollo	Las etapas son herméticas, hasta que no finaliza una etapa de desarrollo no comienza la siguiente	El desarrollo se divide en "Sprints", al final de cada uno se debe de haber creado un incremento terminado, utilizable y desplegable del producto final
Presencia del cliente	La presencia del cliente no es obligatoria durante el desarrollo	La presencia del cliente es obligatoria durante el desarrollo ya que aporta feedback constante sobre el producto que se está desarrollando
Desarrollo	El diseño se realiza en una etapa temprana, permitiendo estudiar las tecnologías y evitando problemas en futuras etapas	Pueden existir problemas de implementación por falta de diseño o complicaciones por cambios continuos exigidos por el cliente
Obtención de requisitos	El producto final puede no satisfacer las expectativas del cliente por problemas de comunicación al inicio del desarrollo	Obtención de requisitos de calidad dado que el cliente está implicado en el desarrollo

Figura 10. Tabla comparativa de metodologías

Como adición al desarrollo en cascada, se han utilizado también tableros Kanban [29] para llevar un control sobre las tareas (ver [figura 11](#)). Los tableros Kanban, como su propio nombre indica, son tableros utilizados por equipos de desarrollo para llevar un control sobre las tareas necesarias para el desarrollo del proyecto. Estos tableros se dividen en distintas subsecciones, las cuales pueden variar según las necesidades del equipo. Por lo general, siempre suele haber tres bloques principales: tareas a realizar, en progreso y completadas. De esta forma, al ser muy visuales estos tableros, se puede llevar un control exhaustivo sobre las tareas, pudiendo modificar el estado de las tareas o añadiendo nuevas si fuese necesario.

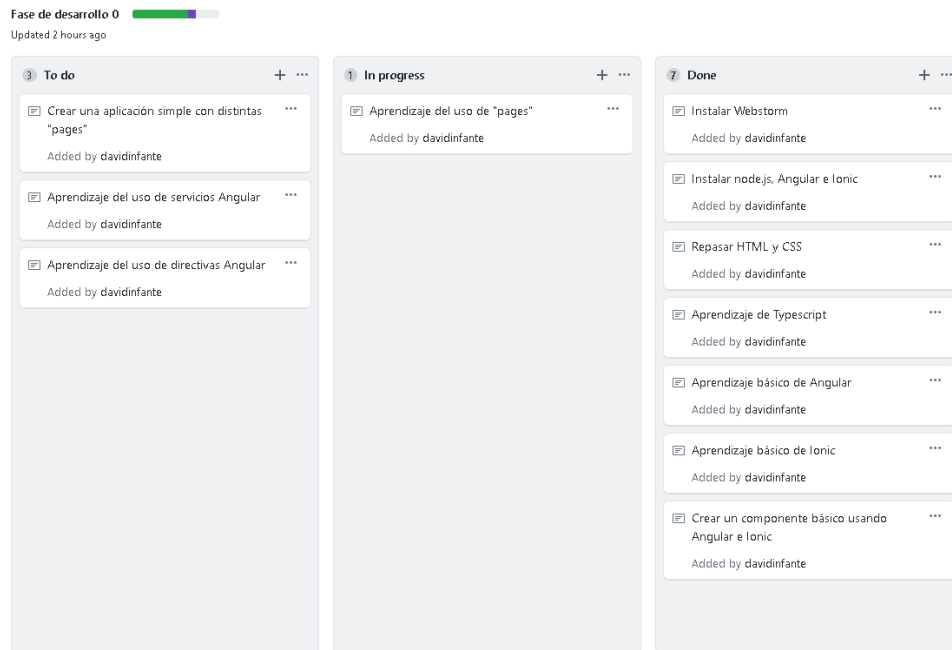


Figura 11. Ejemplo de tablero Kanban

Por otro lado, respecto a las tecnologías que podrían usarse para el desarrollo de este proyecto, se consideraron varias opciones entre las que destacan Angular, React y Vue [30][31]. React es una biblioteca creada por Facebook que ayuda al desarrollo de interfaces de usuario y que, al igual que Angular, se basa en componentes. Por otro lado, Vue.js es un framework de desarrollo para el frontend con gran versatilidad y rendimiento. Se van a comparar las ventajas e inconvenientes de estas tecnologías:

- Posibilidad de usar el patrón de diseño Modelo-Vista-Controlador (MVC). Uno de los principales motivos por los que se decidió elegir Angular frente a otras opciones es que incorpora el patrón modelo-vista-controlador, esto implica una mejor modularización del código y proyecto en general.
- Angular está soportado por Google y React por Facebook, lo cual es una gran fuente de confianza a la hora de escoger una de estas tecnologías, pues podemos confirmar que van a ser actualizadas durante un futuro próximo. Por otro lado, Vue, pese a no estar apoyado por una gran compañía y tener la menor comunidad de desarrolladores de las tres, lleva varios años de desarrollo y actualizaciones, lo cual tampoco lo hace una mala opción.
- Respecto a la documentación disponible sobre cada tecnología, Angular y Vue cuentan con distintas fuentes de recursos para aprender y estar al día con actualizaciones. Sin embargo, React, a pesar de ser actualizado habitualmente, no cuenta con documentación suficiente.
- La curva de aprendizaje de Vue es bastante buena, siendo ideal tanto para desarrolladores noveles como expertos en la tecnología. Por su parte, Angular, al tener tantas características, incrementa la dificultad de aprendizaje. Por último, React, al contar con menos documentación que las las dos propuestas anteriores, podríamos considerarlo el peor en este aspecto.

- Angular, aunque cuenta con un buen rendimiento, se queda atrás respecto a React y Vue, sin embargo cuenta con el mejor tiempo de respuesta en manipulación del DOM, dicho de otro modo, la lista de objetos organizados en forma de árbol que se usan internamente para formar lo que el usuario ve, es decir, la interfaz gráfica [8]. Tanto Vue como React tienen un rendimiento y tiempo de respuesta excepcional usando DOM virtual y renderizando solo aquellos nodos necesarios en cada momento. El funcionamiento del DOM virtual se basa en la creación de árboles virtuales basados en el DOM original cada vez que se realiza un cambio. De este modo, se busca el árbol que menos cambios realice desde el original, por tanto, se renderizan los menos cambios posibles.
- Aunque Angular cuenta con muchas características que son de utilidad a la hora de desarrollar, esto hace que, en general, las aplicaciones Angular sean mucho más pesadas que las que se desarrollan utilizando React y Vue.
- Mientras que React y Vue utilizan JavaScript como lenguaje de desarrollo, Angular utiliza TypeScript [9], una variante de JavaScript compilada, tipada y orientada a objetos, lo cual ofrece grandes ventajas. Entre estas ventajas destacan una mayor facilidad de desarrollo al poder detectar errores en tiempo de compilación y mayor consistencia y limpieza en el código al ser tipado y orientado a objetos.
- Concretamente para este proyecto, los componentes Angular son ideales, ya que permiten la encapsulación y modularización de elementos web de forma sencilla y limpia. En este caso, el poder crear los ejercicios de la plataforma en componentes Angular es una gran ventaja.

En la [figura 12](#) se muestra una tabla comparativa de Angular, React y Vue:

	Angular	React	Vue
Incorpora el patrón Modelo-Vista-Controlador (MCV)	Sí	No	No
Empresa desarrolladora	Google	Facebook	Equipo de desarrollo Vue
Documentación	Gran cantidad de recursos para aprender a desarrollar en Angular	Debido a las frecuentes actualizaciones del framework, la documentación es algo pobre	Comunidad muy activa y buenos recursos para aprender y estar al día con actualizaciones
Curva de aprendizaje	Al disponer de tantas características	Al contar con poca documentación respecto a nuevas	Aumenta lentamente, siendo una

	diferentes que son necesarias aprender, la curva de aprendizaje es abrupta	versiones, lo consideramos un framework complejo de aprender	tecnología sencilla de aprender pero con mucho contenido y características
Tiempos de respuesta y rendimiento	Cuenta con el mejor tiempo de respuesta en manipulación del DOM	Hace uso de DOM virtual renderizando solo aquellos nodos necesarios, tiene muy buenos tiempos de respuesta y rendimiento	Hace uso de DOM virtual renderizando solo aquellos nodos necesarios, tiene muy buenos tiempos de respuesta y rendimiento
Peso de la aplicación	Alto	Bajo	Bajo
Lenguaje de programación	TypeScript	JavaScript	JavaScript

Figura 12. Tabla comparativa de tecnologías

Considerando las ventajas e inconvenientes de cada una de las tres opciones, se valoró positivamente el uso de Angular para este proyecto, siendo seleccionado finalmente.

Como se ha comentado anteriormente, para dotar a la aplicación de un aspecto visual renovado y moderno, se optó por utilizar algún “*cross-platform framework*” para el desarrollo de interfaces de usuario. Las dos principales apuestas a tener en cuenta han sido Ionic y React Native. La decisión de escoger uno de los dos está muy ligada a la elección de tecnología base para el proyecto; si se escogía React, React Native era la opción por defecto, mientras que si la elección era Angular, Ionic era el framework a considerar [10]. Finalmente, con la elección de Angular como tecnología base para el proyecto, se optó por usar Ionic como complemento.

## 2.3. Trabajos relacionados

Con el fin de comprobar que tanto Angular como Ionic se utilizan en desarrollos relevantes en la actualidad, se han recopilado algunos ejemplos de aplicaciones que usan estas tecnologías:

Aplicaciones desarrolladas usando las últimas versiones de Angular [\[11\]](#):

- Microsoft Office Home [\[12\]](#): Microsoft hace uso de Angular en su página de Office Home, haciendo uso de las ventajas que Angular ofrece como por ejemplo las directivas estructurales que modifican la vista según los datos del modelo.
- Delta [\[13\]](#): La página web de la aerolínea estadounidense Delta, hace uso de Angular, ofreciendo la posibilidad de navegar por la mayor parte de esta sin necesidad de recargar la página.
- Banco Santander Brasil [\[14\]](#): La filial brasileña del banco Santander hace uso de Angular en su página principal, modularizando y estructurando la vista para ofrecer una mejor experiencia de usuario.
- Forbes [\[15\]](#): La prestigiosa revista de negocios y finanzas, Forbes, también ha desarrollado su web utilizando Angular, dotándola de una estructura dinámica y orgánica. Aprovechando las ventajas que ofrece Angular mediante sus directivas, es sencillo crear plantillas para las noticias mejorando y acelerando el proceso.
- Blender Video [\[16\]](#): La fundación Blender, desarrolladora del software para el modelado, animación, iluminación y renderización de gráficos en tres dimensiones, hace uso de Angular en su página web Blender Video, destinada a la reproducción de archivos multimedia relacionados con Blender. Angular permite realizar distintas tareas en una misma página, siendo ideal para este tipo de webs, pudiendo realizar distintas acciones mientras el video se reproduce.

Las principales compañías que utilizan Ionic framework son: NASA, BMW, Electronic Arts o NHS [\[17\]](#). Algunos ejemplos de aplicaciones que también implementan esta tecnología son:

- 86400 [\[18\]](#): Es la primera aplicación australiana de banca inteligente, con constantes actualizaciones, desarrollada usando Angular e Ionic, permite a los desarrolladores mejorar la experiencia de usuario mediante gráficos, interacciones y animaciones a la vez que mejorando el rendimiento de la aplicación.
- Sworkit [\[19\]](#): Una de las principales aplicaciones fitness utilizada por millones de usuarios. El equipo de desarrollo detrás de Sworkit, ha desarrollado un total de nueve aplicaciones utilizando Ionic.
- Instant Pot [\[20\]](#): La aplicación oficial de la olla a presión número uno en Estados Unidos. La aplicación utiliza las ventajas que Ionic ofrece para compartir recetas y consejos sobre el uso de la olla.
- Untappd [\[21\]](#): Con cuatro millones de usuarios, esta aplicación para encontrar y conocer entusiastas de la cerveza migró su antigua interfaz de usuario a Ionic. Gracias a este cambio hace uso de “*Toggles*” y “*Checkmarks*” para mejorar la usabilidad de la aplicación y ofrecer la

mejor experiencia de usuario.

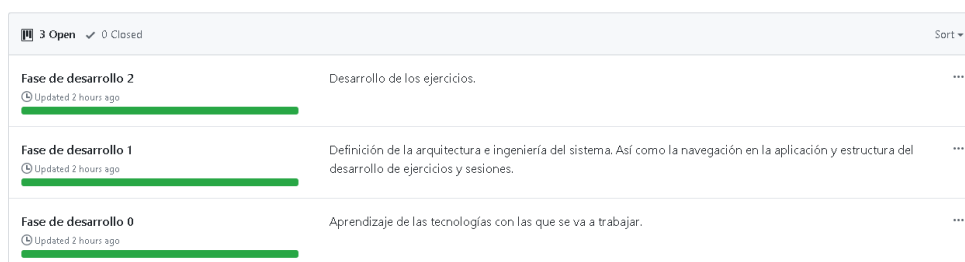
## 3. Propuesta

### 3.1. Metodología de trabajo

Para la realización del proyecto se ha seguido una metodología de desarrollo en cascada. El uso de esta metodología aporta grandes ventajas a proyectos de una baja duración como este, por ejemplo:

- La definición del producto es clara y se realiza desde el inicio, por tanto se tiene desde el principio un objetivo claro al que llegar.
- Las tecnologías con las que se va trabajar no varían durante el desarrollo, evitando problemas de compatibilidad y el tener que aprender nuevas tecnologías continuamente.

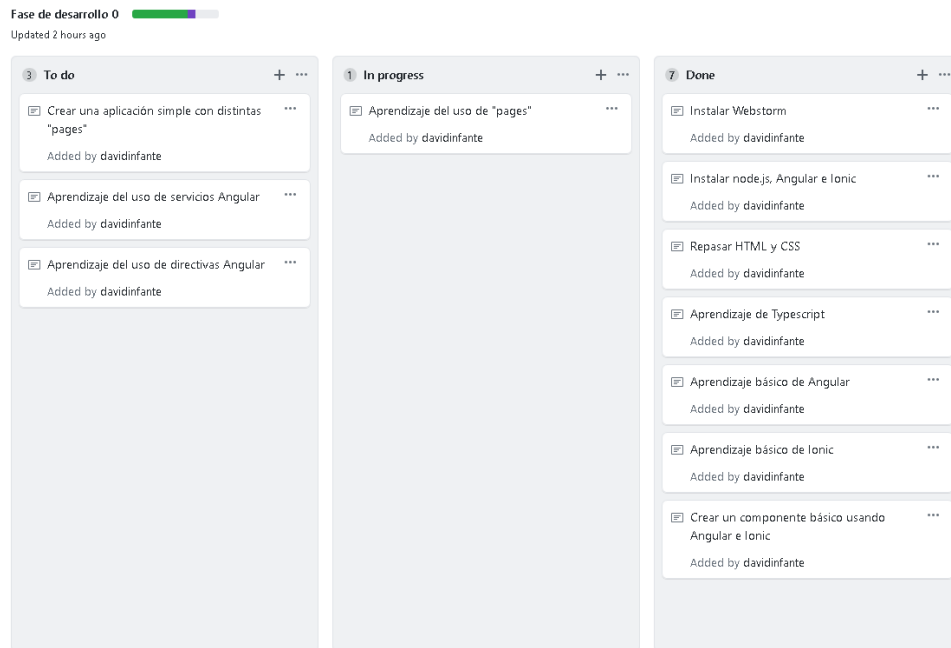
Dado que el equipo de desarrollo ha estado formado por una sola persona, se ha optado por crear un backlog de tareas, subdivididas en tres fases de desarrollo (ver [figura 13](#)), de las cuales, en la primera se ha estudiado las tecnologías y el dominio del problema a resolver, la segunda fase coincide con la primera y segunda de la metodología de desarrollo en cascada, y la tercera fase de desarrollo coincide con la segunda, tercera y cuarta de la metodología de desarrollo en cascada.



3 Open ✓ 0 Closed		Sort ▼
<b>Fase de desarrollo 2</b> Updated 2 hours ago	Desarrollo de los ejercicios.	...
<b>Fase de desarrollo 1</b> Updated 2 hours ago	Definición de la arquitectura e ingeniería del sistema. Así como la navegación en la aplicación y estructura del desarrollo de ejercicios y sesiones.	...
<b>Fase de desarrollo 0</b> Updated 2 hours ago	Aprendizaje de las tecnologías con las que se va a trabajar.	...

Figura 13. Fases de desarrollo del proyecto

Para llevar un mejor control sobre las tareas, se ha hecho uso de la herramienta “*projects*” de la plataforma GitHub. Esta herramienta permite controlar y modificar las tareas de desarrollo y la fase en la que se encuentran mediante tableros Kanban (ver [figura 11](#)). Estos tableros divididos en columnas según el estado de desarrollo de cada tarea son muy útiles para llevar un control sobre el proyecto y son sencillo de usar. Describiremos a continuación cada una de las fases y mostraremos sus tableros.



[Figura 11](#). Tablero Kanban utilizado en el proyecto

**La fase de desarrollo 0,** " (ver [figura 14](#)), se ha centrado en el aprendizaje de las tecnologías con las que se iba a trabajar durante el proyecto, esto incluye, Angular (TypeScript, HTML y SCSS) e Ionic, incluyendo todos sus diferentes aspectos. Para ello fue vital comprender en funcionamiento de los componentes, servicios, directivas, pages y componentes Ionic.

En esta fase, se realizaron pequeñas aplicaciones para poder practicar y comprender mejor estas tecnologías antes de comenzar con el desarrollo del proyecto. Se estudió la implementación original de VIRTRA-EL, para comprender su funcionamiento y se realizaron distintas pruebas con la plataforma para comprender la interacción con la misma.



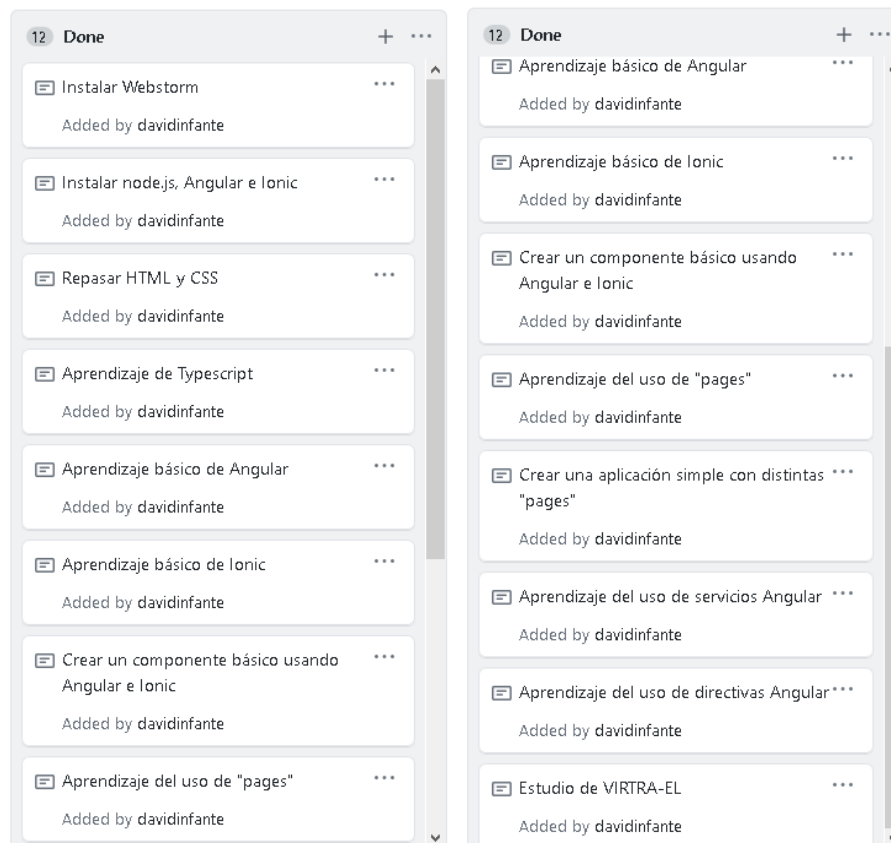


Figura 14. Tablero Kanban de tareas realizadas en la Fase de desarrollo 0

**Durante la fase de desarrollo 1**, (ver [figura 15](#)), se definió la arquitectura e ingeniería del sistema, la navegación en la aplicación y por último la estructura del sistema de carga de ejercicios y sesiones. La aplicación está pensada para ser utilizada en dispositivos móviles también, lo cual se ha tenido en cuenta a la hora de diseñar e implementar.

En esta fase se implementaron las distintas páginas del sistema (Inicio, Proyecto, Participantes, Contacto, Acceder, Registro, Ejercicios, Sesión), así como los componentes más básicos, el Header, Footer, Header de usuario y Asistente. También se desarrolló el sistema de carga dinámica de ejercicios. Por último, se desarrolló un backend utilizando Node.js y MongoDB como base de datos para el intercambio de información entre el frontend y el backend.

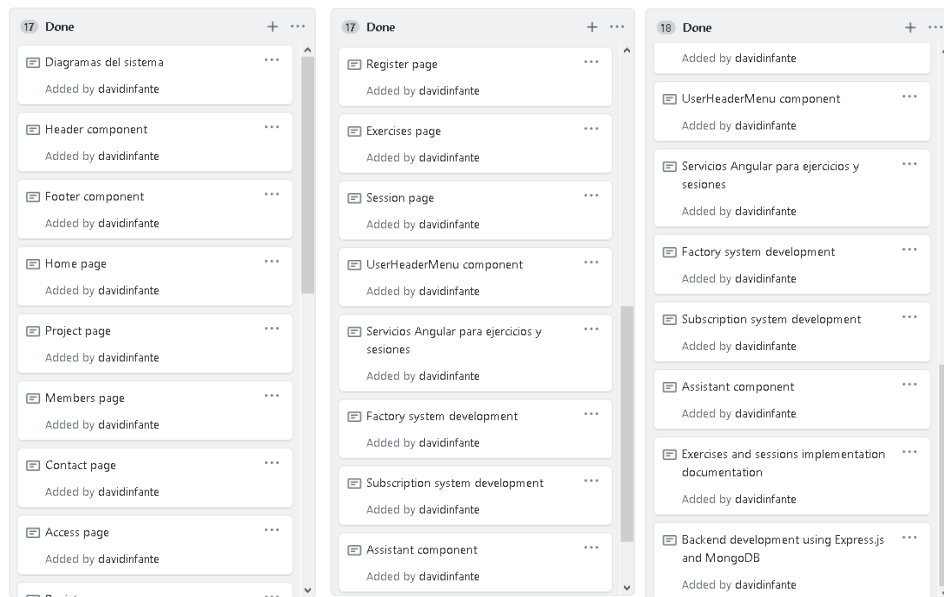


Figura 15. Tablero Kanban Fase de desarrollo 1

La última fase de desarrollo, denominada “Fase 2” (ver [figura 16](#)), se centró única y exclusivamente en el desarrollo de los ejercicios para la plataforma. Esto incluye la parte visual, la lógica que hace funcionar a cada ejercicio, los datos que usa el ejercicio y el intercambio de los mismos entre el frontend y backend.

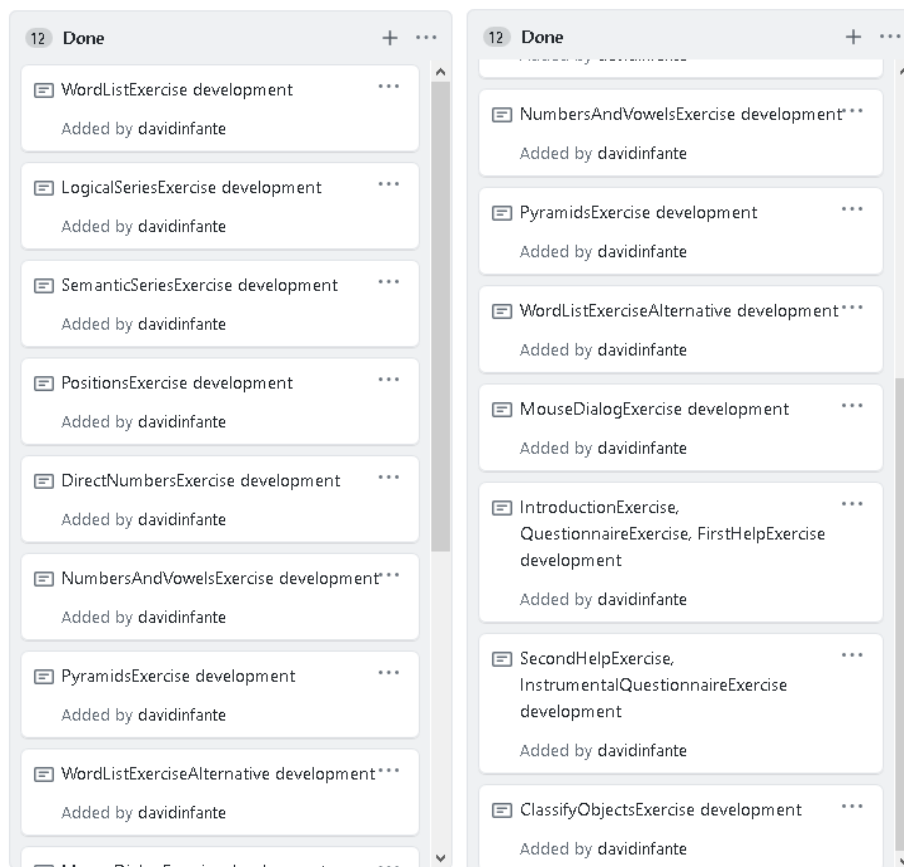


Figura 16. Tablero Kanban Fase de desarrollo 2

En la [figura 17](#) se puede ver la planificación temporal que se ha seguido durante los tres meses de proyecto, todas las tareas de cada Fase, sus fechas de inicio, fin y total de horas de trabajo en cada una de las tareas.

Fase de desarrollo	Tarea	Fecha Inicio	Fecha Fin	Horas de trabajo
Fase 0	Instalar Webstorm	05/03/20	05/03/20	1
	Instalar node.js, Angular e Ionic	05/03/20	05/03/20	1
	Repasar HTML y CSS	06/03/20	06/03/20	3
	Aprendizaje de Typescript	07/03/20	09/03/20	4
	Aprendizaje básico de Angular	07/03/20	09/03/20	8
	Aprendizaje básico de Ionic	10/03/20	11/03/20	4
	Crear un componente básico usando Angular e Ionic	12/03/20	13/03/20	5
	Aprendizaje del uso de "pages"	14/03/20	14/03/20	2
	Crear de una aplicación simple con distintas "pages"	15/03/20	16/03/20	4
	Aprendizaje del uso de servicios Angular	17/03/20	17/03/20	3
	Aprendizaje del uso de directivas Angular	18/03/20	18/03/20	2
	Estudio de VIRTRA-EL	18/03/20	19/03/20	4
Fase 1	Diagramas del sistema	20/03/20	20/03/20	5
	Header component	21/03/20	22/03/20	6
	Footer component	23/03/20	24/03/20	4
	Home page	25/03/20	26/03/20	3
	Project page	25/03/20	26/03/20	1
	Members page	25/03/20	26/03/20	1
	Contact page	25/03/20	27/03/20	2
	Access page	25/03/20	28/03/20	2
	Register page	25/03/20	28/03/20	2
	Exercises page	25/03/20	28/03/20	2
	Session page	29/03/20	02/03/20	6
	UserHeaderMenu component	29/03/20	29/03/20	2
	Servicios Angular para ejercicios y sesiones	30/03/20	30/03/20	4
	Desarrollo del sistema factory	31/03/20	02/04/20	10
	Desarrollo del sistema subscription	31/03/20	02/04/20	6
	Assistant component	31/03/20	02/04/20	4
Fase 2	Documentación sobre la implementación de ejercicios y sesiones	03/04/20	03/04/20	2
	Desarrollo del backend y conexión con la base de datos en MongoDB	03/04/20	05/04/20	8
	WordListExercise development	05/04/20	07/04/20	8
	LogicalSeriesExercise development	05/04/20	12/04/20	12
	SemanticSeriesExercise development	10/04/20	20/04/20	6
	PositionsExercise development	15/04/20	29/04/20	15
	DirectNumbersExercise development	30/04/20	04/05/20	14
	NumbersAndVowelsExercise development	05/05/20	05/05/20	4
	PyramidsExercise development	06/05/20	06/05/20	8
	WordListExerciseAlternative development	07/05/20	07/05/20	6
	MouseDialogExercise development	08/05/20	08/05/20	4
	IntroductionExercise, QuestionnaireExercise, FirstHelpExercise development	09/05/20	10/05/20	6
	SecondHelpExercise, InstrumentalQuestionnaireExercise development	11/05/20	13/05/20	3
	ClassifyObjectsExercise development	14/05/20	17/05/20	10

Figura 17. Tabla de planificación temporal del proyecto

## 3.2. Fase 0

Durante la fase 0 de desarrollo se estudiaron las distintas tecnologías con las que se iba a trabajar durante el proyecto. Esta fase no cuenta con demasiada información relevante para el desarrollo del proyecto pues se basa en la práctica para aprender las tecnologías a utilizar.

Antes de comenzar con este estudio, se instalaron aquellas tecnologías que serían necesarias durante el desarrollo del proyecto. Se instaló el entorno de desarrollo “Webstorm”, ideal para el desarrollo utilizando Angular e Ionic. A continuación se instaló Node.js y utilizando su gestor de paquetes “npm” se instalaron Angular e Ionic.

Una vez instaladas las tecnologías se hizo un repaso de las etiquetas básicas de HTML y de algunas de las propiedades de CSS, dado que Angular trabaja con estos lenguajes. Para finalizar el estudio básico de los lenguajes con los que trabaja Angular, se estudió TypeScript. Para ello se hizo uso de la documentación que podemos encontrar en la propia web de TypeScript [\[22\]](#). Esta documentación incluye los tipos básicos de los que consta el lenguaje, una guía rápida de aprendizaje y una serie de ejemplos que poder modificar para empezar a practicar.

Comprendidas las bases de estos lenguajes y habiendo practicado un poco con ellos se comenzó a estudiar y practicar con Angular. Fundamentalmente se utilizaron las guías en la web de Angular para el desarrollo de esta tarea. Primero, se realizó un tutorial básico sobre aplicaciones Angular [\[23\]](#). Con este tutorial se comprendió el funcionamiento de los componentes Angular, algunos de sus tipos y directivas. Una vez terminado, se realizó el tutorial “Tour of Heroes” [\[24\]](#). Este segundo tutorial se centró más en temas como la utilización de servicios Angular y la obtención de datos de un servidor mediante llamadas HTTP.

A continuación se estudió Ionic. Como se había estudiado Angular previamente, el aprendizaje de Ionic fue más rápido usando para ello la documentación oficial de Ionic [\[25\]](#). Durante esta tarea se estudió el funcionamiento de las páginas de Ionic denominadas “pages” y los componentes que ofrece el framework [\[26\]](#).

Por último y para finalizar esta fase de desarrollo se realizaron distintas pruebas con la versión original de la plataforma VIRTRA-EL con el fin de conocer su funcionamiento y aspecto visual. Algunos aspectos de esta implementación se detallan en las secciones [Dominio del problema a resolver](#) y [Ejercicios y sesiones](#).

Una vez estudiadas las tecnologías con las que se iba a trabajar, comienza la Fase 1 de desarrollo, donde se diseñarán e implementarán las bases del sistema.

### 3.3. Fase 1

Durante esta fase de desarrollo se diseñan e implementan las bases del sistema, incluyendo su arquitectura, navegabilidad, componentes básicos, estructura de los ejercicios y sesiones, sistema de carga dinámica de ejercicios, el backend, la base de datos y las comunicaciones en el sistema. Dado que también se decidió que la aplicación pudiese ser utilizada también en dispositivos móviles, a lo largo de esta fase se diseñarán y desarrollarán elementos específicos para la versión móvil y para la de sobremesa.

#### 3.3.1. Arquitectura y navegabilidad del sistema

Para comenzar el proyecto se realizó un diagrama de la arquitectura del sistema (ver [figura 19](#)) incluyendo el paso de información y la interacción del usuario con el sistema. Este diagrama se centra en la arquitectura de la parte de los ejercicios, objetivo principal de este proyecto. Cabe destacar que la base de datos ha cambiado en esta nueva versión de relacional con MySQL a NoSQL con MongoDB, lo cual se detalla en la sección [Comunicaciones frontend-backend](#). En la [figura 18](#) se puede observar una leyenda de los iconos usados en este diagrama.

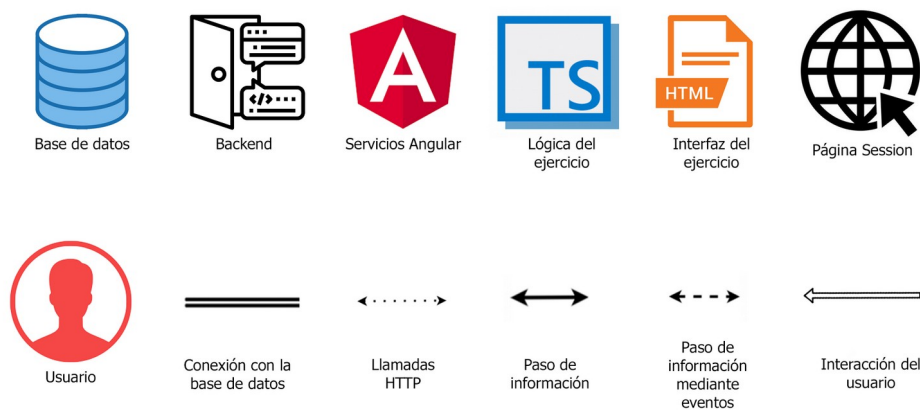


Figura 18. Leyenda arquitectura del sistema

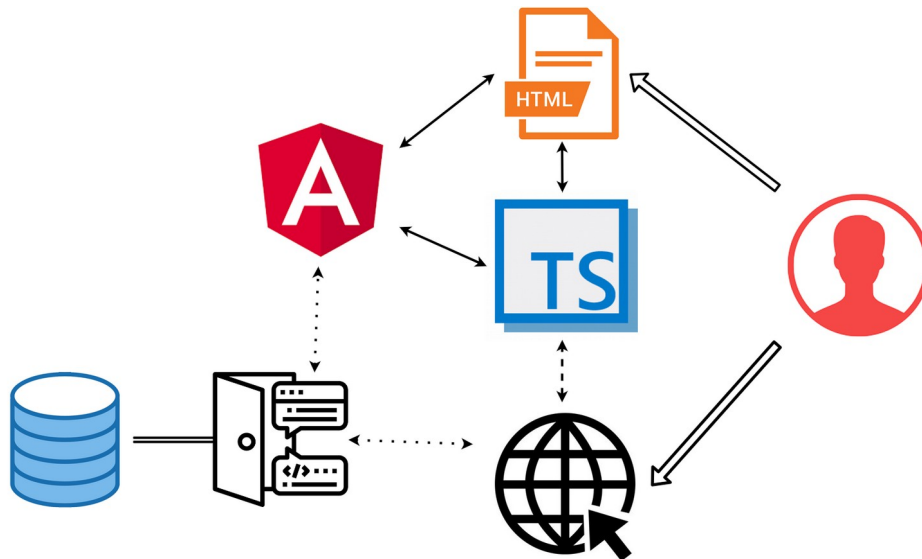


Figura 19. Arquitectura del sistema

El diagrama de la arquitectura del sistema ([figura 19](#)) se divide en tres partes principales. A la izquierda se puede apreciar la base de datos y el backend, el cual intercambia información con servicios Angular y la página Session. Por su lado, los servicios Angular proporcionan la información a la lógica e interfaz de los ejercicios y estos intercambian información entre sí. Para notificar el fin de un ejercicio o el paso de parámetros de inicio, la página Session intercambia información con la lógica del ejercicio mediante eventos. Por último, el usuario interactúa tanto con la página Session como con los ejercicios que se muestran en ella.

Antes de comenzar con el desarrollo de los ejercicios se realizó una estructura base de páginas del sistema. El sistema de navegación en la aplicación es el siguiente: la aplicación consta de una serie de páginas por las que el usuario podrá navegar libremente. Desde las páginas “Exercises” y “Access” se accede a la página “Session”, lugar donde se encuentran los ejercicios que el usuario debe realizar. Dicha estructura es igual a la del proyecto original y se desarrolló con el fin de dotar al proyecto de cierta similitud al original. La navegabilidad entre páginas se muestra en el siguiente diagrama (ver [figura 20](#)).

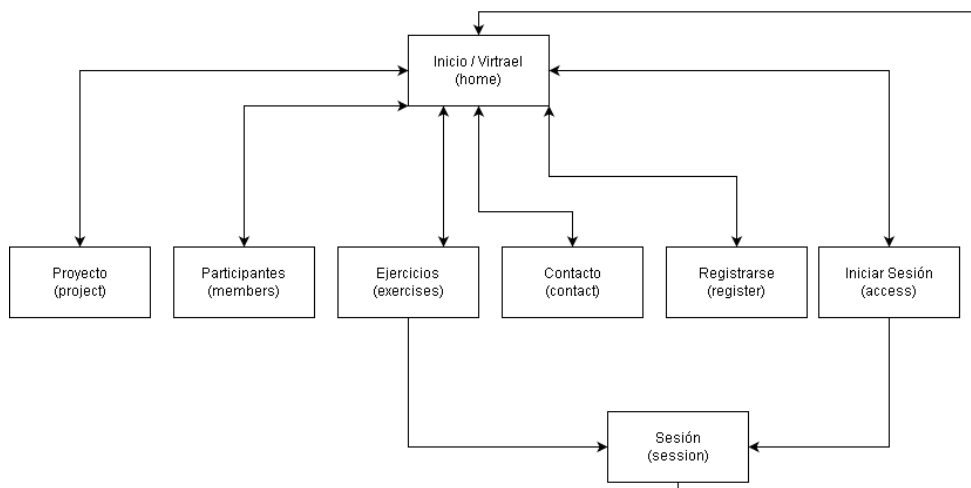


Figura 20. Diagrama de navegabilidad

### 3.3.2. Páginas del sistema

El sistema, tal y como se ha detallado en la sección anterior, se divide inicialmente en una serie de páginas por las que el usuario puede navegar libremente antes de acceder a la zona de ejercicios.

Estas páginas se han desarrollado utilizando Ionic pages, las cuales se encargan de encapsular y mostrar páginas específicas, identificadas por URLs. El contenido de estas es extremadamente similar al de las páginas originales, solo se han modificado algunos aspectos visuales, como las entradas de texto y los botones, que han sido adaptadas utilizando componentes proporcionados por Ionic para el desempeño de estas tareas. Estos cambios se han realizado con el fin de dotar a las interfaces de un aspecto más moderno. Los cambios que se han realizado respecto a la versión original y el contenido de las páginas se detallan a continuación:

- home: muestra información básica de la web (ver [figura 21](#)).



Figura 21. Home page

- project: recoge información sobre el proyecto y su desarrollo (ver [figura 22](#)).

#### VIRTRAEL

##### Plataforma Virtual de Evaluación e Intervención Cognitiva en Mayores

El objetivo principal de este proyecto es desarrollar e implementar una plataforma virtual para la evaluación y estimulación cognitiva de personas mayores a través de Internet. Se trata de una estrategia destinada a la prevención e intervención sobre el deterioro cognitivo disponible para muchas personas y que será accesible a aquellas personas mayores que viven alejadas de los recursos asistenciales y con dificultades de desplazamiento para acceder a ellos. Las pruebas de evaluación y las estrategias de estimulación cognitiva tendrán un enfoque ecológico con el fin de obtener una mejor estimación del grado de discapacidad y dependencia de la persona evaluada y como resultado final una disminución de ésta.

Nuestra finalidad última es prevenir y retrasar la dependencia, así como intervenir en sus estadios iniciales. Para alcanzar nuestro objetivo se ofrecen nuevos instrumentos de evaluación neuropsicológica y funcional que permitirán realizar una evaluación de la persona mayor a distancia, y nuevas tareas que permitirán realizar la estimulación o rehabilitación a distancia de esas personas. Para ello, VIRTRA-EL permite la configuración y administración de las tareas, supervisión y monitorización en línea de las actividades que realiza la persona mayor, así como el seguimiento de su evolución.

La plataforma software se ha desarrollado prestando especial atención al cumplimiento de ciertos atributos de calidad (escalabilidad, portabilidad, usabilidad, extensibilidad, etc.) que garanticen fundamentalmente:

1. Adaptación a perfiles de usuario e incluso ciudadanos concretos
2. Fácil implantación, configuración, mantenimiento y ampliación para asegurar la operatividad incluso en ausencia de personal experto, existiendo como única necesidad la disponibilidad de un ordenador con acceso a Internet (domicilios particulares y centros de acceso público/privado a Internet), con el propósito de abarcar todo el territorio andaluz.

Además, en este proyecto se han incorporado tecnologías de realidad virtual y aumentada que dotarán a las pruebas de evaluación y estimulación cognitiva de una mayor validez ecológica y de una mejor estimación del grado de discapacidad y dependencia de la persona evaluada.

Figura 22. Project page

- members: lista de miembros y entidades que han trabajado en el desarrollo del proyecto (ver [figura 23](#)).

## Participantes del proyecto

### Grupo MYDASS de la UGR

- Rodríguez Fórtiz, María José
- Hurtado Torres, María Vistación
- Espín Martín, Vanesa
- Rodríguez Domínguez, Carlos
- Benghazi Akhlaki, Kavatar
- Fernández López, Álvaro
- Garrido Sullejos, José Luis
- Hornos Barranco, Miguel Juan
- Noguera García, Manuel
- Rodríguez Almendros, María Luisa
- Samos Jiménez, José

### Grupo TARVIS

- Revelles Moreno, Jorge

### Grupo GIIG

- Cano Olivares, Pedro

### Grupo del Departamento de Informática de la Universidad de Almería

- Torres Gil, Manuel

### Grupo de Envejecimiento y Salud

- Villaverde Gutiérrez, María del Carmen
- Ramírez Rodrigo, Jesús María
- Argente del Castillo Lechuga, Josefa
- Ruiz Villaverde, Alberto

### Grupo NeuroPsicología y Psiconeuroinmunología Clínica

- Caracul Romero, Alfonso
- Pérez García, Miguel
- Rute Pérez, Sandra
- Santiago Ramajo, Sandra
- Vilar López, Raquel

### Grupo de la Escuela Andaluza de Salud Pública

- Espín Balbino, Jaime
- García Mochón, Leticia

### Empresas colaboradoras

- Virtual Solutions
- Virtum Graphics
- Everyware Technologies

### Entidades Colaboradoras

- Consorcio Fernando de los Ríos (aulas Guadalinfo)
- Ayuntamiento de Granada
- Ayuntamiento de Maracena
- Diputación de Granada
- Federación de Organizaciones Andalcas de Mayores
- Fundación Andaluza de Servicios Sociales
- CITIC-UGR

Figura 23. Members page

- **exercises:** acceso a los ejercicios de la plataforma, la página contiene un login (ver [figura 24](#)). Para dotar de un aspecto más moderno al formulario, este se ha recogido en una “*ion-card*”. Este elemento de Ionic permite dar al formulario este aspecto de “tarjeta”. Del mismo modo, se han utilizado los elementos “*ion-card-header*” e “*ion-card-content*” para el título y contenido del formulario. Para las entradas de texto se ha hecho uso de “*ion-label*” para etiquetar la entrada de texto e “*ion-input*” para su entrada. El texto de las etiquetas es flotante, y cambia una vez introducimos texto. Por último, en el botón acceder que es de tipo “*ion-button*” se ha añadido un icono proporcionado por Ionic para dotar de un mejor aspecto al botón. A continuación se muestra una comparación entre la nueva versión y la original (ver [figura 24](#)).

regístrate aquí'. It features input fields for 'E-Mail' and 'Contraseña', a checkbox for 'Mantener conectado', and a blue 'ACCEDER' button with a right-pointing arrow icon." data-bbox="292 520 762 650"/>

## Usuario no registrado

Para acceder a esta página deberás darte de alta como usuario de VIRTRA-EL.

Por favor introduce tus datos de usuario en el siguiente formulario o [regístrate aquí](#)

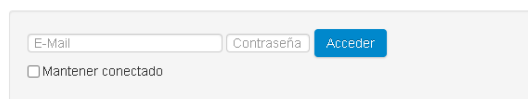


Figura 24. Exercises page

- **contact:** formulario de contacto (ver [figura 25](#)). Al igual que para la página “exercises”, de nuevo se ha utilizado el componente “*ion-card*”, “*ion-label*”, “*ion-input*” e “*ion-button*” para el desarrollo de este formulario. Como adición, se ha cambiado el color y añadido un icono al botón



“Limpiar” de modo que sea más visual y fácil de entender para el usuario. En la [figura 25](#) se muestra una comparativa entre la nueva versión y la original.

**Contacto**  
Si tiene alguna duda o desea más información sobre VIRTRA-EL, contacte con nosotros a través del siguiente formulario.

Nombre

E-Mail

Asunto

Texto

Resultado de 1 + 3

Esto evita que se haga un mal uso del formulario de contacto

[+ ENVIAR](#) [LIMPIAR](#)

---

**Contacto**  
Si tiene alguna duda o desea más información sobre VIRTRA-EL, contacte con nosotros a través del siguiente formulario.

Nombre

E-Mail

Asunto

Texto

Resultado de 10 + 1  Esto evita que se haga un mal uso del formulario de contacto.

[Enviar](#) [Limpiar](#)

Figura 25. Contact page

- register: formulario de registro a la plataforma (ver [figura 26](#)). De nuevo se ha introducido el uso de “*ion-card*”, “*ion-label*”, “*ion-input*” e “*ion-button*” para el desarrollo de este formulario. En la [figura 26](#) se ve una comparación entre la nueva versión y la original.

**Registro**  
Debes registrarte para poder realizar los ejercicios de VIRTRA-EL. Por favor, rellena el siguiente formulario. Si eres un terapeuta o un cuidador, por favor, [contacta con el administrador](#) para que te dé de alta.

Nombre

Apellidos

E-Mail

Confirmar E-Mail

Compruebe que su e-mail sea correcto. Se le enviará un mensaje para que confirme este registro antes de activar su cuenta.

Contraseña

Confirmar contraseña

Resultado de 1 + 3

Esto evita que se haga un mal uso del formulario de contacto

[+ ENVIAR](#) [LIMPIAR](#)

## Registro

Debes registrarte para poder realizar los ejercicios de VIRTRA-EL. Por favor, rellena el siguiente formulario. Si eres un terapeuta o un cuidador, por favor, [contacta con el administrador](#) para que te dé de alta.

Nombre

Apellidos

E-Mail

Confirmar E-Mail

Compruebe que su e-mail sea correcto. Se le enviará un mensaje para que confirme este registro antes de activar su cuenta.

Contraseña

Confirmar contraseña

Resultado de 8 + 5  Esto evita que se haga un mal uso del formulario de registro.

Figura 26. Register page

- access: login para acceder a las sesiones, muestra el mismo contenido que [ejercices](#) (ver [figura 27](#)). A continuación se muestra una comparación entre la nueva versión y la original.

**Iniciar Sesión**

Introduce tus datos de acceso a VIRTRA-EL para acceder.

E-Mail

Contraseña

☐ Mantener conectado

## Iniciar Sesión

Introduce tus datos de acceso a VIRTRA-EL para acceder.

E-Mail  Contraseña

☐ Mantener conectado

Figura 27. Access page

- session: vista en la que se muestran las sesiones y ejercicios, su contenido es dinámico (ver [figura 28](#)). Se ha utilizado también una “*ion-card*” lugar donde se cargarán los componentes de los ejercicios. En la sección [Página Session y componente Assistant](#) se darán más detalles sobre el funcionamiento de esta página y el sistema de carga de ejercicios en la misma.



**Sesión 1**

¡Bienvenido a VIRTRA-EL! Esta es tu primera sesión, así que te explicaremos en qué consistirá tu plan de trabajo. Por favor, trata de realizar la sesión completa sin interrupción.

Figura 28. Session page

Las páginas [exercises](#) y [access](#) no cuentan con un login funcional, para acceder a los ejercicios y sesiones basta con pulsar el botón “Acceder”. Dado que el desarrollo de esta funcionalidad no estaba previsto para este proyecto, se ha

dejado para una futura fase de desarrollo, en la que también se desarrollarían los formularios de las páginas “register” y “contact” (ver [Trabajos futuros](#)).

### 3.3.3. Componentes básicos del sistema

Los componentes básicos del sistema son: **Header** (y el **menú lateral**), **Footer** y **UserHeaderMenú**.

Para complementar estas páginas y añadir navegabilidad a la aplicación se crea el componente “**Header**” (ver [figura 29](#)). Este Header contiene una serie de botones que redireccionan a cada una de las páginas mencionadas en la sección anterior [Páginas del sistema](#). Este componente ya existía en la versión original de VIRTRA-EL y se ha adaptado a nuevas tecnologías con el fin de facilitar la navegación por la aplicación. En concreto, se ha desarrollado utilizando un componente Angular, para que pueda ser fácilmente mantenido en caso de necesitar añadir más botones o funcionalidades al mismo.

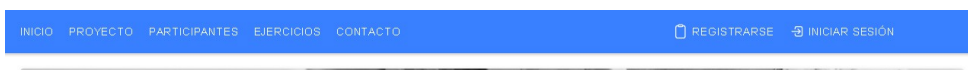


Figura 29. Header component versión escritorio

Para la versión móvil se ha programado que el header cambie a un **menú lateral** (ver [figura 30](#)). El uso de un menú lateral en aplicaciones de dispositivos móviles está muy extendido a día de hoy, es por esto que se decidió desarrollarlo, de forma que la navegación por la página sea lo más familiar y sencilla posible para el usuario. Utilizar un header como el de la [figura 29](#) en un dispositivo móvil no es viable dado que la pantalla es mucho más pequeña y no habría espacio suficiente para seleccionar los botones con precisión. Es por esto que se decidió utilizar un **menú lateral** como el previamente citado para la versión móvil.

Sin embargo, existe un “problema conocido” o “*known issue*” en inglés, relacionado a este menú lateral. Dicho menú se desarrolla utilizando “*ion-menu*”, un componente que Ionic ofrece para el desarrollo de menús laterales. Bajo ciertas circunstancias (el cierre y apertura continuada del menú aparentemente) el menú se bloquea, no siendo posible su utilización y siendo necesario refrescar la página para que pueda volver a usarse. El equipo de desarrollo de Ionic está trabajando en una solución a este problema y será solucionado en una futura actualización. A continuación se muestra el menú lateral:

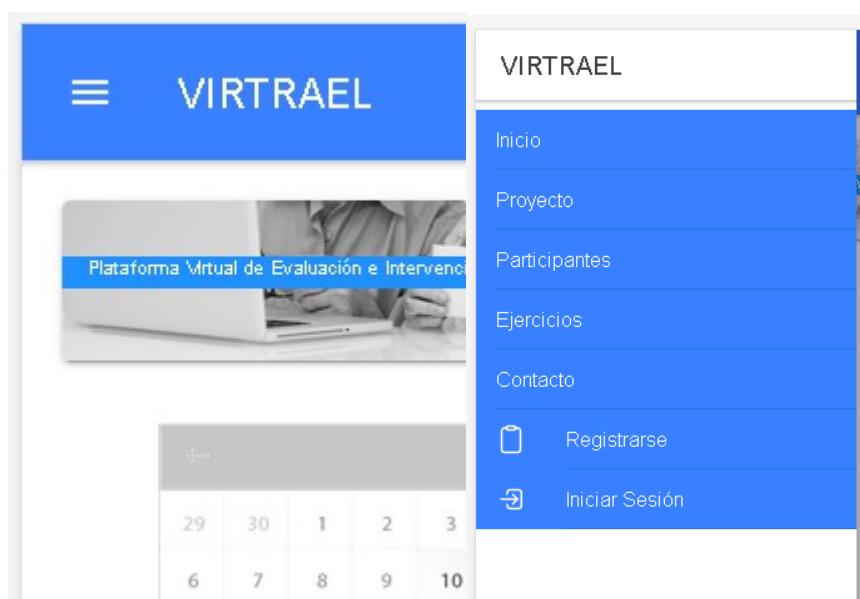


Figura 30. Header component versión móvil

También se ha añadido un componente **Footer** (ver [figura 31](#)) que contiene información referente a la plataforma. En su versión original incluía una serie de botones para cambiar el idioma de la página; en esta versión también los incluye, utilizando el sistema de internacionalización de Angular para la traducción de la página. Sin embargo, aunque se ha utilizado este sistema, las traducciones de los textos no han sido realizadas, y se realizarían en una futura fase de desarrollo. Al igual que el componente Header, el Footer también se ha desarrollado en un componente Angular para facilitar su mantenimiento.



Figura 31. Footer component

Por último, cuando el usuario ingresa en los ejercicios, el componente Header cambia a un “UserHeaderMenu” (ver [figura 32](#)). Este menú, al igual que el Header, cuenta con una vista distinta según si el dispositivo es un smartphone o tablet (ver [figura 33](#)), o un ordenador o portátil. Tal y como se mencionó en el apartado [Dominio del problema a resolver](#), las funcionalidades de este componente no han sido implementadas en este proyecto puesto que no estaba previsto su desarrollo, pero podrían ser desarrolladas en una futura iteración (ver [Trabajos futuros](#)).



Figura 32. UserHeaderMenu modo escritorio

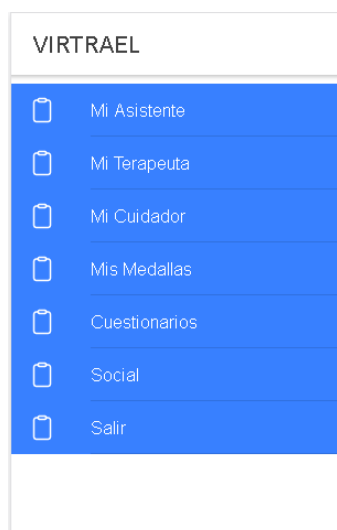


Figura 33. UserHeaderMenu modo móvil

### 3.3.4. Ejercicios y sesiones

Dado que la plataforma se basa en la evaluación y estimulación cognitiva de sus usuarios, un ejercicio es un elemento de la aplicación con el cual el usuario interactúa con un fin concreto, estimular su memoria, comprobar su razonamiento lógico o medir su nivel de atención por ejemplo.

En la versión original cada uno de los ejercicios estaban desarrollados de la siguiente forma:

- Un fichero con la especificación del ejercicio en JavaScript, indicando su título, fichero con interfaces gráficas y las funciones auxiliares necesarias para el desarrollo del ejercicio en cuestión.
- Fichero PHP con la interfaz gráfica del ejercicio en HTML, el cual incluye el CSS, así como las funciones que hacen funcionar el ejercicio.
- Un fichero denominado “series” con todas las posibles variantes que puede tener el ejercicio en cuanto a datos mostrados en formato XML. Estos datos son las imágenes o texto que muestra el ejercicio. Los datos se agrupan en distintas series, así se puede repetir el ejercicio con otros datos para que el usuario no tenga que repetir siempre el ejercicio con los mismos datos.
- Un fichero XML denominado “plan” que indica qué datos se deben de tomar del fichero “series” según las veces que se haya repetido el ejercicio o el número de sesión.
- Todos los ejercicios se referencian en un fichero JavaScript que declara una única instancia de una clase, la cual devuelve una nueva instancia de un ejercicio asociado a un identificador.

En la [figura 34](#) se puede ver un diagrama de la implementación de los ejercicios en la versión original.

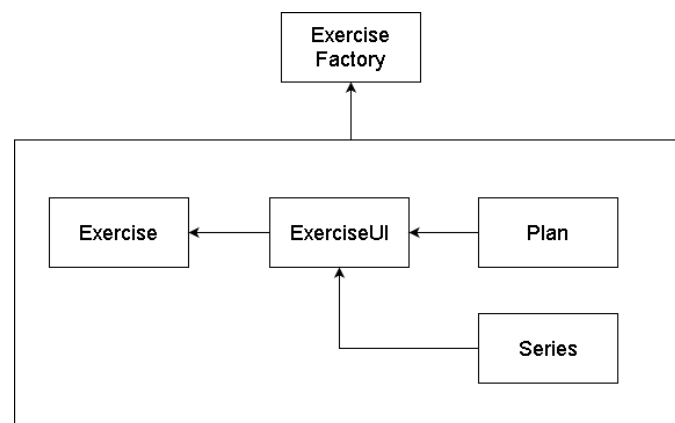


Figura 34. Diagrama de la implementación de los ejercicios en la versión original

En la nueva implementación, los ejercicios se han desarrollado utilizando componentes y servicios Angular siguiendo el siguiente esquema:

- Un componente Angular formado por tres partes principales:
  - Un fichero HTML que contiene la interfaz gráfica del ejercicio.

Para una mayor simplicidad de la interfaz y código, se ha subdividido el ejercicio en fases, mostrando únicamente la fase actual del ejercicio en cada momento. Para ello se ha utilizado la directiva “*\*ngIf*” la cual permite ocultar o mostrar un bloque de código HTML según una expresión booleana.

- Un fichero CSS para modificar el estilo de los elementos que se encuentran en el fichero HTML.
- Un fichero TypeScript. Este fichero es la parte lógica del ejercicio y contiene todas las funciones que lo controlan. Desde aquí se modifica la fase actual del ejercicio, se notifica cuándo ha acabado o se envía la puntuación obtenida.
- Servicios Angular. Permiten las conexiones con el backend con el fin de obtener información necesaria para el ejercicio y también almacenan algunos datos relativos a los ejercicios. Esta información incluye rutas a imágenes o cálculo de resultados. Al estar separados del frontend, dejan una aplicación mejor estructurada.
- Al igual que en la versión original, todos los ejercicios se referencian en un fichero, siendo esta vez un servicio Angular para una mejor organización. Cada ejercicio cuenta con un identificador y la clase asociada al componente.

En el siguiente diagrama (ver [figura 35](#)) se puede apreciar la estructura de la implementación de los ejercicios en la nueva versión.

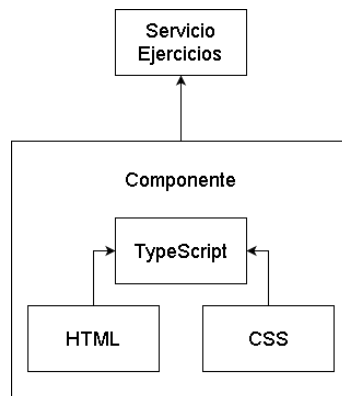


Figura 35. Diagrama de la implementación de los ejercicios en la nueva versión

Los ejercicios se agrupan en sesiones con el fin de organizarlos para que el usuario pueda realizar varios seguidos en un determinado espacio de tiempo. Originalmente se planearon un total de quince sesiones para completar el curso de la plataforma. Cada sesión cuenta con entre tres y nueve ejercicios, seleccionados según su duración y temática, de forma que se complementen.

En la implementación original, las sesiones se encuentran definidas en un fichero XML, el cual el sistema consulta cada vez que necesite cargar una nueva sesión o consultar cuál es el siguiente ejercicio en la misma.

En la nueva versión, al igual que para los ejercicios, las sesiones también se referencian en un servicio Angular, tienen un identificador y la lista de ejercicios que contiene cada una de las ellas, desapareciendo el archivo XML de la versión original y siendo sustituido por el servicio Angular.

Las conexiones e intercambio de información de estos elementos se detalla en la sección [Sistemas de comunicación](#).



### 3.3.4.1. Página Session y componente Assistant

La página Session tiene una de las misiones principales de este proyecto, cargar los componentes de los ejercicios y las sesiones dinámicamente. Por esto se ha creado una sección exclusiva para esta página donde se detallará su funcionamiento.

La carga de ejercicios se realiza cargando en la página Session componentes Angular que incluyen los ejercicios de forma dinámica. Para ello hace uso de la directiva “*anchor*” y la etiqueta “*ng-template*” [27]. La directiva *anchor* nos permite indicar a Angular en qué lugar queremos insertar los componentes. Por otro lado, *ng-template*, como su nombre indica, actúa de plantilla, permitiendo crear plantillas en la interfaz de usuario, que pueden ser referenciadas en el momento que sea necesario mostrar su contenido. Al combinar *ng-template* con la directiva *anchor*, creamos una plantilla en la cual Angular sabe que puede insertar componentes y que podemos mostrar cuando sea necesario.

Una vez que Angular sabe dónde queremos insertar los componentes, es el momento de crearlos dinámicamente, para ello, utilizamos la clase “*ComponentFactoryResolver*”. Esta clase permite la creación de instancias de componentes, lo cual nos permite instanciar el componente asociado al ejercicio que sea necesario en cada momento. Juntando estos tres elementos podemos cargar ejercicios dinámicamente en una sola página (ver [figura 36](#)).

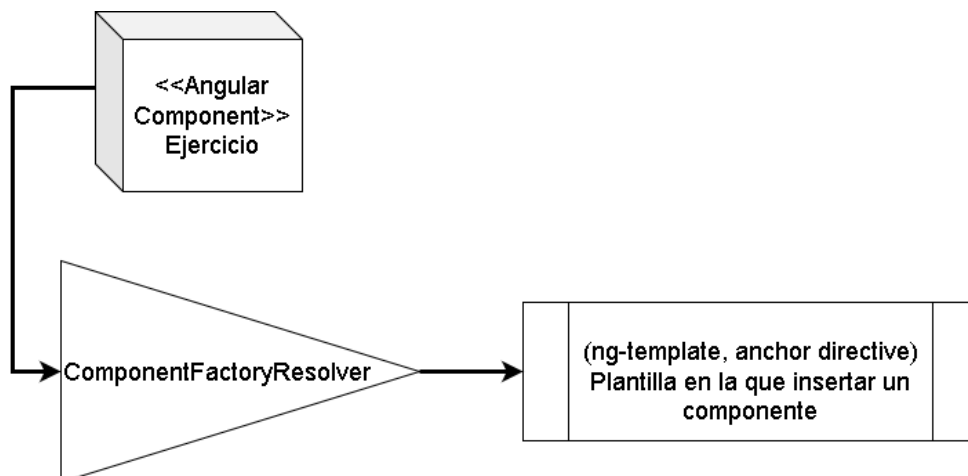


Figura 36. Diagrama del funcionamiento de la página Session

Relacionado a la generación dinámica de ejercicios, se ha creado el componente “assistant” (ver [figura 37](#)). Este componente contiene una imagen del asistente, un título y una descripción. Está presente en todos los ejercicios y sesiones para explicar lo que sucede o se muestra en pantalla. Se ha desarrollado en un componente Angular para facilitar su mantenimiento en un futuro, donde se requiera añadir o cambiar su aspecto o funcionalidades. Este componente está incrustado en la página Session, no en los ejercicios.



### **¡Prueba de Puntería!**

Antes de comenzar con la explicación y la realización de ejercicios vamos a comprobar tu puntería. Para ello, tendrás que pulsar con el ratón en los cuadrados azules que aparezcan por la pantalla. Pulsa en el botón Continuar cuando estés listo para comenzar.

**Figura 37. Assistant component**

### 3.3.5. Sistemas de comunicación

La aplicación cuenta con dos sistemas principales para el paso de información y comunicaciones, uno para las comunicaciones dentro del propio frontend y otro para las comunicaciones del frontend con el backend. La información que se intercambia se detalla en las siguientes subsecciones.

#### 3.3.5.1. Comunicaciones frontend-frontend

Se ha desarrollado un sistema de comunicación mediante eventos para el paso de información en el frontend. Este sistema permite a la página Session, el componente Assistant (ver [Página Session y componente Assistant](#)) y los ejercicios encapsulados en componentes Angular enviar y recibir información sin necesidad de conocerse entre sí.

Para ello, se ha creado la clase “*ExerciseManager*” la cual define los emisores de eventos y las funciones que desempeñan. La clase es instanciada y usable por todos los elementos del frontend que la necesiten. Esta clase tiene tres objetivos de comunicación que se detallan a continuación. Normalmente este intercambio de información se realiza en el orden que se muestra.

- Notificar el identificador de usuario y los parámetros de inicio a los ejercicios que se inserten dinámicamente en la página Session. Estos datos permiten al ejercicio conocer su identificador y el del usuario que está lo está realizando para poder hacer las peticiones correspondientes al backend mediante servicios Angular. Por otro lado, los parámetros de inicio permiten al ejercicio saber cuántas veces se tiene que repetir o cuál es el límite de tiempo para completarlo. Todos los ejercicios deben estar suscritos a este evento con el fin de poder recibir estos datos.
- Notificar un cambio al componente Assistant. Dado que el componente Assistant está presente durante la realización de todos los ejercicios con el fin de apoyar y ofrecer información sobre lo que sucede, el ejercicio debe ser capaz de modificar el estado (mostrar y ocultar) y contenido del asistente (título y descripción) en función de la situación actual. El componente Assistant está suscrito a este evento, modificando su estado según las peticiones de los ejercicios.
- Notificar a la página Session que el ejercicio ha finalizado. Una vez que un ejercicio ha finalizado, se lanza este evento al que está suscrito la página Session, para que elimine el ejercicio actual y pase al siguiente. En el caso de no haber más ejercicios, se finaliza la sesión.

A continuación se muestra un diagrama con el funcionamiento de este sistema de comunicación por eventos (ver [figura 38](#)):

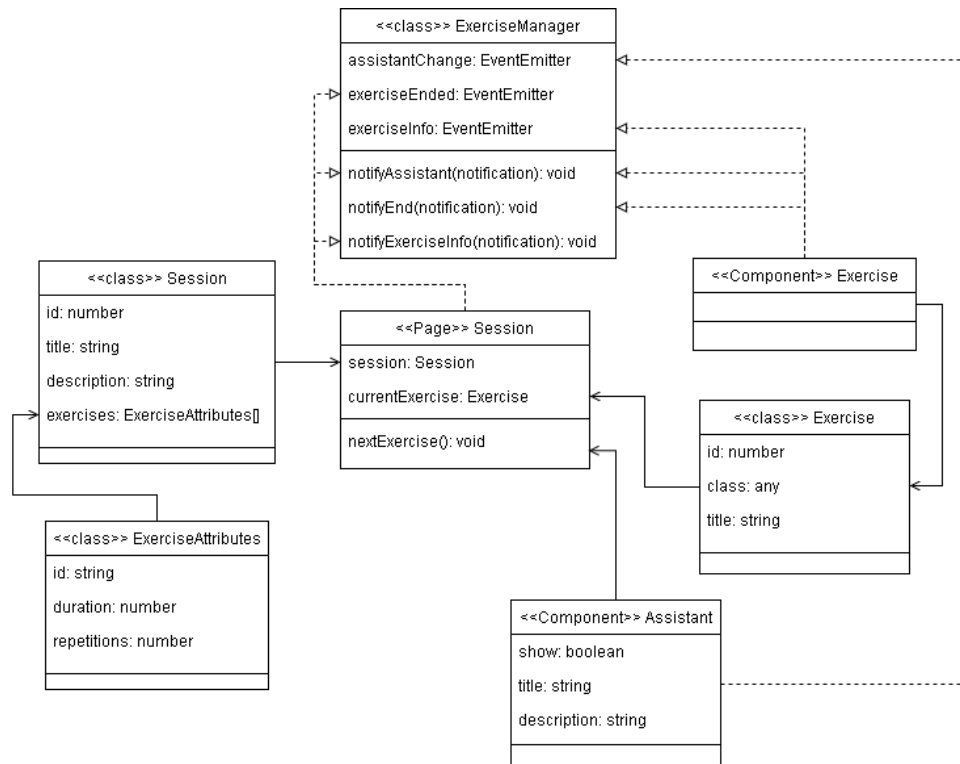


Figura 38. Diagrama del sistema de comunicación por eventos

En el centro del diagrama se puede observar la página Session. Esta cuenta con distintos atributos, entre ellos, un atributo Session que hace referencia a la sesión actual que realiza el usuario (<<class>> Session). Esta sesión tiene un identificador, un título y descripción para que el componente Assistant los muestre y una lista con los ejercicios que forman la sesión (<<class>> ExerciseAttributes). También cuenta con un atributo Exercise, el cual almacena un objeto de este tipo. Es esencial que la página Session sepa cual es el ejercicio actual (<<class>> Exercise), ya que referencia la clase del componente Angular que contiene el ejercicio y que usará para cargarlo dinámicamente. En la parte inferior del diagrama vemos el componente Assistant incrustado en la página Session. Finalmente, en la zona superior está la clase ExerciseManager. Con líneas discontinuas podemos apreciar los eventos a los que está suscrita cada clase, así como los métodos que utiliza.

### 3.3.5.2. Comunicaciones frontend-backend

Para el proyecto se ha desarrollado un backend utilizando Express.js, un framework para Node.js. En él se han implementado una serie de funcionalidades basadas en métodos, las cuales usan endpoints para el envío y recogida de información. Esta información está definida según unos modelos de datos definidos también en el backend. En cuanto a la base de datos, se ha desarrollado utilizando MongoDB, en vez de MySQL como en la versión original. MongoDB es un sistema de base de datos NoSQL, es decir, los datos no se guardan en tablas como en las bases de datos relacionales, sino en estructuras de datos de tipo "BSON", similar a "JSON". Aunque provisionalmente los usuarios se almacenan en esta base de datos NoSQL, si fuese necesario se podría crear un nuevo microservicio que tratase la información relativa a estos en una base de datos relacional, como en la versión original, aunque como esto no estaba previsto en el desarrollo de este proyecto, se ha dejado para una etapa de desarrollo futura (ver

[Trabajos futuros](#)). La base de datos contiene información relativa a:

- Información sobre los usuarios que utilizan el sistema, concretamente su identificador, la sesión y el ejercicio por los que van. Con esta información, el sistema puede volver al ejercicio por el que se quedó el usuario en el caso de tener que dejar una sesión a medias. Como se ha mencionado previamente, los usuarios creados tienen información muy básica, pero suficiente para llevar un control sobre los ejercicios y sesiones que han completado. Cualquier otra información adicional de la versión original deberá ser implementada en una futura versión, ya que en esta no estaba contemplada.
- Información sobre los ejercicios: imágenes necesarias para los ejercicios o datos para su realización. Esto incluye rutas a imágenes.
- Recogida de información o puntuaciones. La base de datos también almacena la puntuación que ha obtenido cada usuario en cada ejercicio, así como las respuestas que da el usuario a los distintos cuestionarios que la plataforma ofrece.
- Cálculo y almacenaje de medallas. Gracias a las puntuaciones obtenidas del punto anterior, el backend calcula una medalla asociada a la puntuación obtenida y la almacena.

Las comunicaciones entre frontend y backend (ver [figura 39](#)) se realizan mediante servicios Angular. En total se han implementado dieciocho servicios. Ocho de ellos obtienen información relativa a su correspondiente ejercicio como se ha detallado previamente. Los servicios Exercises y Sessions almacenan las listas de ejercicios y sesiones y las comunican con la aplicación. ExercisesResults y Medals, conectan con el backend para almacenar y calcular resultados. Questionnaire e InstrumentalQuestionnaire mandan al backend las respuestas de los cuestionarios. El servicio User obtiene y actualiza información sobre los usuarios del sistema, por ejemplo, cada vez que completan un ejercicio. Por último, servicios como Functions, Pages o HeaderOptions guardan información o funcionalidades específicas para el sistema.

Estos servicios implementan funciones las cuales realizan peticiones HTTP al backend para el envío o recogida de datos. Finalmente los servicios se inyectan en los componentes o páginas que los requieran. A continuación se muestra un diagrama:

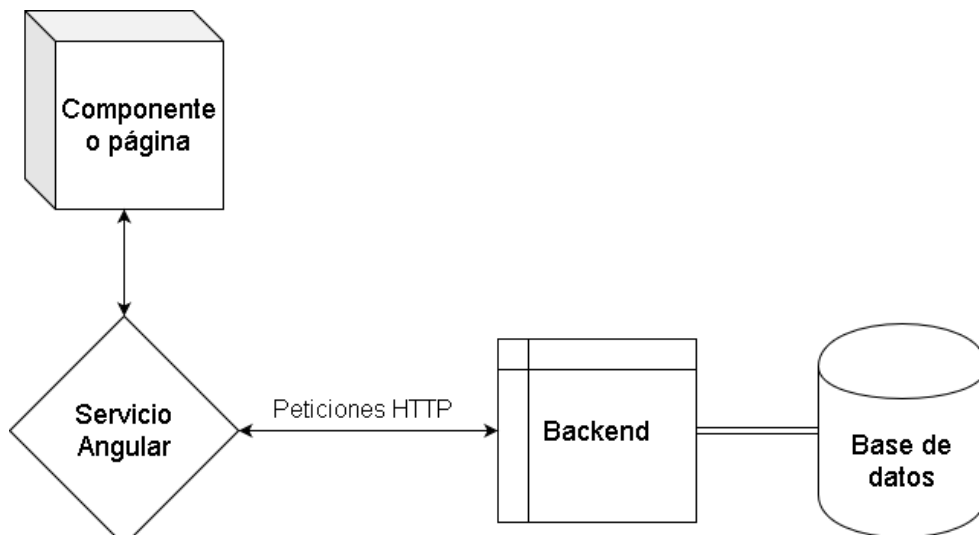


Figura 39. Diagrama de comunicación entre frontend y backend

### 3.3.5.2.1. Lista de endpoints y modelos de datos

Lista de endpoints del backend:

- **/api/users**
  - Modelo de datos:
    - id: Number
    - session: Number
    - currentExercise: Number
  - **POST** / Añade un nuevo usuario
  - **GET** /:id Devuelve un usuario por su identificador
  - **POST** /:id Actualiza el currentExercise o session de un usuario
- **/api/exerciseResults**
  - Modelo de datos:
    - id: Number
    - userId: Number
    - exerciseId: Number
    - correctCount: Number
    - failCount: Number
    - omissionCount: Number
    - finalScore: Number
    - seconds: Number
  - **POST** / Añade un nuevo resultado
- **/api/medals**
  - Modelo de datos:
    - id: Number
    - userId: Number
    - exerciseId: Number
    - medal: String
  - **GET** /:id Devuelve la medalla del usuario especificado
  - **POST** /:finalScore Crea y añade una medalla para el usuario y ejercicio especificado
- **/api/questionnaire**

- Modelo de datos:
  - userId: Number
  - gender: String
  - placeOfBirth: String
  - yearOfBirth: Number
  - monthOfBirth: Number
  - dayOfBirth: Number
  - maritalStatus: String
  - liveWith: String
  - bathing: String
  - getDressed: String
  - getReady: String
  - eating: String
  - urinating: String
  - defecating: String
  - toilet: String
  - bedSofa: String
  - walking: String
  - stairs: String
  - education: String
  - read: String
  - workshop: String
  - physicalExercise: String
  - computer: String
  - phoneNumber: String
- **POST** / Añade un nuevo resultado para el cuestionario o lo actualiza
- **/api/instrumentalQuestionnaire**
  - Modelo de datos:
    - userId: Number
    - telephone: String
    - shopping: String
    - cooking: String
    - householdChores: String
    - laundry: String
    - transport: String
    - medicine: String
    - economicAffairs: String
  - **POST** / Añade un nuevo resultado para el cuestionario o lo actualiza
- **/api/classifyObjects**
  - Modelo de datos:
    - id: String
    - img: Mixed
  - **GET** /:id Devuelve la información de una imagen asociada al identificador
  - **POST** / Añade una nueva imagen
- **/api/directNumbers**
  - Modelo de datos:
    - id: String
    - img: Mixed
  - **GET** / Devuelve todas las imágenes
  - **POST** / Añade una nueva imagen

- **/api/numbersAndVowels**
  - Modelo de datos:
    - id: String
    - img: Mixed
  - **GET** / Devuelve todas las imágenes
  - **POST** / Añade una nueva imagen
- **/api/positions**
  - Modelo de datos:
    - id: String
    - img: Mixed
  - **GET** / Devuelve todas las imágenes
  - **POST** / Añade una nueva imagen
- **/api/pyramids**
  - Modelo de datos:
    - id: String
    - img: Mixed
  - **GET** / Devuelve todas las imágenes
  - **POST** / Añade una nueva imagen
- **/api/logicalSeries**
  - Modelo de datos:
    - id: String
    - img: Mixed
  - **GET** / Devuelve todas las imágenes
  - **POST** / Añade una nueva imagen



## 3.4. Fase 2

Durante esta fase del proyecto se hizo el desarrollo de algunos de los ejercicios de la plataforma. El implementar una serie de ejercicios se hizo con el objetivo de dejar una base para que el resto de ejercicios puedan ser implementados en un futuro. Todos los ejercicios han sido desarrollados basados en los originales, pero se han introducido cambios visuales (que varían en función de cada ejercicio) y funcionales (con el fin de mejorar la estructura o funcionamiento de los mismos). Cabe destacar que, como se ha mencionado en secciones previas, todos los ejercicios se han desarrollado en componentes Angular, de modo que se puedan cargar dinámicamente sobre la página Session.

Por otro lado, para una mejor estructuración y legibilidad del código HTML de los ejercicios, estos se han dividido en **fases** las cuales equivalen a **pasos** o **etapas**, esto es, a cada vista diferente del ejercicio se le ha asignado una **etapa**. Gracias a la directiva de Angular “*\*ngIf*”, podemos mostrar y ocultar código HTML mediante expresiones booleanas. Combinando esta directiva y la estructuración de los ejercicios en fases, es posible estructurar el código de forma limpia y legible, de forma que sea sencillo su mantenimiento en el caso de necesitar ser modificado en un futuro. En la sección de cada ejercicio se explicarán con más detalle sus fases.

Para mostrar los datos en pantalla, la mayor parte de los ejercicios utilizan la directiva de Angular “*\*ngFor*”. Esta directiva permite iterar bloques de código HTML para cada elemento en una colección. De esta forma, no sería necesario modificar la implementación de la interfaz en el caso de necesitar introducir más listas de palabras o modificar el tamaño de las mismas. El uso de esta directiva permite iterar sobre todo tipo de colecciones, de modo que se pueden, por ejemplo, crear listas de imágenes, botones, palabras, o conjuntos de estos a partir de una lista de datos.

Los resultados que se obtienen en los ejercicios se van introduciendo en un atributo de tipo “Score”, el cual almacena el número de aciertos, fallos, preguntas sin responder, tiempo usado y calificación final. Para el almacenaje de resultados y cálculo de medallas, los ejercicios inyectan los servicios “*ExerciseResultsService*” y “*MedalsService*” que conectan directamente con el backend. En el caso de los cuestionarios, lógicamente, no se almacenan resultados ni calculan medallas, pero si se almacenan las respuestas del usuario. Para ello, cada cuestionario inyecta un servicio específico para el almacenaje de estos datos, “*QuestionnaireService*” o “*InstrumentalQuestionnaireService*”.

En la versión original había un total de veinticinco ejercicios, en esta nueva versión, se han implementado un total de quince ejercicio, basados en los originales. Sin embargo los ejercicios *IntroductionExercise*, *QuestionnaireExercise* y *FirstHelpExercise* se han agrupado en un solo ejercicio y los ejercicios *SecondHelpExercise* y *InstrumentalQuestionnaireExercise* se han agrupado en otro. Los motivos de este cambio se pueden encontrar en sus respectivas subsecciones [IntroductionExercise](#) e [InstrumentalQuestionnaireExercise](#). Este cambio hace un total de doce componentes Angular basados en ejercicios.

Cada ejercicio ha tenido que ser desarrollado de forma individual, ya que son muy diferentes entre sí. Para ello se ha tenido que desarrollar una interfaz visual única para ejercicio utilizando los métodos explicados previamente. Lo mismo ocurre para la lógica del ejercicio, la cual se ha desarrollado según el ejercicio, teniendo en cuenta la finalidad del mismo y la interacción del usuario con este. Por ejemplo, el ejercicio *NumbersAndVowelsExercise*, requiere de un panel de números y otro de vocales, cada uno con sus botones correspondientes, una serie de relojes que

determinen el tiempo en el que aparece en pantalla cada símbolo y un servicio Angular con el que comprobar si las respuestas proporcionadas son correctas. Por otro lado, un ejercicio como PyramidsExercise requiere una interfaz basada en múltiples imágenes que pueden ser seleccionadas, sistemas de cuenta atrás, comprobación de elementos seleccionados en pantalla y un servicio Angular que permita comprobar las respuestas.

A continuación se listan los ejercicios, su división en fases y su funcionamiento.

### 3.4.1. WordListExercise

WordListExercise es un ejercicio que estimula la memoria del usuario. En este ejercicio se debe memorizar una lista de palabras durante un periodo de tiempo, una vez finalizado, el usuario deberá escribir las palabras que recuerde en una entrada de texto. El ejercicio se repetirá un número fijo de veces, dando la posibilidad al usuario de recordar más palabras de la lista.

#### 3.4.1.1. Fases

1. INTRO: durante esta fase se describe al usuario el dominio del ejercicio mediante un mensaje del asistente (ver [figura 40](#)).



Figura 40. Fase INTRO WordListExercise

2. COUNTDOWN: se muestra una cuenta atrás para el comienzo del ejercicio, esta fase es recurrente en gran parte de los ejercicios desarrollados en este proyecto (ver [figura 41](#)).

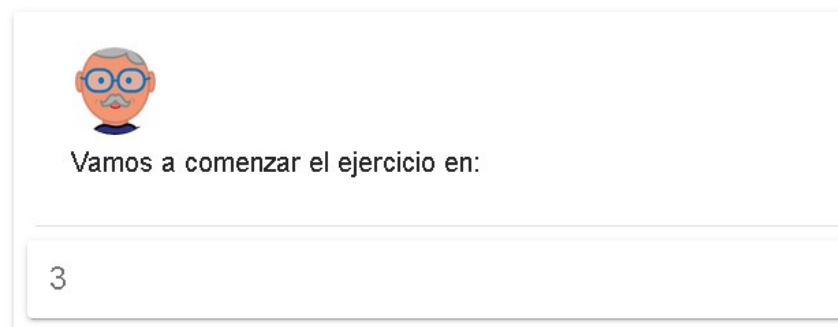


Figura 41. Fase COUNTDOWN

3. READ: durante esta fase del ejercicio se muestra por pantalla la lista de palabras que el usuario debe memorizar (ver [figura 42](#)). La lista permanecerá en pantalla durante un tiempo fijo especificado en la parte inferior.

- Buitre

- Clavel

- Licor

- Silla

- Orquídea

- Águila

- Lámpara

- Anís

- Pavo

- Armario


- Jazmín

- Cofiac

Tiempo restante: 43/60

Figura 42. Fase READ WordListExercise

4. **WRITE:** una vez acabado el tiempo se mostrará una entrada de texto para añadir las palabras que se recuerden y un botón para continuar a la siguiente fase una vez que se hayan escrito las palabras (ver [figura 43](#)). Las palabras se pueden introducir sin mayúsculas y sin tildes con el fin de facilitar la entrada de datos a todo tipo de personas.

 **Escribe las palabras**  
 A continuación, escribe una a una las palabras que recuerdes sin importar el orden. Para ello, introduce cada palabra en el recuadro de abajo y pulsa 'Escribir Siguiente Palabra'. Cuando hayas terminado pulsa 'No recuerdo más palabras'.

Introduce una palabra:  
 silla

ESCRIBIR SIGUIENTE PALABRA NO RECUERDO MÁS PALABRAS

Figura 43. Fase WRITE WordListExercise

5. **REPEAT:** esta fase permite al ejercicio detectar si se han completado el número de repeticiones asignadas o si es necesario repetir la lista de palabras mostrada de nuevo (ver [figura 44](#)).

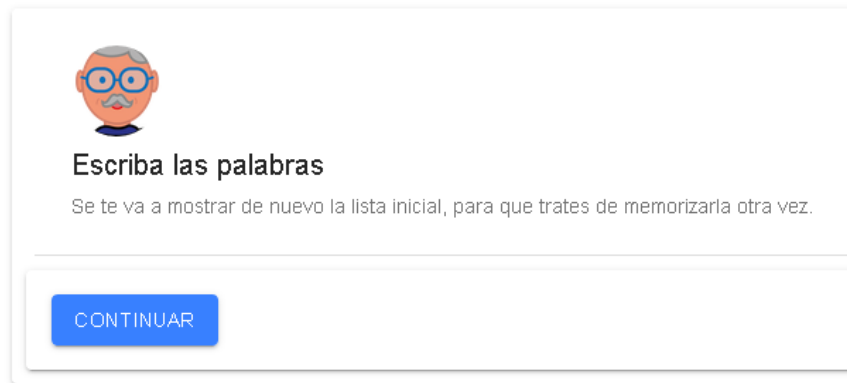


Figura 44. Fase REPEAT WordListExercise

6. END: fase de finalización del ejercicio, en esta fase se guarda la puntuación obtenida en el ejercicio y se manda al backend para su almacenaje y cálculo de la medalla para este ejercicio que se mostrará en pantalla (ver [figura 45](#)). Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.

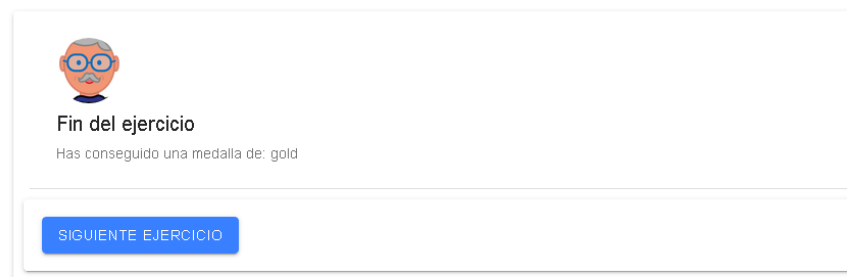


Figura 45. Fase END

### 3.4.1.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitemos que sean mostrados en pantalla.

Para mostrar la lista de palabras de la fase READ se ha utilizado la directiva `*ngFor` de Angular. Esta directiva permite iterar sobre una colección de elementos repitiendo el código HTML que contenga la etiqueta donde se incluyó la directiva. Así, no es necesaria la repetición constante de bloques de código que, en este caso, serían la lista de palabras que se muestra y que el usuario debe memorizar.

El código final obtiene una serie de valores positivos respecto a una implementación en la que se repitiese el código. En primer lugar, el código queda mucho más limpio y legible, algo fundamental y que aporta ventajas para el mantenimiento de la aplicación en un futuro, dejando un código breve y fácil de entender. Además, en el caso de que se necesite añadir más listas de palabras, se podría hacer sin necesidad de modificar la vista pues este código muestra por pantalla cualquier lista de palabras que se le pase.

### 3.4.1.3. Lógica del ejercicio

Para la cuenta atrás que se muestra en la fase COUNTDOWN y el tiempo límite de memorización de la lista mostrado en la fase READ, se han implementado dos temporizadores. Para ello se ha utilizado el método “*setInterval*” de Angular, el cual permite iterar una acción cada vez que haya pasado un tiempo determinado. De este modo podemos controlar los segundos que pasan desde que el usuario comienza el ejercicio o empieza la cuenta atrás.

Con el objetivo de que el ejercicio sea lo más sencillo posible de realizar, se pensó que sería ideal que el usuario pudiese introducir las palabras de la forma más cómoda para este. Esto quiere decir que el usuario pueda introducir las palabras en minúsculas, mayúsculas, con mayúsculas y minúsculas, con o sin tildes. Para ello se ha implementado una función a la que se ha llamado “*lowerCaseRemoveAccents*” la cual da formato a una cadena de texto introducida, cambiando todas las mayúsculas por minúsculas e intercambiando todas las letras con acentos por su versión sin este. También se ha implementado este mecanismo para símbolos que no son del español, como acentos circunflejos o diéresis en las vocales, dando soporte a lenguas extranjeras. Todas las respuestas que aporte el usuario se pasarán por esta función para su comprobación con la solución.

Para la comprobación de palabras, se comprueba si son correctas comparándolas con el array de palabras correctas. Si lo son, se comprueba si la palabra ya ha sido añadida al array de palabras correctas respondidas para evitar repeticiones. Acertar o fallar palabras, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

### 3.4.1.4. Comunicaciones

El ejercicio toma los datos del servicio Angular “*WordListService*”.

Cuando el componente se inicializa, se suscribe al evento “*exerciseInfo*” mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento “*assistantChange*” al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, se realiza una petición HTTP de tipo POST al backend utilizando el servicio Angular denominado “*MedalsService*”. Esta petición envía la puntuación final obtenida en el ejercicio, la cual la recoge un endpoint implementado en el backend. Dicho endpoint tiene asociada una función que calcula la medalla asociada a la puntuación recibida y la almacena en la base de datos. Una vez calculada, el ejercicio hace una petición de tipo GET a otro endpoint con el fin de obtener la medalla calculada y poder mostrarla en pantalla. A continuación, utilizando el servicio Angular “*ExerciseResultsService*” se envía a otro endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento “*exerciseEnded*” al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

## 3.4.2. LogicalSeriesExercise

En este ejercicio el usuario verá series de imágenes y deberá intentar adivinar cuál es el patrón que siguen con el fin de encontrar la siguiente imagen de la serie. Una vez escogida una solución, la aplicación mostrará con colores cuál era la respuesta correcta. LogicalSeriesExercise pretende estimular el razonamiento lógico del usuario.

### 3.4.2.1. Fases

1. INTRO: muestra una explicación sobre la temática del ejercicio y un ejemplo para que el usuario entienda lo que propone el ejercicio (ver [Figura 46](#)).

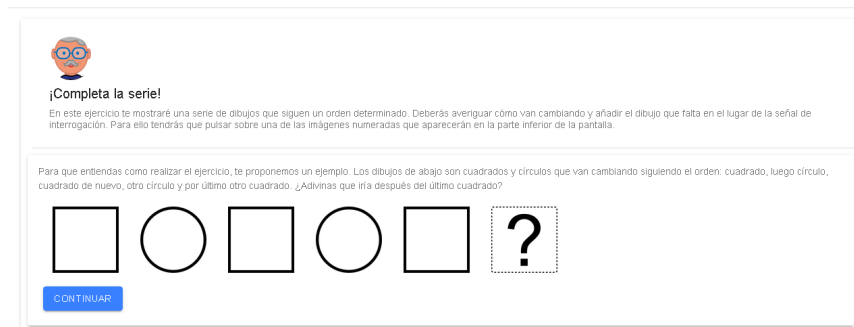


Figura 46. Fase INTRO LogicalSeriesExercise

2. DEMO: en esta fase se permite al usuario escoger una respuesta a la serie mostrada durante la fase INTRO. Hasta que el usuario no haya seleccionado una respuesta no se le permitirá avanzar en el ejercicio (ver [figura 47](#)).

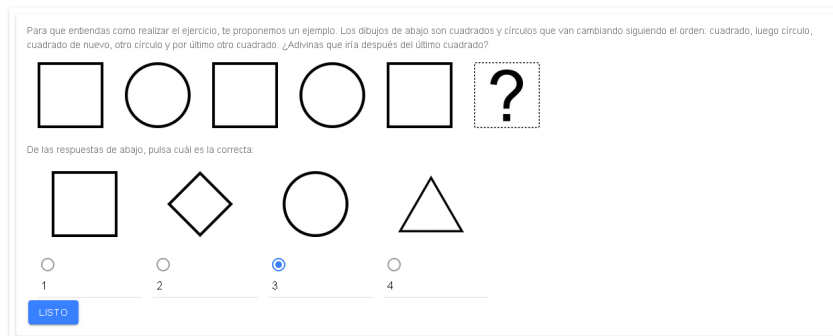


Figura 47. Fase DEMO LogicalSeriesExercise

3. COUNTDOWN: se muestra una cuenta atrás para el comienzo del ejercicio, esta fase es recurrente en gran parte de los ejercicios desarrollados en este proyecto (ver [figura 41](#)).
4. EXERCISE: durante esta fase se irán mostrando las distintas series del ejercicio hasta que no haya más series que mostrar. Una vez se haya seleccionado y confirmado una respuesta, se mostrará con colores cuál era la respuesta correcta y se completará la serie con la respuesta correcta (ver [figura 48](#)).



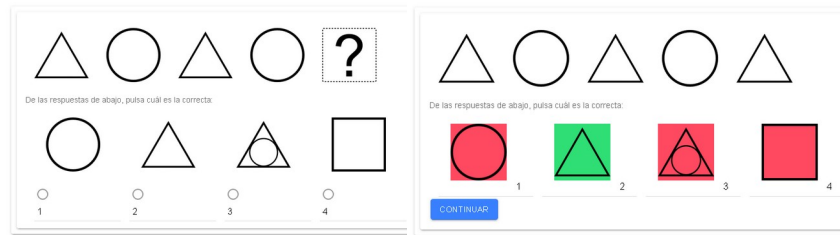


Figura 48. Fase EXERCISE LogicalSeriesExercise

5. END: fase de finalización del ejercicio, en esta fase se guarda la puntuación obtenida en el ejercicio y se manda al backend para su almacenaje y cálculo de la medalla para este ejercicio que se mostrará en pantalla (ver [figura 45](#)). Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.

### 3.4.2.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitemos que sean mostrados en pantalla.

Se han utilizado botones de tipo radio proporcionados por Ionic para dotar al ejercicio de un aspecto más moderno e intuitivo. Los botones radio solo permiten que uno de ellos esté seleccionado cada vez, siendo ideales para este tipo de ejercicios.

Para mostrar las figuras y los botones radio se ha utilizado la directiva `*ngFor` de Angular. Esta directiva permite iterar sobre una colección de elementos repitiendo el código HTML que contenga la etiqueta donde se incluyó la directiva. Así, no es necesaria la repetición constante de bloques de código. Se han implementado dos series utilizando la directiva `*ngFor`, una para la lista de imágenes que se muestra como ejemplo y otra para la lista de respuestas. Mientras que para la primera solo se itera sobre imágenes, para la segunda se itera sobre bloques de imágenes, botones radio e identificadores.

El código final obtiene una serie de valores positivos respecto a una implementación en la que se repitiese el código. En primer lugar, el código queda mucho más limpio y legible, algo fundamental y que aporta ventajas para el mantenimiento de la aplicación en un futuro, dejando un código breve y fácil de entender. Además, en el caso de que se necesite añadir más listas de imágenes, se podría hacer sin necesidad de modificar la vista pues este código muestra por pantalla cualquier lista de imágenes que se le pase.

### 3.4.2.3. Lógica del ejercicio

Para la cuenta atrás que se muestra en la fase COUNTDOWN, se ha implementado un temporizador. Para ello se ha utilizado el método `setInterval` de Angular, el cual permite iterar una acción cada vez que haya pasado un tiempo determinado. De este modo podemos controlar los segundos que pasan desde que empieza la cuenta atrás.

Para evitar preguntas sin responder y hacer más sencillo de entender lo que

sucede en pantalla, se ha limitado el poder pasar a la siguiente serie hasta que no se seleccione una opción de la lista de respuestas.

También se lleva un control sobre la imagen con una interrogación que hace de placeholder antes de responder. Una vez que se ha respondido, la imagen cambia con la solución y se añade un color rojo y verde a las respuestas para indicar cuál era la respuesta correcta.

Dado que los botones se generan de forma iterativa con la directiva `*ngFor` tal y como se explicó en la sección anterior, podría no ser trivial el hecho de saber cuál es la opción seleccionada de la lista de respuestas, para ello se hace uso de la etiqueta `ion-radio-group` ofrecida por Ionic y la directiva de Angular `[(ngModel)]`. La etiqueta `ion-radio-group` permite agrupar una serie de botones radio (ver sección anterior). De este modo Ionic sabe cuántos botones generamos con la directiva `*ngFor`. Por otro lado, la directiva `[(ngModel)]` permite asociar el valor que toma `ion-radio-group` a una variable. Así, podemos saber la opción seleccionada o modificar su contenido desde el TypeScript.

Para la verificar si la opción seleccionada es correcta, se hace una comparación del elemento seleccionado (que está asociado a una variable como se ha explicado previamente) con la opción correcta proporcionada por el servicio Angular `LogicalSeriesService`. Acertar o fallar respuestas, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

#### 3.4.2.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento `exerciseInfo` mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

También durante la inicialización del componente, este hace una petición GET mediante el servicio Angular `LogicalSeriesService` al backend, para que este le envíe todas las rutas y atributos de las imágenes que se van a utilizar durante el ejercicio (ver [Comunicaciones frontend-backend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento `assistantChange` al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, se realiza una petición HTTP de tipo POST al backend utilizando el servicio Angular denominado `MedalsService`. Esta petición envía la puntuación final obtenida en el ejercicio, la cual la recoge un endpoint implementado en el backend. Dicho endpoint tiene asociada una función que calcula la medalla asociada a la puntuación recibida y la almacena en la base de datos. Una vez calculada, el ejercicio hace una petición de tipo GET a otro endpoint con el fin de obtener la medalla calculada y poder mostrarla en pantalla. A continuación, utilizando el servicio Angular `ExerciseResultsService` se envía a otro endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento `exerciseEnded`

al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

### 3.4.3. SemanticSeriesExercise

SemanticSeries pretende estimular el razonamiento lógico del usuario mostrándole series de palabras. Las palabras mostradas en cada serie están relacionadas semánticamente de algún modo entre sí exceptuando una de ellas. Con este ejercicio se pretende que el usuario sea capaz de analizar las relaciones y averiguar qué palabra no guarda una relación con las demás.

#### 3.4.3.1. Fases

1. INTRO: durante esta fase inicial se muestra al usuario una explicación sobre el ejercicio (ver [figura 49](#)) y un ejemplo sobre el tipo de series que va a encontrar y el tipo de relaciones que pueden tener.

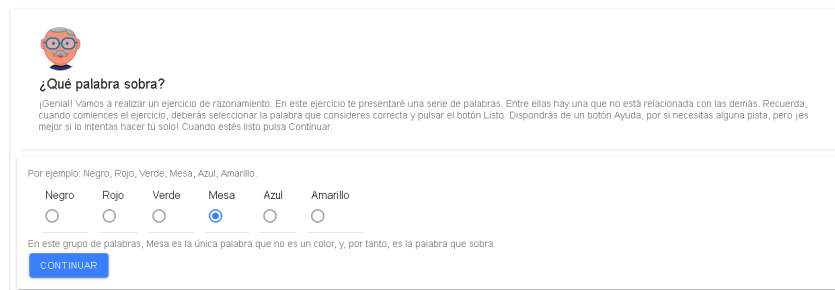


Figura 49. Fase INTRO SemanticSeriesExercise

2. COUNTDOWN: se muestra una cuenta atrás para el comienzo del ejercicio, esta fase es recurrente en gran parte de los ejercicios desarrollados en este proyecto (ver [figura 41](#)).
3. EXERCISE: durante esta fase se irán mostrando las distintas series del ejercicio hasta que no haya más series que mostrar. Una vez se haya seleccionado y confirmado una respuesta, se mostrará con colores cuál era la respuesta correcta (ver [figura 50](#)).

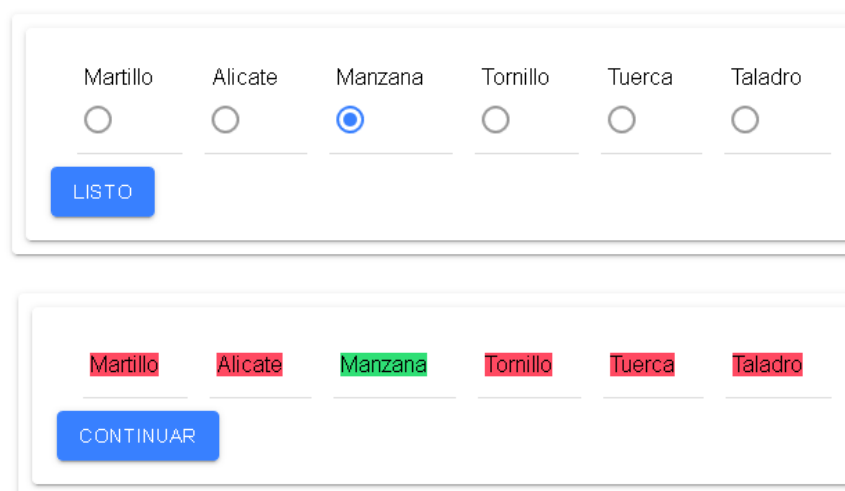


Figura 50. Fase EXERCISE SemanticSeriesExercise

4. END: fase de finalización del ejercicio, en esta fase se guarda la puntuación obtenida en el ejercicio y se manda al backend para su

almacenaje y cálculo de la medalla para este ejercicio que se mostrará en pantalla (ver [figura 45](#)). Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.

### 3.4.3.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitemos que sean mostrados en pantalla.

Se han utilizado botones de tipo radio proporcionados por Ionic para dotar al ejercicio de un aspecto más moderno e intuitivo. Los botones radio solo permiten que uno de ellos esté seleccionado cada vez, siendo ideales para este tipo de ejercicios.

Para mostrar las palabras y los botones radio se ha utilizado la directiva `*ngFor` de Angular. Esta directiva permite iterar sobre una colección de elementos repitiendo el código HTML que contenga la etiqueta donde se incluyó la directiva. Así, no es necesaria la repetición constante de bloques de código. La directiva itera sobre las palabras y los botones radio.

El código final obtiene una serie de valores positivos respecto a una implementación en la que se repitiese el código. En primer lugar, el código queda mucho más limpio y legible, algo fundamental y que aporta ventajas para el mantenimiento de la aplicación en un futuro, dejando un código breve y fácil de entender. Además, en el caso de que se necesite añadir más listas de palabras, se podría hacer sin necesidad de modificar la vista pues este código muestra por pantalla cualquier lista de palabras que se le pase.

### 3.4.3.3. Lógica del ejercicio

Para la cuenta atrás que se muestra en la fase COUNTDOWN, se ha implementado un temporizador. Para ello se ha utilizado el método `setInterval` de Angular, el cual permite iterar una acción cada vez que haya pasado un tiempo determinado. De este modo podemos controlar los segundos que pasan desde que empieza la cuenta atrás.

Para evitar preguntas sin responder y hacer más sencillo de entender lo que sucede en pantalla, se ha limitado el poder pasar a la siguiente serie hasta que no se seleccione una opción de la lista de respuestas.

Una vez que se ha respondido se añade un color rojo y verde a las palabras para indicar cuál era la respuesta correcta.

Dado que los botones se generan de forma iterativa con la directiva `*ngFor` tal y como se explicó en la sección anterior, podría no ser trivial el hecho de saber cuál es la opción seleccionada de la lista de respuestas, para ello se hace uso de la etiqueta `ion-radio-group` ofrecida por Ionic y la directiva de Angular `[(ngModel)]`. La etiqueta `ion-radio-group` permite agrupar una serie de botones radio (ver sección anterior). De este modo Ionic sabe cuántos botones generamos con la directiva `*ngFor`. Por otro lado, la directiva `[(ngModel)]` permite asociar el valor que toma `ion-radio-group` a una variable. Así, podemos saber la opción seleccionada o modificar su contenido desde el TypeScript.

Para la verificar si la opción seleccionada es correcta, se hace una comparación del elemento seleccionado (que está asociado a una variable como se ha explicado previamente) con la opción correcta proporcionada por el servicio Angular “*SemanticSeriesService*”. Acertar o fallar respuestas, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

### 3.4.3.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento “*exerciseInfo*” mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento “*assistantChange*” al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, se realiza una petición HTTP de tipo POST al backend utilizando el servicio Angular denominado “*MedalsService*”. Esta petición envía la puntuación final obtenida en el ejercicio, la cual la recoge un endpoint implementado en el backend. Dicho endpoint tiene asociada una función que calcula la medalla asociada a la puntuación recibida y la almacena en la base de datos. Una vez calculada, el ejercicio hace una petición de tipo GET a otro endpoint con el fin de obtener la medalla calculada y poder mostrarla en pantalla. A continuación, utilizando el servicio Angular “*ExerciseResultsService*” se envía a otro endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento “*exerciseEnded*” al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

### 3.4.4. PositionsExercise

El ejercicio PositionsExercise se desarrolló con el fin de estimular la memoria del usuario. Para ello, el ejercicio muestra una serie de ventanas, como si de un edificio se tratase. Algunas de estas ventanas se presentarán abiertas durante un corto periodo de tiempo durante el que usuario debe intentar memorizar su posición. Una vez acabe este tiempo, se mostrará el edificio con las ventanas cerradas y el usuario deberá seleccionar aquellas que estaban abiertas durante la fase de memorización. Si el usuario acierta, avanzará en el ejercicio, teniendo que repetir cada edificio dos veces.

#### 3.4.4.1. Fases

1. INTRO: muestra una breve descripción del ejercicio y los iconos que se mostrarán durante este para familiarizar al usuario con ellos (ver [figura 51](#)).



Figura 51. Fase INTRO PositionsExercise

2. COUNTDOWN: se muestra una cuenta atrás para el comienzo del ejercicio, esta fase es recurrente en gran parte de los ejercicios desarrollados en este proyecto (ver [figura 41](#)).
3. BUILDING: durante esta fase se mostrará el edificio con las ventanas abiertas para que el usuario lo memorice (ver [figura 52](#)). También se indica el tiempo restante en la zona del asistente.

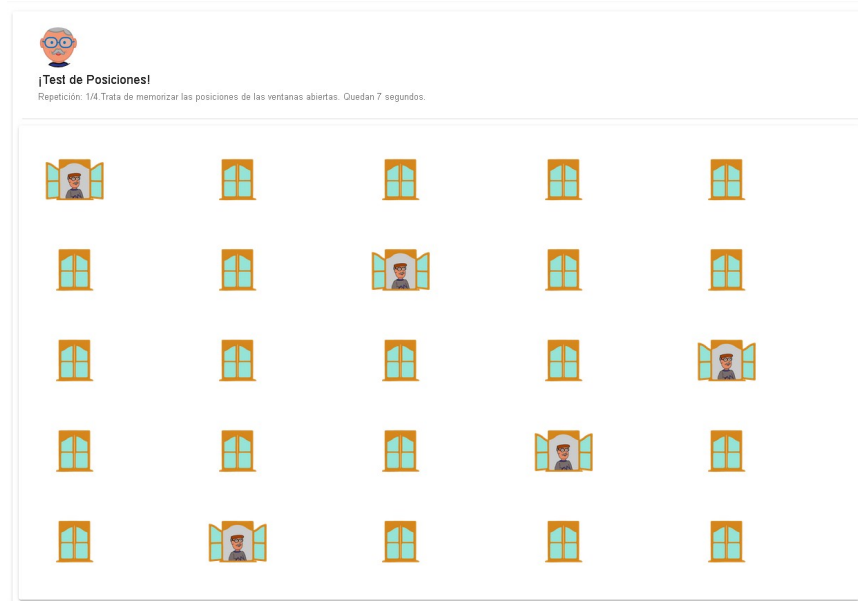


Figura 52. Fase BUILDING PositionsExercise

4. ANSWER: cuando acabe el tiempo de memorización, se muestran las ventanas cerradas para que el usuario seleccione aquellas que estaban abiertas (ver [figura 53](#)). Una vez que haya terminado de seleccionar se comprobará si ha respondido correctamente.

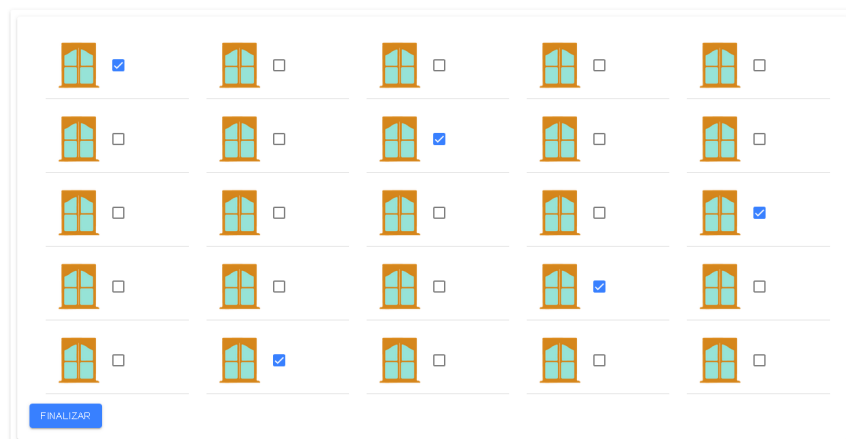


Figura 53. Fase ANSWER PositionsExercise

5. REP\_TRY\_AGAIN: en el caso de que el usuario acierte con su respuesta, se le mostrará el siguiente mensaje (ver [figura 54](#)) y tendrá que repetir el edificio una vez más.



Figura 54. Fase REP\_TRY\_AGAIN PositionsExercise

6. NEXT\_BUILDING: en el caso de que el usuario acierte por segunda vez



el edificio se le mostrará el siguiente mensaje (ver [figura 55](#)) y se avanzará a la introducción del siguiente edificio.



Figura 55. Fase NEXT\_BUILDING PositionsExercise

7. NEXT\_BUILDING\_INTRO: fase de introducción al siguiente edificio (ver [figura 56](#)).

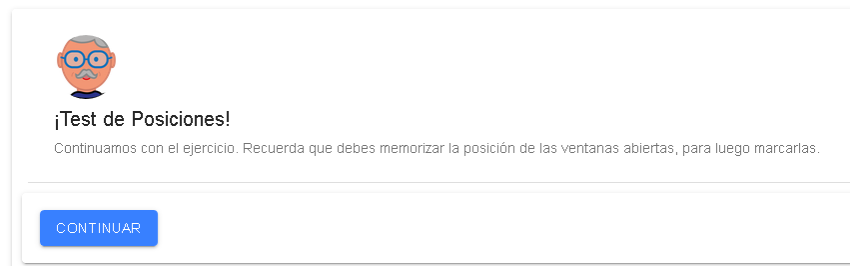


Figura 56. Fase NEXT\_BUILDING\_INTRO PositionsExercise

8. END: fase de finalización del ejercicio, en esta fase se guarda la puntuación obtenida en el ejercicio y se manda al backend para su almacenaje y cálculo de la medalla para este ejercicio que se mostrará en pantalla (ver [figura 45](#)). Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.
9. ERR\_TRY\_AGAIN: si el usuario ha fallado a la hora de seleccionar las ventanas durante la fase ANSWER, se le llevará a una fase de error y después se le mostrará esta pantalla para repetir el ejercicio (ver [figura 57](#)).

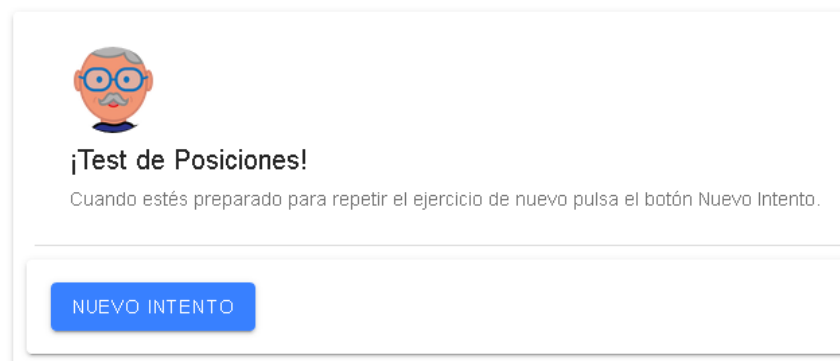


Figura 57. Fase ERR\_TRY\_AGAIN PositionsExercise

10. ERR\_1: si el usuario ha fallado por primera vez al seleccionar las ventanas, se le mostrará esta fase, en la que podrá ver la solución el tiempo que necesite (ver [figura 58](#)).

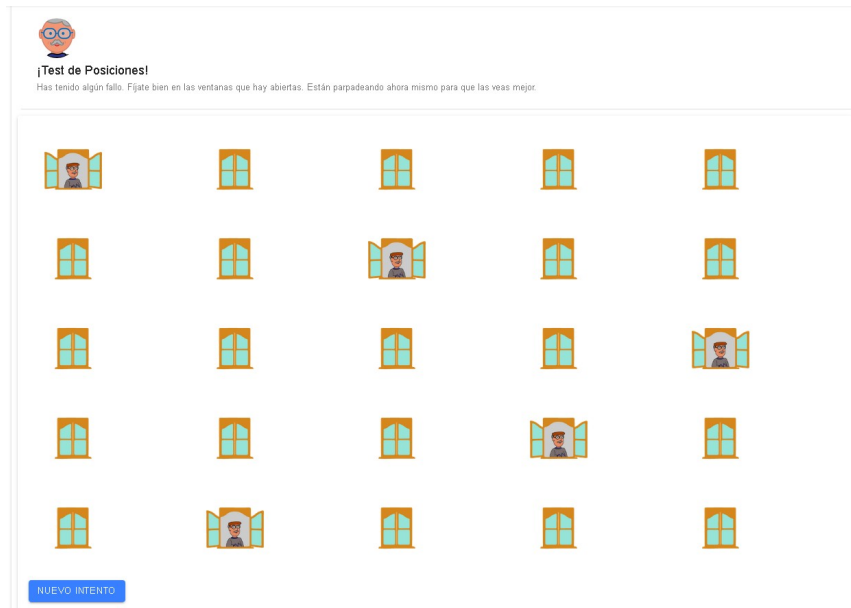


Figura 58. Fase ERR\_1 PositionsExercise

11. ERR\_2: en el caso de que el usuario falle una segunda vez al seleccionar las soluciones, se le mostrará el siguiente mensaje (ver [figura 59](#)) ofreciendo pistas sobre cómo recordar las ventanas abiertas.

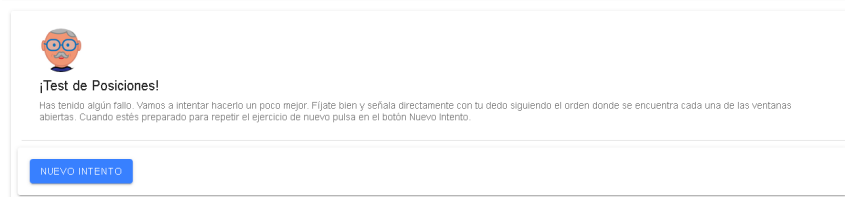


Figura 59. Fase ERR\_2 PositionsExercise

12. ERR\_3: si el usuario falla una tercera vez, se le mostrará por pantalla un ejemplo sobre cómo recordar las ventanas abiertas en función de la posición en la que se encuentran (ver [figura 60](#)).

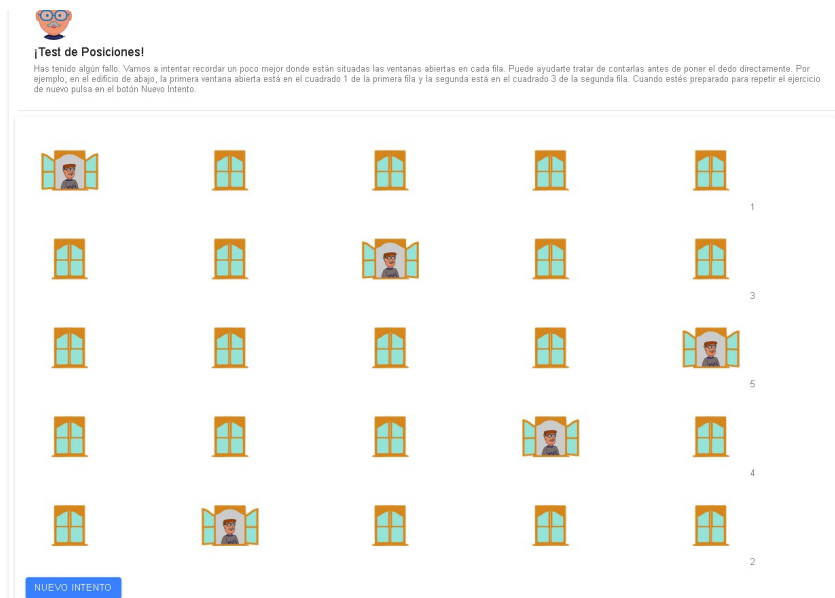


Figura 60. Fase ERR\_3 PositionsExercise

13. ERR\_4: al fallar por cuarta vez, se terminará de explicar el mecanismo para recordar las ventanas abiertas propuesto en la fase ERR\_3 (ver [figura 61](#)). En el caso de que el usuario vuelva a fallar, se realizará un bucle entre las fases ERR\_3 y ERR\_4.

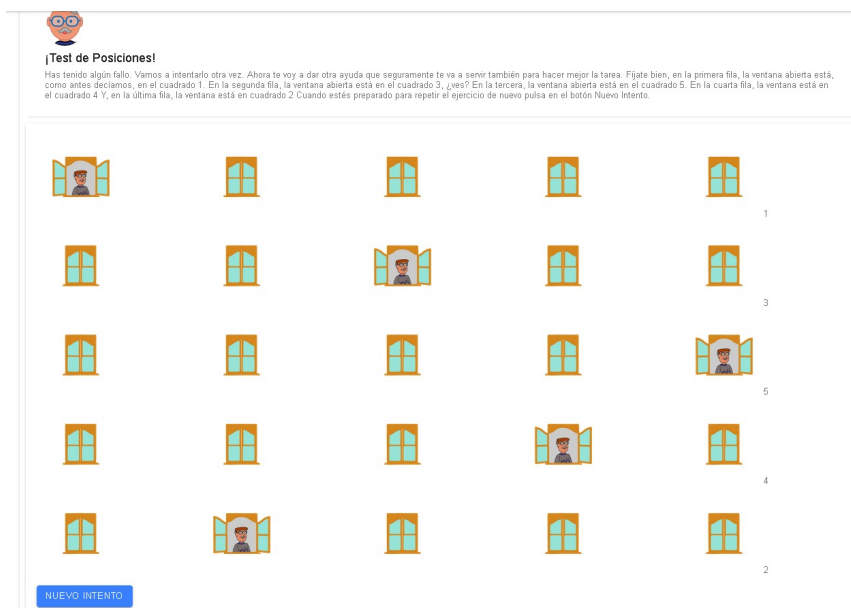


Figura 61. Fase ERR\_4 PositionsExercise

### 3.4.4.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitamos que sean mostrados en pantalla.

Se han utilizado botones de tipo checkbox proporcionados por Ionic para dotar al ejercicio de un aspecto más moderno e intuitivo. Los botones checkbox permiten que varios de ellos estén seleccionados a la vez, siendo una gran opción para un ejercicio donde se requiere pulsar varias ventanas a la vez.

Para mostrar las ventanas y los botones checkbox se ha utilizado la directiva `*ngFor` de Angular. Esta directiva permite iterar sobre una colección de elementos repitiendo el código HTML que contenga la etiqueta donde se incluyó la directiva. Así, no es necesaria la repetición constante de bloques de código. Se han implementado dos series utilizando la directiva `*ngFor`, una para la lista de imágenes que el usuario debe memorizar y otra para la selección de respuestas. Esto se ha complementado con el sistema de grids de Ionic. Este sistema permite ordenar elementos de la interfaz visual en filas y columnas, así, se consigue ese efecto de edificio en la vista final del ejercicio. La primera lista iterada, que el usuario debe memorizar, solo se itera sobre imágenes, para la segunda, en la que el usuario debe responder seleccionando las ventanas que estaban abiertas, se itera sobre bloques de imágenes y botones checkbox.

El código final obtiene una serie de valores positivos respecto a una implementación en la que se repitiese el código. En primer lugar, el código queda mucho más limpio y legible, algo fundamental y que aporta ventajas para el mantenimiento de la aplicación en un futuro, dejando un código breve y fácil de entender. Además, en el caso de que se necesite añadir más listas de ventanas, se podría hacer sin necesidad de modificar la vista pues este código muestra por pantalla cualquier lista de ventanas que se le pase.

### 3.4.4.3. Lógica del ejercicio

Para la cuenta atrás que se muestra en la fase COUNTDOWN y el tiempo de memorización de la fase BUILDING, se han implementado dos temporizadores. Para ello se ha utilizado el método `setInterval` de Angular, el cual permite iterar una acción cada vez que haya pasado un tiempo determinado. De este modo podemos controlar los segundos que pasan desde que empieza la cuenta atrás y el tiempo que se puede ver el edificio.

Dado que el ejercicio puede generar cierta confusión, se ha implementado un sistema de ayuda al usuario en el caso de que falle repetidamente el ejercicio. Este sistema activa las fases ERR\_1, ERR\_2, ERR\_3 y ERR\_4 en el caso de que el usuario no consiga avanzar en el ejercicio. Esto se realiza comprobando la solución aportada cada vez, en el caso de fallar se muestran las ayudas.

Dado que los botones se generan de forma iterativa con la directiva `*ngFor` tal y como se explicó en la sección anterior, podría no ser trivial el hecho de saber cuál es la opción seleccionada de la lista de respuestas, para ello se ha implementado una propiedad asociada a cada botón de la lista de ventanas, el cual indica si el botón está pulsado o no. Esta propiedad y la directiva de Angular `[(ngModel)]` permiten modificar en tiempo real el estado del botón entre seleccionado y deseleccionado. Por tanto, y con el fin de comprobar si la solución que ha seleccionado el usuario es correcta, podemos comprobar el estado de cada botón con la respuesta correcta proporcionada por el servicio Angular `PositionsExerciseService`. Acertar o fallar respuestas, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

### 3.4.4.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento “*exerciseInfo*” mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

También durante la inicialización del componente, este hace una petición GET mediante el servicio Angular “*PositionsExerciseService*” al backend, para que este le envíe todas las rutas y atributos de las imágenes que se van a utilizar durante el ejercicio (ver [Comunicaciones frontend-backend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento “*assistantChange*” al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, se realiza una petición HTTP de tipo POST al backend utilizando el servicio Angular denominado “*MedalsService*”. Esta petición envía la puntuación final obtenida en el ejercicio, la cual la recoge un endpoint implementado en el backend. Dicho endpoint tiene asociada una función que calcula la medalla asociada a la puntuación recibida y la almacena en la base de datos. Una vez calculada, el ejercicio hace una petición de tipo GET a otro endpoint con el fin de obtener la medalla calculada y poder mostrarla en pantalla. A continuación, utilizando el servicio Angular “*ExerciseResultsService*” se envía a otro endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento “*exerciseEnded*” al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

### 3.4.5. DirectNumbersExercise

DirectNumbersExercise es un ejercicio de memoria en el cual se muestran al usuario una serie de números. Tras un periodo de memorización, el usuario debe utilizar el pad numérico que se le muestra para introducir los números en el mismo orden que se le mostraron.

#### 3.4.5.1. Fases

1. INTRO: en esta fase se muestran las instrucciones del ejercicio y una pizarra para familiarizar al usuario con el ejercicio (ver [figura 62](#)).



Figura 62. Fase INTRO DirectNumbersExercise

2. DEMO\_INTRO: antes de comenzar el ejercicio se realizan tres pruebas para determinar que el usuario es capaz de realizar el ejercicio (ver [figura 63](#)).

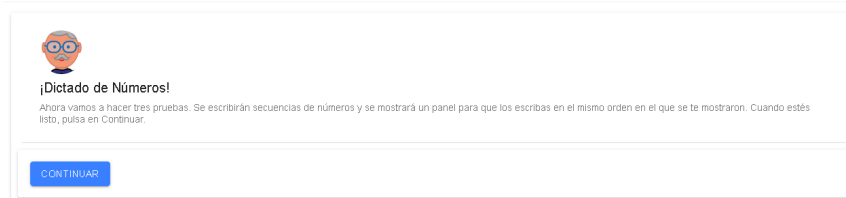


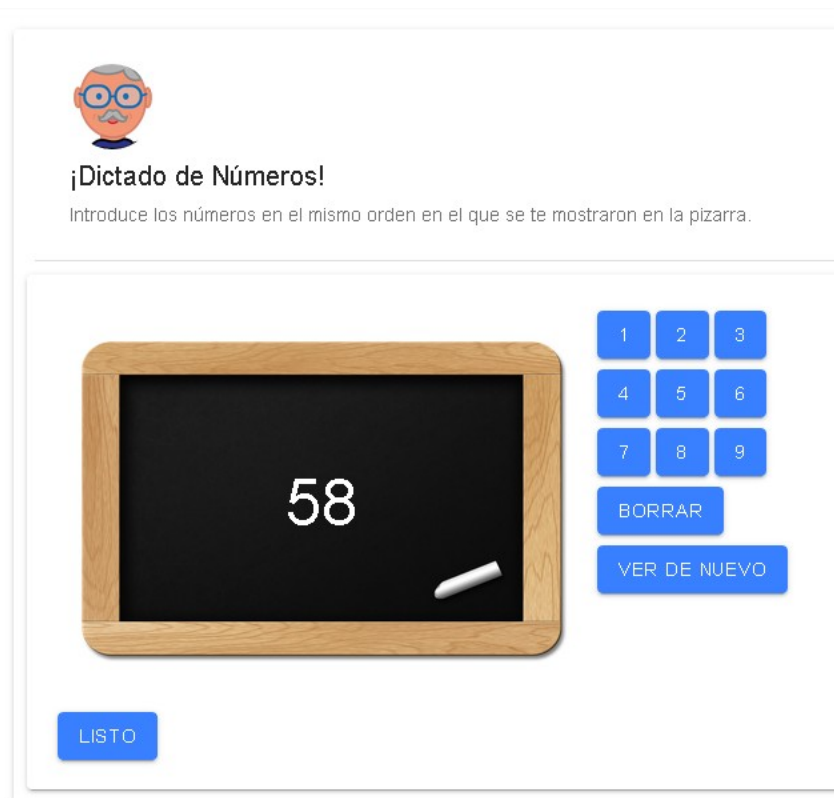
Figura 63. Fase DEMO\_INTRO DirectNumbersExercise

3. WATCH: esta fase se utilizará durante todo el ejercicio para mostrar en la pizarra las series de números (ver [figura 64](#)).



Figura 64. Fase WATCH DirectNumbersExercise

4. ANSWER\_DEMO: se muestra un panel con números para que el usuario introduzca la serie que acaba de ver, también tiene la opción de borrar o ver de nuevo la solución (ver [figura 65](#)).



¡Dictado de Números!

Introduce los números en el mismo orden en el que se te mostraron en la pizarra.

58

1 2 3  
4 5 6  
7 8 9

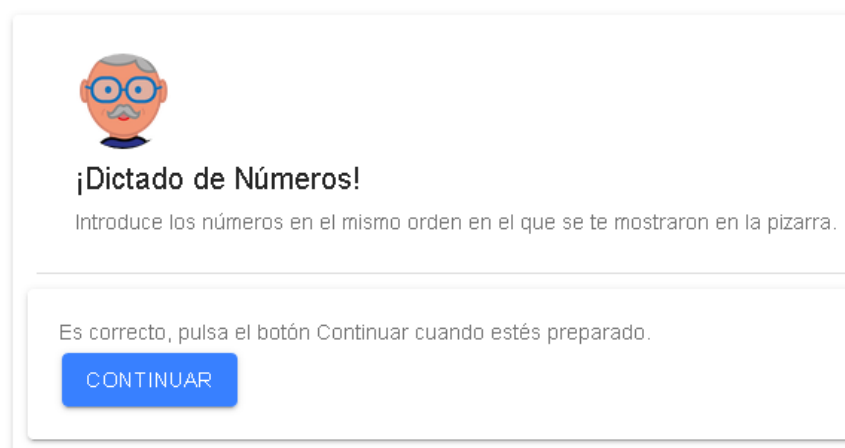
BORRAR

VER DE NUEVO

LISTO

Figura 65. Fase ANSWER\_DEMO DirectNumbersExercise

5. CORRECT\_ANSWER: indica al usuario que su respuesta es correcta y puede avanzar al siguiente ejercicio de la demo (ver [figura 66](#)).



¡Dictado de Números!

Introduce los números en el mismo orden en el que se te mostraron en la pizarra.

Es correcto, pulsa el botón Continuar cuando estés preparado.

CONTINUAR

Figura 66. Fase CORRECT\_ANSWER DirectNumbersExercise

6. ERR\_1: en el caso de que el usuario se equivoque durante la demo, se le mostrará un panel indicándole la solución correcta (ver [figura 67](#)).

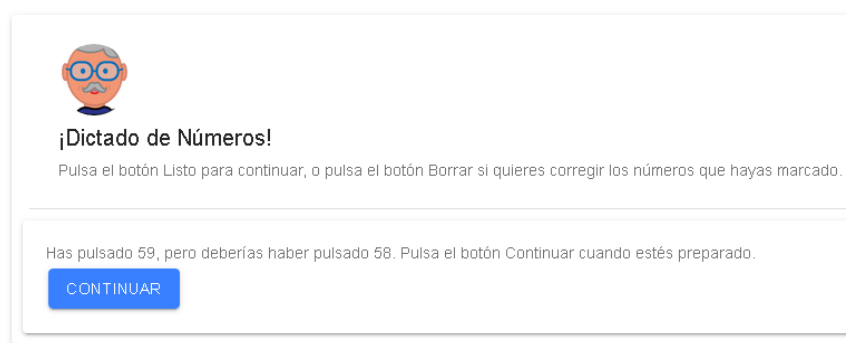


Figura 67. Fase ERR\_1 DirectNumbersExercise

7. DEMO\_END: una vez finalizadas las pruebas se mostrará el siguiente mensaje (ver [figura 68](#)).

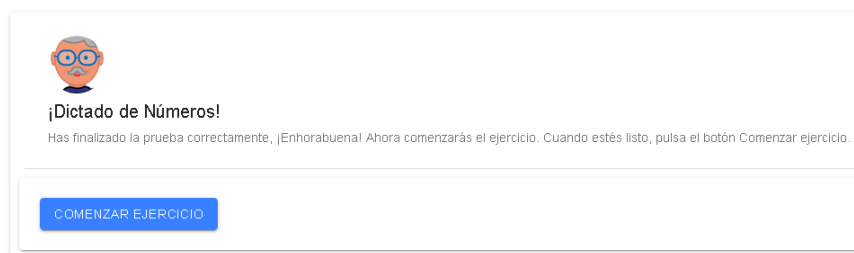


Figura 68. Fase DEMO\_END DirectNumbersExercise

8. COUNTDOWN: se muestra una cuenta atrás para el comienzo del ejercicio, esta fase es recurrente en gran parte de los ejercicios desarrollados en este proyecto (ver [figura 41](#)).
9. ANSWER: panel similar al de ANSWER\_DEMO, con la diferencia de que este no tiene la opción de volver a ver la serie (ver [figura 69](#)).

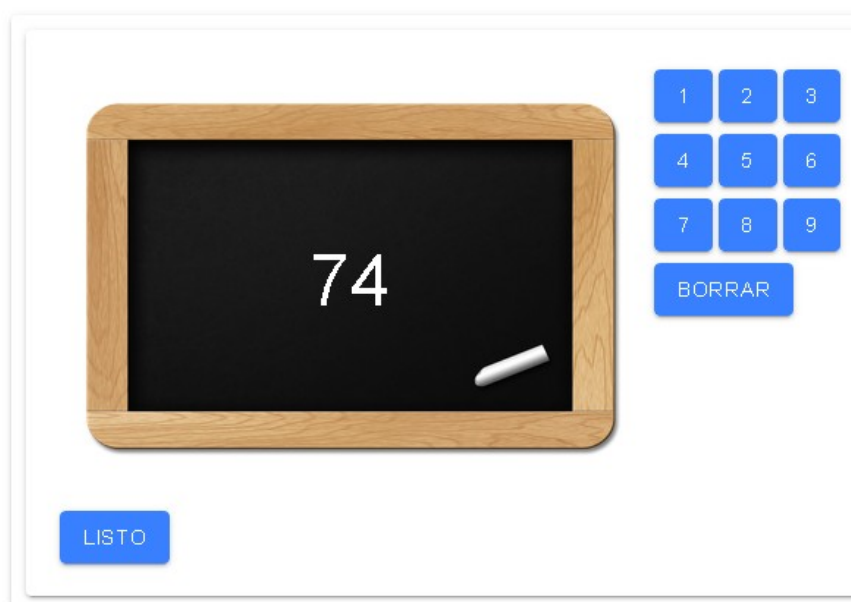


Figura 69. Fase ANSWER DirectNumbersExercise



10. NEXT: fase de transición entre ejercicios (ver [figura 70](#)).

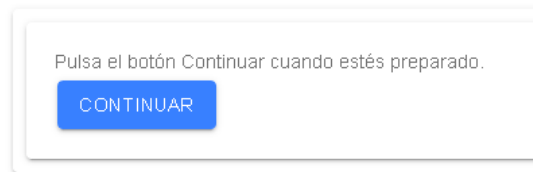


Figura 70. Fase NEXT DirectNumbersExercise

11. END: fase de finalización del ejercicio, en esta fase se guarda la puntuación obtenida en el ejercicio y se manda al backend para su almacenaje y cálculo de la medalla para este ejercicio que se mostrará en pantalla (ver [figura 45](#)). Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.

### 3.4.5.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitemos que sean mostrados en pantalla.

En este ejercicio ha sido necesaria la organización de los paneles de botones para seleccionar las respuestas. Para ello se ha utilizado el sistema de grids de Ionic, el cual permite organizar por filas y columnas los elementos de una interfaz. Con este sistema se han organizado los botones numéricos y opciones para que tengan un aspecto de pad numérico como el de un teléfono móvil.

También ha sido necesario modificar distintas propiedades CSS para poder mostrar los números sobre la pizarra. Estas propiedades han sido `position` y `transform` para poder traer a un primer plano los números y moverlos al centro de la pizarra.

### 3.4.5.3. Lógica del ejercicio

Para la cuenta atrás que se muestra en la fase COUNTDOWN y el tiempo de memorización de la fase WATCH, se han implementado dos temporizadores. Para ello se ha utilizado el método `setInterval` de Angular, el cual permite iterar una acción cada vez que haya pasado un tiempo determinado. De este modo podemos controlar los segundos que pasan desde que empieza la cuenta atrás y el tiempo que pasa entre cada cambio de carácter en la pizarra.

Para mostrar los números en la pizarra durante la fase WATCH, se ha implementado un sistema que separa los números en cifras individuales y se asignan a una variable. Esta variable cambia cada un cierto tiempo, controlado por el temporizador previamente descrito. De este modo cambiamos el contenido de la variable dinámicamente, dejando un código limpio al que se le puede pasar cualquier tipo de número que sea necesario en un futuro.

Acertar o fallar respuestas, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

### 3.4.5.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento “*exerciseInfo*” mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

También durante la inicialización del componente, este hace una petición GET mediante el servicio Angular “*DirectNumbersService*” al backend, para que este le envíe todas las rutas y atributos de las imágenes que se van a utilizar durante el ejercicio (ver [Comunicaciones frontend-backend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento “*assistantChange*” al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, se realiza una petición HTTP de tipo POST al backend utilizando el servicio Angular denominado “*MedalsService*”. Esta petición envía la puntuación final obtenida en el ejercicio, la cual la recoge un endpoint implementado en el backend. Dicho endpoint tiene asociada una función que calcula la medalla asociada a la puntuación recibida y la almacena en la base de datos. Una vez calculada, el ejercicio hace una petición de tipo GET a otro endpoint con el fin de obtener la medalla calculada y poder mostrarla en pantalla. A continuación, utilizando el servicio Angular “*ExerciseResultsService*” se envía a otro endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento “*exerciseEnded*” al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

### 3.4.6. NumbersAndVowelsExercise

NumbersAndVowelsExercise es un ejercicio de memoria en el cual se muestran al usuario una serie de números y vocales. Tras un periodo de memorización, el usuario debe utilizar dos grupos de botones, uno de números y otro de vocales, para introducir la serie que vio previamente pero de forma ordenada. El orden que debe seguir es el siguiente, primero los números en orden ascendente y después las vocales en orden: a, e, i, o y u.

#### 3.4.6.1. Fases

1. **INTRO:** en esta fase se muestran las instrucciones del ejercicio y una pizarra para familiarizar al usuario con el ejercicio (ver [figura 71](#)).

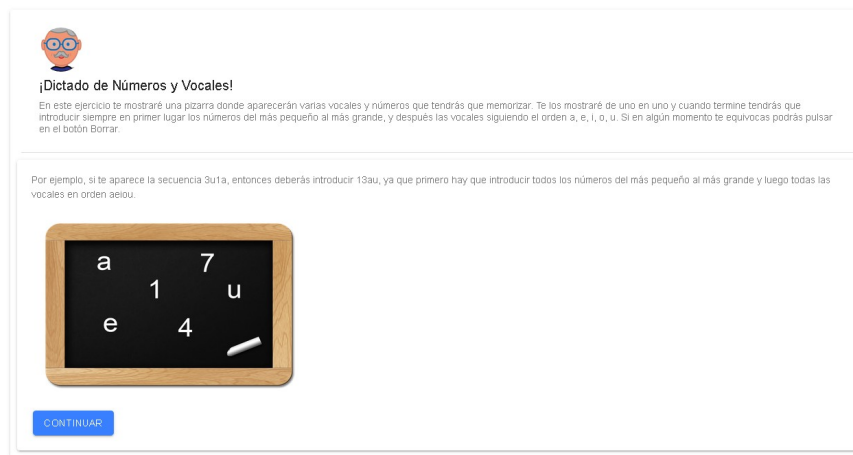


Figura 71. Fase INTRO NumbersAndVowelsExercise

2. **DEMO\_INTRO:** antes de comenzar el ejercicio se realizan tres pruebas para determinar que el usuario es capaz de realizar el ejercicio (ver [figura 72](#)).

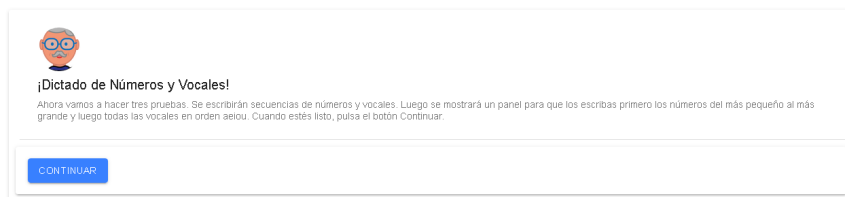



Figura 72. Fase DEMO\_INTRO NumbersAndVowelsExercise

3. **WATCH:** esta fase se utilizará durante todo el ejercicio para mostrar en la pizarra las series de números y vocales (ver [figura 73](#)).




Figura 73. Fase WATCH NumbersAndVowelsExercise

4. ANSWER\_DEMO: se muestran dos paneles, uno con números y otro con vocales para que el usuario introduzca la serie que acaba de ver ordenada, también tiene la opción de borrar o ver de nuevo la solución (ver [figura 74](#)).



**¡Dictado de Números y Vocales!**


Introduce primero los números que se mostraron en la pizarra del más pequeño al más grande y luego las vocales en orden aeioü.



1	2	3
4	5	6
7	8	9
A	E	I
O	U	

Figura 74. Fase ANSWER\_DEMO NumbersAndVowelsExercise

5. CORRECT\_ANSWER: indica al usuario que su respuesta es correcta y puede avanzar al siguiente ejercicio de la demo (ver [figura 75](#)).



**¡Dictado de Números y Vocales!**

Introduce primero los números que se mostraron en la pizarra del más pequeño al más grande y luego las vocales en orden aeioü.

Es correcto, pulsa el botón Continuar cuando estés preparado.

Figura 75. Fase CORRECT\_ANSWER NumbersAndVowelsExercise

6. ERR\_1: en el caso de que el usuario se equivoque durante la demo, se le mostrará un panel indicándole la solución correcta (ver [figura 76](#)).

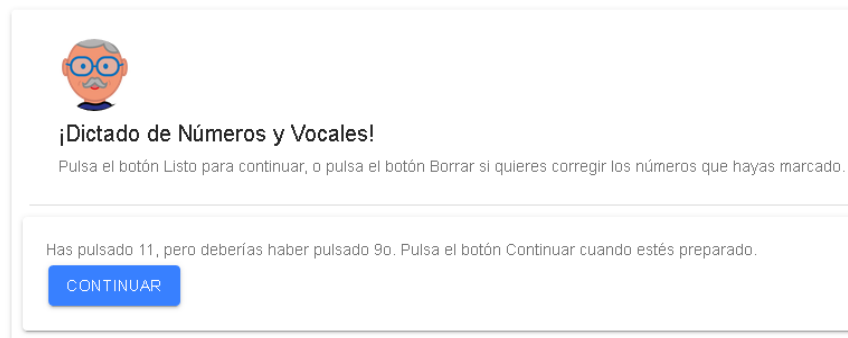


Figura 76. Fase ERR\_1 NumbersAndVowelsExercise

7. DEMO\_END: una vez finalizadas las pruebas se mostrará el siguiente mensaje (ver [figura 77](#)).

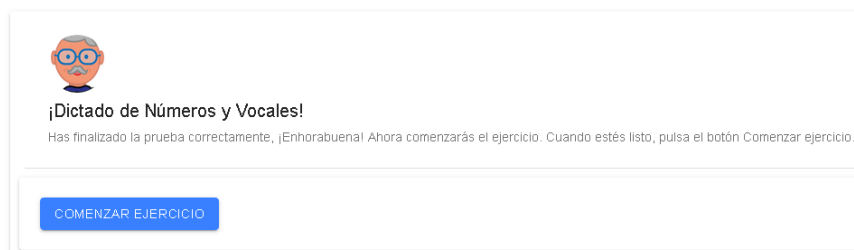


Figura 77. Fase DEMO\_END NumbersAndVowelsExercise

8. COUNTDOWN: se muestra una cuenta atrás para el comienzo del ejercicio, esta fase es recurrente en gran parte de los ejercicios desarrollados en este proyecto (ver [figura 41](#)).
9. ANSWER: panel similar al de ANSWER\_DEMO, con la diferencia de que este no tiene la opción de volver a ver la serie (ver [figura 78](#)).

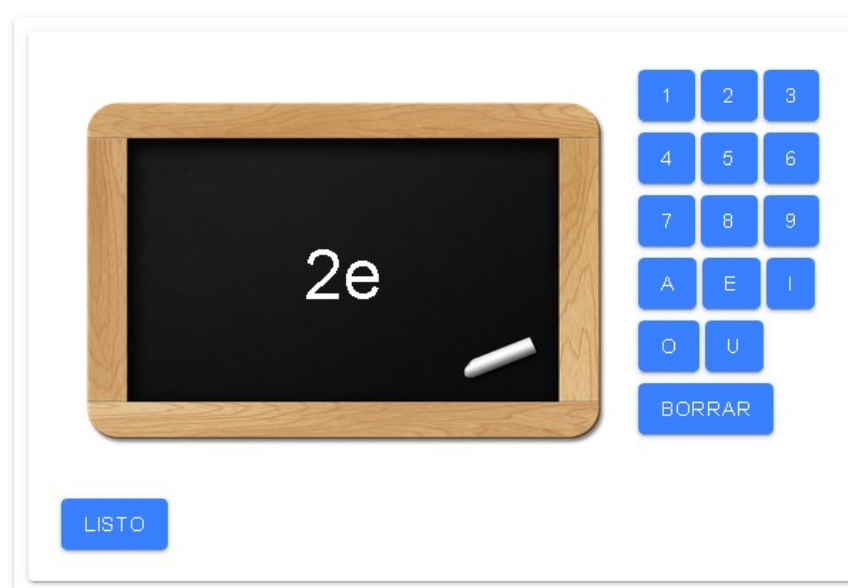


Figura 78. Fase ANSWER NumbersAndVowelsExercise

10. NEXT: fase de transición entre ejercicios (ver [figura 79](#)).

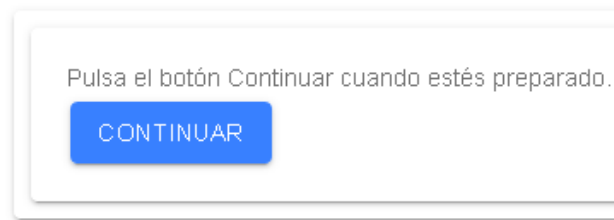


Figura 79. Fase NEXT NumbersAndVowelsExercise

11. END: fase de finalización del ejercicio, en esta fase se guarda la puntuación obtenida en el ejercicio y se manda al backend para su almacenaje y cálculo de la medalla para este ejercicio que se mostrará en pantalla (ver [figura 45](#)). Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.

### 3.4.6.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitemos que sean mostrados en pantalla.

En este ejercicio ha sido necesaria la organización de los paneles de botones para seleccionar las respuestas. Para ello se ha utilizado el sistema de grids de Ionic, el cual permite organizar por filas y columnas los elementos de una interfaz. Con este sistema se han organizado los botones numéricos, de vocales y opciones para que tengan un aspecto de pad numérico como el de un teléfono móvil.

También ha sido necesario modificar distintas propiedades CSS para poder mostrar los números y vocales sobre la pizarra. Estas propiedades han sido `position` y `transform` para poder traer a un primer plano los símbolos y moverlos al centro de la pizarra.

### 3.4.6.3. Lógica del ejercicio

Para la cuenta atrás que se muestra en la fase COUNTDOWN y el tiempo de memorización de la fase WATCH, se han implementado dos temporizadores. Para ello se ha utilizado el método `setInterval` de Angular, el cual permite iterar una acción cada vez que haya pasado un tiempo determinado. De este modo podemos controlar los segundos que pasan desde que empieza la cuenta atrás y el tiempo que pasa entre cada cambio de carácter en la pizarra.

Para mostrar los números y vocales en la pizarra durante la fase WATCH, se ha implementado un sistema que separa las cadenas de caracteres en caracteres individuales y se asignan a una variable. Esta variable cambia cada un cierto tiempo, controlado por el temporizador previamente descrito. De este modo cambiamos el contenido de la variable dinámicamente, dejando un código limpio al que se le puede pasar cualquier tipo de cadena que sea necesaria en un futuro.

Acertar o fallar respuestas, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

#### 3.4.6.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento “*exerciseInfo*” mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

También durante la inicialización del componente, este hace una petición GET mediante el servicio Angular “*NumbersAndVowelsService*” al backend, para que este le envíe todas las rutas y atributos de las imágenes que se van a utilizar durante el ejercicio (ver [Comunicaciones frontend-backend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento “*assistantChange*” al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, se realiza una petición HTTP de tipo POST al backend utilizando el servicio Angular denominado “*MedalsService*”. Esta petición envía la puntuación final obtenida en el ejercicio, la cual la recoge un endpoint implementado en el backend. Dicho endpoint tiene asociada una función que calcula la medalla asociada a la puntuación recibida y la almacena en la base de datos. Una vez calculada, el ejercicio hace una petición de tipo GET a otro endpoint con el fin de obtener la medalla calculada y poder mostrarla en pantalla. A continuación, utilizando el servicio Angular “*ExerciseResultsService*” se envía a otro endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento “*exerciseEnded*” al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

### 3.4.7. PyramidsExercise

Este ejercicio pretende medir y estimular la capacidad de atención del usuario. Con este fin expone una serie de reglas al usuario para completar satisfactoriamente el ejercicio. Durante el mismo, se le mostrarán por tandas series de imágenes de pirámides. El usuario debe seleccionar aquellas que contengan una pirámide roja y dos pequeñas con puerta en el lado soleado. Como se ha mencionado, estas series de imágenes se muestran en tandas y el usuario tendrá un tiempo limitado para ver y seleccionar las imágenes antes de pasar a la siguiente tanda.

#### 3.4.7.1. Fases

1. INTRO1: durante esta fase se muestran los iconos del ejercicio para familiarizar al usuario con estos y se ofrecen unas instrucciones (ver [figura 80](#)).



Figura 80. Fase INTRO1 PyramidsExercise

2. INTRO2: durante esta fase se explica que hay imágenes que pueden no ser correctas y se ofrecen distintos ejemplos (ver [figura 81](#)).



Figura 81. Fase INTRO2 PyramidsExercise

3. DEMO: se realiza una pequeña demostración para que el usuario se acostumbre al ejercicio. Debe seleccionar un número concreto de imágenes que cumplan los requerimientos previamente descritos para poder avanzar (ver [figura 82](#)).





Figura 82. Fase DEMO PyramidsExercise

4. DEMO\_END: una vez seleccionadas todas las imágenes correctas de la fase DEMO, se muestra este mensaje antes de comenzar el ejercicio (ver [figura 83](#)).

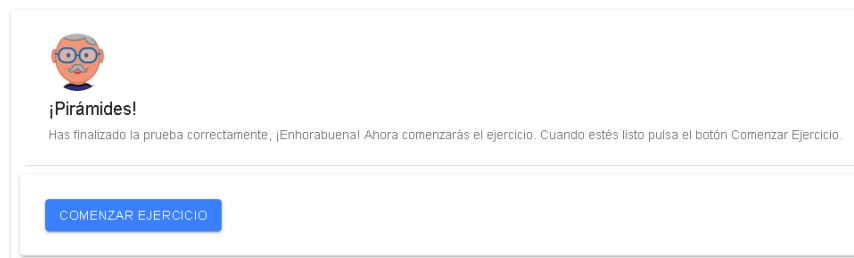


Figura 83. Fase DEMO\_END PyramidsExercise

5. COUNTDOWN: se muestra una cuenta atrás para el comienzo del ejercicio, esta fase es recurrente en gran parte de los ejercicios desarrollados en este proyecto (ver [figura 41](#)).
6. EXERCISE: durante cada iteración del ejercicio se muestra una serie de imágenes para ser seleccionadas (ver [figura 84](#)) durante un periodo concreto de tiempo.

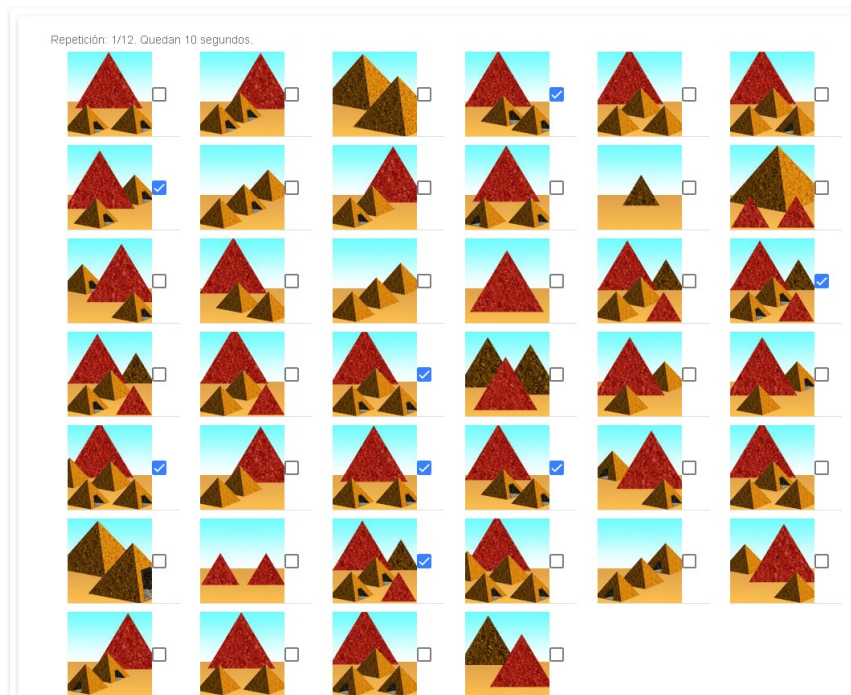


Figura 84. Fase EXERCISE PyramidsExercise

7. EXERCISE\_CHANGE: se muestra entre cambios de series de imágenes (ver [figura 85](#)). El ejercicio consta de doce iteraciones en su versión actual.

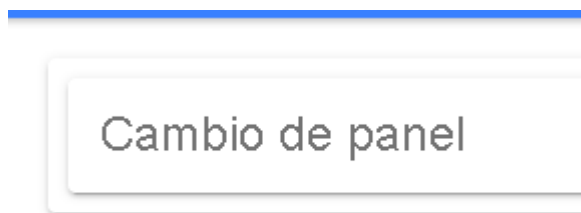


Figura 85. Fase EXERCISE\_CHANGE PyramidsExercise

8. END: fase de finalización del ejercicio, en esta fase se guarda la puntuación obtenida en el ejercicio y se manda al backend para su almacenaje y cálculo de la medalla para este ejercicio que se mostrará en pantalla (ver [figura 45](#)). Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.

### 3.4.7.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitamos que sean mostrados en pantalla.

Se han utilizado botones de tipo checkbox proporcionados por Ionic para dotar al ejercicio de un aspecto más moderno e intuitivo. Los botones checkbox permiten

que varios de ellos estén seleccionados a la vez, siendo una gran opción para un ejercicio donde se requiere pulsar varias pirámides a la vez.

Para mostrar las imágenes de pirámides y los botones checkbox se ha utilizado la directiva `*ngFor` de Angular. Esta directiva permite iterar sobre una colección de elementos repitiendo el código HTML que contenga la etiqueta donde se incluyó la directiva. Así, no es necesaria la repetición constante de bloques de código. La directiva `*ngFor` se ha complementado con el sistema de grids de Ionic. Este sistema permite ordenar elementos de la interfaz visual en filas y columnas, así, se consigue ese efecto de matriz en la vista final del ejercicio. Para crear la lista de imágenes seleccionables, se itera sobre bloques de imágenes y botones checkbox.

El código final obtiene una serie de valores positivos respecto a una implementación en la que se repitiese el código. En primer lugar, el código queda mucho más limpio y legible, algo fundamental y que aporta ventajas para el mantenimiento de la aplicación en un futuro, dejando un código breve y fácil de entender. Además, en el caso de que se necesite añadir más listas de pirámides, se podría hacer sin necesidad de modificar la vista pues este código muestra por pantalla cualquier lista de pirámides que se le pase.

### 3.4.7.3. Lógica del ejercicio

Para la cuenta atrás que se muestra en la fase COUNTDOWN y el tiempo de selección de la fase EXERCISE, se han implementado dos temporizadores. Para ello se ha utilizado el método `setInterval` de Angular, el cual permite iterar una acción cada vez que haya pasado un tiempo determinado. De este modo podemos controlar los segundos que pasan desde que empieza la cuenta atrás y el tiempo de cada serie.

Dado que los botones se generan de forma iterativa con la directiva `*ngFor` tal y como se explicó en la sección anterior, podría no ser trivial el hecho de saber cuál es la opción seleccionada de la lista de respuestas, para ello se ha implementado una propiedad asociada a cada botón de la lista de ventanas, el cual indica si el botón está pulsado o no. Esta propiedad y la directiva de Angular `[(ngModel)]` permiten modificar en tiempo real el estado del botón entre seleccionado y deseleccionado. Por tanto, y con el fin de comprobar si la solución que ha seleccionado el usuario es correcta, podemos comprobar el estado de cada botón con la respuesta correcta proporcionada por el servicio Angular `PyramidsService`. Acertar o fallar respuestas, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

### 3.4.7.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento `exerciseInfo` mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

También durante la inicialización del componente, este hace una petición GET mediante el servicio Angular `PyramidsService` al backend, para que este le envíe todas las rutas y atributos de las imágenes que se van a utilizar durante el ejercicio (ver [Comunicaciones frontend-backend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la

fase actual del ejercicio y envía una notificación con el evento “*assistantChange*” al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, se realiza una petición HTTP de tipo POST al backend utilizando el servicio Angular denominado “*MedalsService*”. Esta petición envía la puntuación final obtenida en el ejercicio, la cual la recoge un endpoint implementado en el backend. Dicho endpoint tiene asociada una función que calcula la medalla asociada a la puntuación recibida y la almacena en la base de datos. Una vez calculada, el ejercicio hace una petición de tipo GET a otro endpoint con el fin de obtener la medalla calculada y poder mostrarla en pantalla. A continuación, utilizando el servicio Angular “*ExerciseResultsService*” se envía a otro endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento “*exerciseEnded*” al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

### 3.4.8. WordListExerciseAlternative

WordListExerciseAlternative pretende estimular la memoria del usuario haciéndole recordar la lista de palabras mostrada en el ejercicio WordListExercise. Como es lógico, es necesario haber mostrado el ejercicio WordListExercise al usuario antes de este, ya que es necesario que conozca las palabras que se muestran en ese ejercicio. Con este objetivo, se pide al usuario que introduzca aquellas palabras que recuerde de dicha lista. Una vez introducidas, se le mostrarán palabras sueltas y el usuario deberá decir si esa palabra estaba en la lista o no.

#### 3.4.8.1. Fases

1. INTRO: en esta fase se pone en contexto al usuario y se ofrecen unas instrucciones sobre lo que debe hacer, escribir las palabras que recuerde de la lista (ver [figura 86](#)).



¡Lista de Palabras (Largo Plazo)!

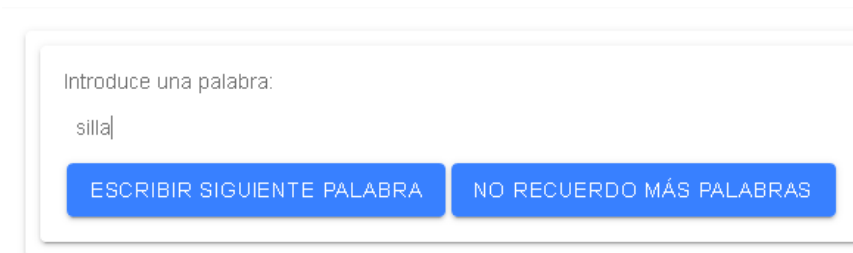
Ahora vamos a ver si recuerdas la LISTA AZUL que memorizaste al principio de esta sesión. Se te mostró tres veces. Para ello, en primer lugar, tendrás que escribir las palabras que recuerdes en un recuadro que se te mostrará, sin importar el orden ni las faltas de ortografía. Lo importante es que sepamos cuáles recuerdas. Intenta ser rápido para que no se te olvide ninguna palabra. Cada vez que escribas una palabra debes pulsar el botón Escribir siguiente palabra. Cuando no recuerdes ninguna más, pulsa No recuerdo más palabras. Pulsa Continuar cuando estés preparado.

Deberás escribir las palabras que recuerdes de la LISTA AZUL. Para ello, deberás ir escribiendo cada palabra en un recuadro que se te mostrará y pulsar el botón Escribir Siguiente Palabra. Cuando termines de escribir las palabras que recuerdes deberás pulsar el botón No recuerdo más palabras.

CONTINUAR

Figura 86. Fase INTRO WordListExerciseAlternative

2. COUNTDOWN: se muestra una cuenta atrás para el comienzo del ejercicio, esta fase es recurrente en gran parte de los ejercicios desarrollados en este proyecto (ver [figura 41](#)).
3. WRITE: el usuario debe introducir en el cuadro de texto aquellas palabras que recuerde (ver [figura 87](#)) y pulsar el botón “No recuerdo más palabras” para finalizar esta fase.



Introduce una palabra:

silla

ESCRIBIR SIGUIENTE PALABRA NO RECUERDO MÁS PALABRAS

Figura 87. Fase WRITE WordListExerciseAlternative

4. INTRO2: se muestran instrucciones sobre la segunda parte del ejercicio (ver [figura 88](#)), donde el usuario debe decidir si la palabra estaba o no en la lista original.



¡Lista de Palabras (Largo Plazo)!

Ahora se te van a mostrar distintas palabras de una en una, algunas de ellas son las que aparecieron en la LISTA AZUL que se te mostró tres veces y que tuviste que memorizar, otras no las has memorizado en ningún momento. Tienes que intentar recordar y pulsar Si, en aquellas palabras que estaban en esa lista y, NO en aquellas que no estaban en la lista.

CONTINUAR

Figura 88. Fase INTRO2 WordListExerciseAlternative

5. EXERCISE: el usuario debe seleccionar el botón “Sí” en el caso de que la palabra estuviese en la lista original y “No” en caso contrario (ver [figura 89](#)).

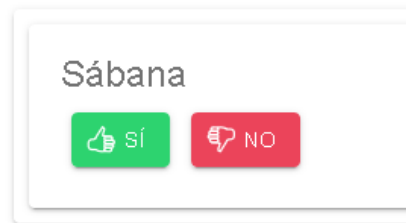


Figura 89. Fase EXERCISE WordListExerciseAlternative

6. END: fase de finalización del ejercicio, en esta fase se guarda la puntuación obtenida en el ejercicio y se manda al backend para su almacenaje y cálculo de la medalla para este ejercicio que se mostrará en pantalla (ver [figura 45](#)). Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.

### 3.4.8.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitamos que sean mostrados en pantalla.

### 3.4.8.3. Lógica del ejercicio

Para la cuenta atrás que se muestra en la fase COUNTDOWN, se ha implementado un temporizador. Para ello se ha utilizado el método `setInterval` de Angular, el cual permite iterar una acción cada vez que haya pasado un tiempo determinado. De este modo podemos controlar los segundos que pasan desde que empieza la cuenta atrás.

Con el objetivo de que el ejercicio sea lo más sencillo posible de realizar, se pensó que sería ideal que el usuario pudiese introducir las palabras de la forma más cómoda para este. Esto quiere decir que el usuario pueda introducir las palabras en minúsculas, mayúsculas, con mayúsculas y minúsculas, con o sin tildes. Para ello se ha implementado una función a la que se ha llamado `lowerCaseRemoveAccents` la cual da formato a una cadena de texto introducida, cambiando todas las mayúsculas por minúsculas e intercambiando todas las letras con acentos por su versión sin este. También se ha implementado este mecanismo para símbolos que no son del español, como acentos circunflejos o diéresis en las vocales, dando soporte a lenguas extranjeras. Todas las respuestas que aporte el usuario se pasarán por esta función para su comprobación con la solución.

Para la comprobación de palabras, se comprueba si son correctas comparándolas con el array de palabras correctas. Si lo son, se comprueba si la palabra ya ha sido añadida al array de palabras correctas respondidas para evitar repeticiones.

En la segunda parte del ejercicio, se muestra una serie de palabras y el usuario deberá seleccionar si existían o no en la lista de palabras original. La palabra mostrada se compara con la lista original con el fin de comprobar si su selección es correcta o no.

Acertar o fallar palabras, en cualquiera de ambas fases del ejercicio, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

### 3.4.8.4. Comunicaciones

El ejercicio toma los datos del servicio Angular “*WordListService*”.

Cuando el componente se inicializa, se suscribe al evento “*exerciseInfo*” mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento “*assistantChange*” al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, se realiza una petición HTTP de tipo POST al backend utilizando el servicio Angular denominado “*MedalsService*”. Esta petición envía la puntuación final obtenida en el ejercicio, la cual la recoge un endpoint implementado en el backend. Dicho endpoint tiene asociada una función que calcula la medalla asociada a la puntuación recibida y la almacena en la base de datos. Una vez calculada, el ejercicio hace una petición de tipo GET a otro endpoint con el fin de obtener la medalla calculada y poder mostrarla en pantalla. A continuación, utilizando el servicio Angular “*ExerciseResultsService*” se envía a otro endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento “*exerciseEnded*” al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

### 3.4.9. MouseDialogExercise

Este ejercicio sirve para determinar si el usuario es capaz de utilizar un ratón en el caso de acceder a la plataforma en un dispositivo de sobremesa o de usar una pantalla táctil si se trata de un móvil o tablet. MouseDialogExercise consiste en pulsar una serie de botones que aparecerán en distintas posiciones de la pantalla. El completar exitosamente este ejercicio determinará que el usuario es apto para realizar los ejercicios de la plataforma.

#### 3.4.9.1. Fases

1. INTRO: muestra una breve explicación del ejercicio (ver [figura 90](#)) antes de comenzar su realización.



Figura 90. Fase INTRO MouseDialogExercise

2. EXERCISE: durante esta fase aparecerán botones en distintas posiciones con el fin de que el usuario los seleccione (ver [figura 91](#)). El número de repeticiones se puede especificar en la creación del ejercicio.

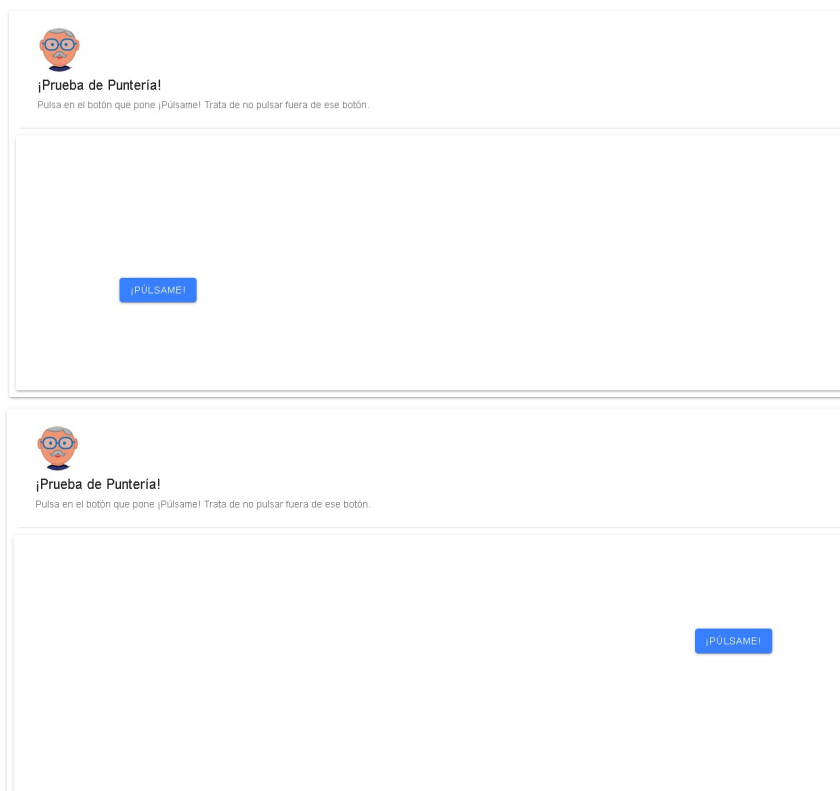


Figura 91. Fase EXERCISE MouseDialogExercise

3. END: fase de finalización del ejercicio, en esta fase se guarda la



puntuación obtenida en el ejercicio y se manda al backend para su almacenaje. Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.

### 3.4.9.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitemos que sean mostrados en pantalla.

Se ha creado un panel central para este ejercicio, donde irán apareciendo botones según se vayan seleccionando.

### 3.4.9.3. Lógica del ejercicio

Para la generación de los botones se ha creado una función que calcula una nueva posición para el botón teniendo en cuenta si el dispositivo es de sobremesa o móvil. Además asegura que el botón nunca saldrá del área delimitada en el ejercicio. Una vez calculada la nueva posición, se envía a la interfaz mediante una variable que contiene los valores CSS de position, left y top. Esta variable se asigna al nuevo botón mediante la directiva Angular `[ngStyle]`. Esta directiva permite proporcionar valores CSS a un elemento HTML de forma dinámica.

Acertar o fallar respuestas, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

### 3.4.9.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento `exerciseInfo` mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento `assistantChange` al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, utilizando el servicio Angular `ExerciseResultsService` se envía a un endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento `exerciseEnded` al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

### 3.4.10. IntroductionExercise

IntroductionExercise recoge tres ejercicios de la versión original (IntroductionExercise, QuestionnaireExercise, FirstHelpExercise) y los unifica en un solo ejercicio. El motivo de esta unificación se debe a que estos ejercicios se realizaban seguidos durante la primera sesión y nunca se volvían a repetir a lo largo de las sesiones, por eso se consideró oportuno unificarlos en un solo ejercicio. El ejercicio se divide en tres partes principales. INTROS, que coincide con la versión original de IntroductionExercise, QUESTIONS, que coincide con la versión original de QuestionnaireExercise y END que coincide con FirstHelpExercise.

#### 3.4.10.1. Fases

1. INTRO1, INTRO2, INTRO3: introducen al usuario sobre el objetivo de la plataforma (ver [figura 92](#)), planificación, sus sesiones y cómo deben llevarse a cabo.



Figura 92. Fase INTRO1 IntroductionExercise

2. INTRO4: se pregunta al usuario si ha utilizado alguna vez un ordenador (ver [figura 93](#)).



Figura 93. Fase INTRO4 IntroductionExercise

3. INTRO5: se pide al usuario que lea y acepte los términos y condiciones de la plataforma para poder continuar (ver [figura 94](#)).



Figura 94. Fase INTRO5 IntroductionExercise

4. QUESTION1-16: una vez finalizada la introducción, se realizará un cuestionario al usuario. Este cuestionario consta de un total de dieciséis fases en las que se realizarán preguntas de carácter personal al usuario con el fin de obtener datos estadísticos o que puedan ayudar a su terapeuta (ver [figura 95](#)).

Figura 95. Fase QUESTION1 IntroductionExercise

5. END: fase de finalización del ejercicio, en esta fase se guardan todas las respuestas al cuestionario dadas por el usuario y se mandan al backend para su almacenaje. Una vez pulsado en el botón continuar se avanzará al siguiente ejercicio (ver [figura 96](#)).

Figura 96. Fase END IntroductionExercise

El ejercicio inyecta el servicio “*QuestionnaireService*” para poder mandar las respuestas obtenidas en el cuestionario al backend para su almacenaje.

### 3.4.10.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular `*ngIf`, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitemos que sean mostrados en pantalla.

Durante la primera fase del ejercicio, asociada a `IntroductionExercise`, se muestran una serie de paneles con información de la plataforma, para que el usuario pueda continuar, debe aceptar cada uno de los paneles. Una vez aceptados, se mostrará un documento PDF con términos y condiciones de la plataforma.

En la segunda fase, que hace referencia a `QuestionnaireExercise`, se muestran una serie de preguntas que el usuario debe responder, para su implementación se han utilizado botones de tipo radio proporcionados por Ionic para dotar al ejercicio de un aspecto más moderno e intuitivo. Los botones radio solo permiten que uno de ellos esté seleccionado cada vez, siendo ideales para cuestionarios de tipo test. Para las preguntas que requieren escritura, se han utilizado los componentes Ionic `"ion-label"` e `"ion-input"`. `ion-label` permite dar formato a una etiqueta de texto, además, cambiando su propiedad `position` a `"floating"`, podemos hacer que aparezca encima de la entrada de datos con un aspecto moderno. La entrada de datos está proporcionada por `ion-input`. Se han modificado los valores `type` de `ion-input` para especificar el tipo de entrada de datos de cada campo.

### 3.4.10.3. Lógica del ejercicio

Es obligatorio aceptar los términos y condiciones mostrados en el PDF para poder avanzar y seguir utilizando la plataforma.

Para saber qué respuestas se han seleccionado de los botones radio durante el cuestionario se hace uso de la etiqueta `"ion-radio-group"` ofrecida por Ionic y la directiva de Angular `"[(ngModel)]"`. La etiqueta `ion-radio-group` permite agrupar una serie de botones radio (ver sección anterior). De este modo Ionic sabe cuántos botones generamos con la directiva `*ngFor`. Por otro lado, la directiva `[(ngModel)]` permite asociar el valor que toma `ion-radio-group` a una variable. Así, podemos saber la opción seleccionada o modificar su contenido desde el TypeScript.

Todas las respuestas que se dan en el cuestionario se guardan en un objeto de la clase `QuestionnaireAnswers`, la cual contiene un atributo por cada pregunta del cuestionario.

### 3.4.10.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento `"exerciseInfo"` mediante el cual la página `session` le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

Para cambiar el contenido del componente `Assistant`, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento `"assistantChange"` al componente `Assistant` para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, utilizando el servicio Angular “*QuestionnaireService*” se envía a un endpoint las respuestas a todas las preguntas del cuestionario para que sean almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento “*exerciseEnded*” al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

### 3.4.11. InstrumentalQuestionnaireExercise

InstrumentalQuestionnaireExercise realiza una tarea similar al IntroductionExercise, unificando dos ejercicios que se realizaban seguidos y una sola vez durante el curso de VIRTRA-EL, SecondHelpExercise e InstrumentalQuestionnaireExercise. SecondHelpExercise coincide con la fase INTRO de esta nueva implementación y InstrumentalQuestionnaireExercise con las fases QUESTION1-8. El objetivo de este ejercicio es recopilar algunos datos más sobre el usuario una vez haya finalizado la primera sesión de VIRTRA-EL

#### 3.4.11.1. Fases

1. INTRO: esta fase simplemente indica al usuario que se le van a realizar otra tanda de preguntas antes de finalizar la sesión (ver [figura 97](#)).

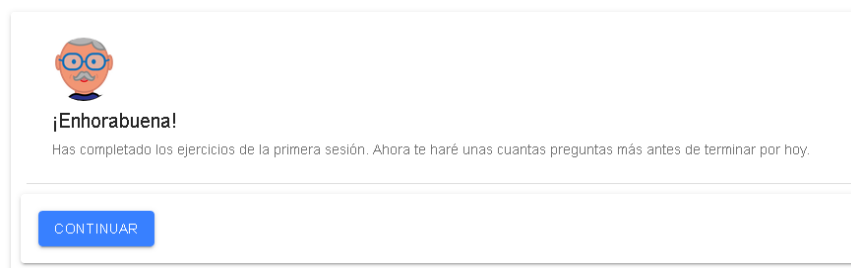


Figura 97. Fase INTRO InstrumentalQuestionnaireExercise

2. QUESTION1-8: las fases QUESTION tienen ocho variantes, en cada una de ellas se realiza una pregunta diferente al usuario (ver [figura 98](#)). Una vez finalizadas las preguntas, el ejercicio finaliza.

Figura 98. Fase QUESTION1 InstrumentalQuestionnaireExercise

#### 3.4.11.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular \*ngIf, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitemos que sean mostrados en pantalla.

Durante la fase del cuestionario se muestran una serie de preguntas que el usuario debe responder, para su implementación se han utilizado botones de tipo radio proporcionados por Ionic para dotar al ejercicio de un aspecto más moderno e intuitivo. Los botones radio solo permiten que uno de ellos esté seleccionado cada vez, siendo ideales para cuestionarios de tipo test. Para las preguntas que requieren escritura, se han utilizado los componentes Ionic *“ion-label”* e *“ion-input”*. Ion-label permite dar formato a una etiqueta de texto, además, cambiando su propiedad position a *“floating”*, podemos hacer que aparezca encima de la entrada de datos con un aspecto moderno. La entrada de datos está proporcionada por ion-input. Se han modificado los valores type de ion-input para especificar el tipo de entrada de datos de cada campo.

### 3.4.11.3. Lógica del ejercicio

Para saber qué respuestas se han seleccionado de los botones radio durante el cuestionario se hace uso de la etiqueta *“ion-radio-group”* ofrecida por Ionic y la directiva de Angular *“[(ngModel)]”*. La etiqueta ion-radio-group permite agrupar una serie de botones radio (ver sección anterior). De este modo Ionic sabe cuántos botones generamos con la directiva \*ngFor. Por otro lado, la directiva [(ngModel)] permite asociar el valor que toma ion-radio-group a una variable. Así, podemos saber la opción seleccionada o modificar su contenido desde el TypeScript.

Todas las respuestas que se dan en el cuestionario se guardan en un objeto de la clase InstrumentalQuestionnaireAnswers, la cual contiene un atributo por cada pregunta del cuestionario.

### 3.4.11.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento *“exerciseInfo”* mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento *“assistantChange”* al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, utilizando el servicio Angular *“InstrumentalQuestionnaireService”* se envía a un endpoint las respuestas a todas las preguntas del cuestionario para que sean almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento *“exerciseEnded”* al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

## 3.4.12. ClassifyObjectsExercise

El ejercicio ClassifyObjectsExercise pretende ejercitar la memoria y capacidad de razonamiento del usuario. Para ello se le van a mostrar una serie de imágenes, cada una perteneciente a un campo semántico. El usuario deberá organizar las imágenes según su categoría durante la demo del ejercicio. Una vez finalizada, se mostrarán una serie de imágenes durante un periodo de tiempo que el usuario deberá recordar. Una vez consumido este tiempo el usuario tendrá que seleccionar aquellas imágenes que se le mostraron de entre un grupo de imágenes.

### 3.4.12.1. Fases

1. INTRO: se muestran las instrucciones del ejercicio al usuario (ver [figura 99](#)).



Figura 99. Fase INTRO ClassifyObjectsExercise

2. DEMO1: primera parte de la explicación de la demo (ver [figura 100](#)).



Figura 100. Fase DEMO1 ClassifyObjectsExercise

3. DEMO2: segunda parte de la explicación de la demo (ver [figura 101](#)), previa a la realización del ejercicio.



Figura 101. Fase DEMO2 ClassifyObjectsExercise

4. DEMO\_EXERCISE: fase de la demo del ejercicio. Durante esta fase el usuario debe seleccionar una imagen y seleccionar la posición en la que la quiere colocar con el objetivo de que cada imagen esté en su categoría (ver [figura 102](#)). Una vez que todas las imágenes estén en su posición correcta se avanzará a la siguiente fase.



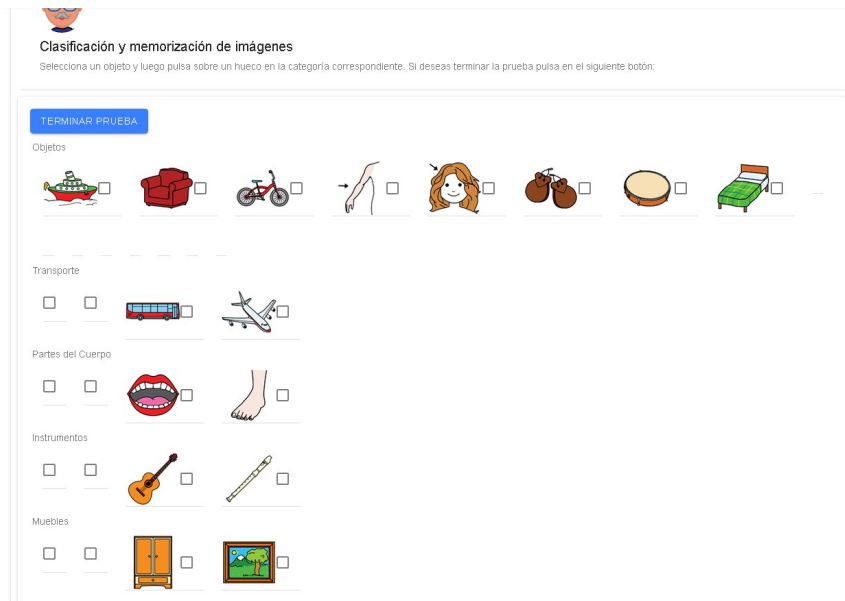


Figura 102. Fase DEMO\_EXERCISE ClassifyObjectsExercise

5. DEMO\_END: una vez finalizada la demo se mostrará el siguiente mensaje y comenzará el ejercicio (ver [figura 103](#)).

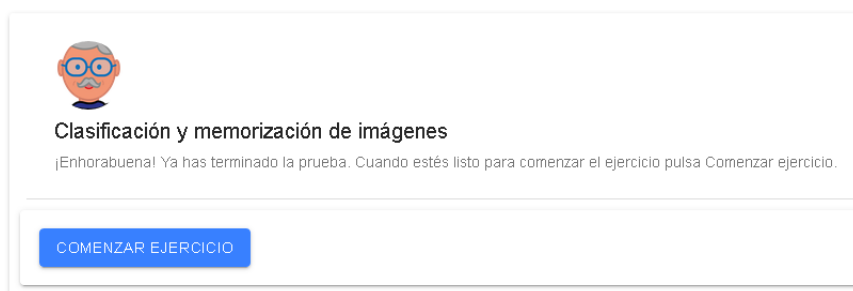


Figura 103. Fase DEMO\_END ClassifyObjectsExercise

6. EXERCISE\_INTRO: mensaje de explicación del ejercicio (ver [figura 104](#)).



Figura 104. Fase EXERCISE\_INTRO ClassifyObjectsExercise

7. COUNTDOWN: se muestra una cuenta atrás para el comienzo del ejercicio, esta fase es recurrente en gran parte de los ejercicios desarrollados en este proyecto (ver [figura 41](#)).
8. MEMORIZE: durante esta fase se muestran las imágenes que debe memorizar el usuario durante un periodo concreto de tiempo, indicado en la parte superior, cuanto mayor sea el grupo de imágenes que debe memorizar, mayor será el tiempo (ver [figura 105](#)).

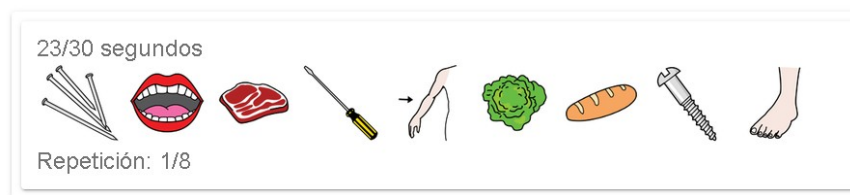


Figura 105. Fase MEMORIZE ClassifyObjectsExercise

9. **ANSWER:** una vez finalizado el tiempo de la fase MEMORIZE comenzará esta fase y se le pedirá al usuario que seleccione de la lista mostrada aquellas imágenes que estaban en la fase MEMORIZE (ver [figura 106](#)). Una vez que haya terminado de seleccionar o no recuerde más, debe pulsar el botón “Listo”.

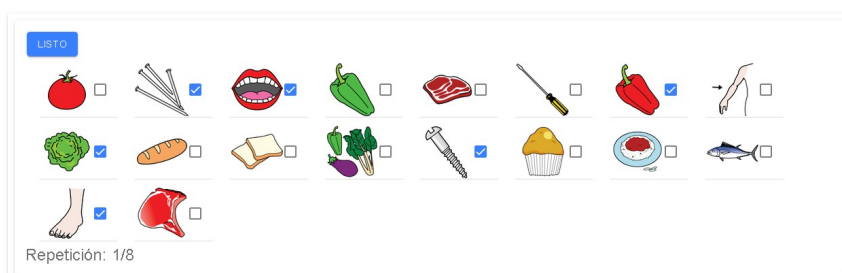


Figura 106. Fase ANSWER ClassifyObjectsExercise

10. **SCORE:** durante esta fase se muestra la lista de objetos así como el número de fallos, aciertos e imágenes olvidadas (ver [figura 107](#)).

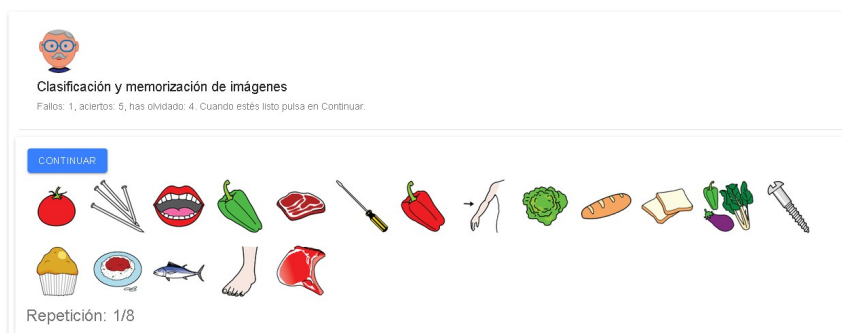


Figura 107. Fase SCORE ClassifyObjectsExercise

11. **NEXT:** fase de iteración entre series de objetos (ver [figura 108](#)), en la versión actual implementada, el ejercicio consta de ocho iteraciones.

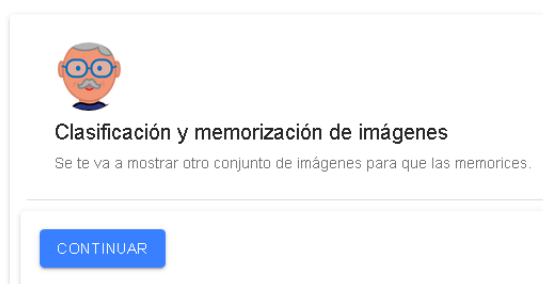


Figura 108. Fase NEXT ClassifyObjectsExercise

12. END: fase de finalización del ejercicio, en esta fase se guarda la puntuación obtenida en el ejercicio y se manda al backend para su almacenaje y cálculo de la medalla para este ejercicio que se mostrará en pantalla (ver [figura 45](#)). Una vez pulsado en el botón finalizar se avanzará al siguiente ejercicio.

Se han utilizado botones de tipo checkbox proporcionados por Ionic para dotar al ejercicio de un aspecto más moderno e intuitivo. Los botones checkbox permiten que varios de ellos estén seleccionados a la vez, siendo ideales para la parte del ejercicio, donde debemos seleccionar varias imágenes a la vez. Para la demo también se han utilizado este tipo de botones pero se ha limitado a que solo dos puedan estar seleccionados a la vez, estos dos botones son los que serán intercambiados. Por otro lado, el ejercicio obtiene los datos del servicio Angular “*ClassifyObjectsService*”. Al comenzar hace una petición al backend para obtener la ruta de todas las imágenes necesarias para el desarrollo del ejercicio. Las imágenes, combinadas con los datos de las series, se muestran en pantalla haciendo uso de la directiva Angular “*\*ngFor*”.

### 3.4.12.2. Interfaz

Para una mejor organización del código de modo que quede más limpio y legible, las fases previamente explicadas se han plasmado en el código utilizando la directiva de Angular *\*ngIf*, la cual muestra y oculta trozos de código HTML mediante la evaluación de una expresión booleana y que, gracias al código TypeScript del componente, se modifica dinámicamente, mostrando y ocultando en cada instante aquellos elementos que necesitemos que sean mostrados en pantalla.

Se han utilizado botones de tipo checkbox proporcionados por Ionic para dotar al ejercicio de un aspecto más moderno e intuitivo. Los botones checkbox permiten que varios de ellos estén seleccionados a la vez, siendo una gran opción para un ejercicio donde se requiere pulsar varias imágenes a la vez.

Para mostrar las imágenes de pirámides y los botones checkbox se ha utilizado la directiva *\*ngFor* de Angular. Esta directiva permite iterar sobre una colección de elementos repitiendo el código HTML que contenga la etiqueta donde se incluyó la directiva. Así, no es necesaria la repetición constante de bloques de código. Para crear la lista de imágenes seleccionables, se itera sobre bloques de imágenes y botones checkbox mientras que para la lista que el usuario debe memorizar solo se itera sobre imágenes.

El código final obtiene una serie de valores positivos respecto a una implementación en la que se repitiese el código. En primer lugar, el código queda mucho más limpio y legible, algo fundamental y que aporta ventajas para el mantenimiento de la aplicación en un futuro, dejando un código breve y fácil de entender. Además, en el caso de que se necesite añadir más listas de imágenes, se podría hacer sin necesidad de modificar la vista pues este código muestra por pantalla cualquier lista de imágenes que se le pase.

### 3.4.12.3. Lógica del ejercicio

Para la cuenta atrás que se muestra en la fase COUNTDOWN y el tiempo de selección de la fase MEMORIZE, se han implementado dos temporizadores. Para ello se ha utilizado el método “*setInterval*” de Angular, el cual permite iterar una acción cada vez que haya pasado un tiempo determinado. De este modo podemos controlar los segundos que pasan desde que empieza la cuenta atrás y el tiempo de cada serie.

Durante la fase DEMO, se muestra un ejercicio un tanto distinto al ejercicio en sí, que se basa en la memorización y no en la clasificación de ejercicios. En esta demo se intercambia la posición de los items mostrados arriba con los espacios en blanco de las categorías. Para ello se ha implementado un sistema que solo permite seleccionar dos checkbox a la vez, e intercambiando el contenido de estos. Esto permite la colocación de las imágenes en las secciones e intercambiarlas en caso de equivocación.

Dado que los botones se generan de forma iterativa con la directiva `*ngFor` tal y como se explicó en la sección anterior, podría no ser trivial el hecho de saber cuál es la opción seleccionada de la lista de respuestas, para ello se ha implementado una propiedad asociada a cada botón de la lista de ventanas, el cual indica si el botón está pulsado o no. Esta propiedad y la directiva de Angular "`[(ngModel)]`" permiten modificar en tiempo real el estado del botón entre seleccionado y deseleccionado. Por tanto, y con el fin de comprobar si la solución que ha seleccionado el usuario es correcta, podemos comprobar el estado de cada botón con la respuesta correcta proporcionada por el servicio Angular "`ClassifyObjectsService`". Acertar o fallar respuestas, incrementa los contadores de acierto y error que se encuentran en el elemento de tipo Score que lleva la cuenta de los resultados del ejercicio.

### 3.4.12.4. Comunicaciones

Cuando el componente se inicializa, se suscribe al evento "`exerciseInfo`" mediante el cual la página session le envía el identificador del usuario que está realizando el ejercicio con el fin de guardar sus puntuaciones y los atributos de inicialización del ejercicio (ver [Comunicaciones frontend-frontend](#)).

También durante la inicialización del componente, este hace una petición GET mediante el servicio Angular "`ClassifyObjectsService`" al backend, para que este le envíe todas las rutas y atributos de las imágenes que se van a utilizar durante el ejercicio (ver [Comunicaciones frontend-backend](#)).

Para cambiar el contenido del componente Assistant, se ha implementado un método que modifica el título, descripción y el estado (mostrar / ocultar) según la fase actual del ejercicio y envía una notificación con el evento "`assistantChange`" al componente Assistant para su modificación (ver [Comunicaciones frontend-frontend](#)).

Antes de finalizar el ejercicio, se realiza una petición HTTP de tipo POST al backend utilizando el servicio Angular denominado "`MedalsService`". Esta petición envía la puntuación final obtenida en el ejercicio, la cual la recoge un endpoint implementado en el backend. Dicho endpoint tiene asociada una función que calcula la medalla asociada a la puntuación recibida y la almacena en la base de datos. Una vez calculada, el ejercicio hace una petición de tipo GET a otro endpoint con el fin de obtener la medalla calculada y poder mostrarla en pantalla. A continuación, utilizando el servicio Angular "`ExerciseResultsService`" se envía a otro endpoint la puntuación final obtenida para que sea almacenada en la base de datos mediante una petición de tipo POST (ver [Comunicaciones frontend-backend](#)).

Una vez finalizado el ejercicio, es obligatorio notificar a la página Session para que la sesión pueda continuar o finalizar, para ello se envía el evento "`exerciseEnded`" al cual la página Session está suscrito, indicando el identificador del ejercicio y si el ejercicio ha finalizado con éxito (ver [Comunicaciones frontend-frontend](#)).

## 4. Conclusiones y trabajos futuros

### 4.1. Conclusiones

Los objetivos presentados en el apartado [Objetivos](#) se han completado satisfactoriamente. Por un lado, los objetivos de investigación, estudio y práctica se han desarrollado en las secciones [Estado del arte](#) y [Fase 0](#). Estos objetivos especificaban el aprendizaje de las tecnologías con las que se ha trabajado durante todo el proyecto, la práctica con dichas tecnologías y el estudio de la aplicación original para desarrollar una adaptación lo más similar posible a la versión original.

Por otro lado, los objetivos de diseño, desarrollo e implementación se han tratado en las secciones [Fase 1](#) y [Fase 2](#). Estos implicaban el diseño de la arquitectura y navegabilidad del sistema, las páginas del sistema y los componentes básicos de la aplicación. También se realizó el diseño y desarrollo de los ejercicios y sesiones, el sistema de carga de ejercicios, el backend y todas las comunicaciones del sistema. Por último se realizó la implementación de una serie de ejercicios que servirán de base para ser ampliada en un futuro.

Por tanto podemos concluir que el objetivo de adaptar la plataforma VIRTRA-EL a nuevas tecnologías ha sido completado con éxito tal y como se ha detallado en este documento y con la aplicación asociada a este. Las tareas que no estaban previstas para ser realizadas durante el desarrollo de este proyecto y que se han ido especificando a lo largo de este documento se agrupan y detallan en la siguiente sección [Trabajos futuros](#).

### 4.2. Trabajos futuros

Aunque el desarrollo de este proyecto ha finalizado, aún quedan una serie de tareas que deben de realizarse con el fin de completar y mejorar la plataforma VIRTRA-EL.

Por ejemplo, tal y como se comentó en la sección [Dominio del problema a resolver](#), se han dejado sin implementar las funcionalidades que contiene el Header una vez accedemos a los ejercicios de la plataforma. Estas funcionalidades incluyen el cambio de la imagen del asistente, los sistemas de asignación de terapeutas y cuidadores, ver las medallas que se han obtenido y poder compartir los resultados por redes sociales.

También se ha dejado para una futura etapa de desarrollo el diseño y la implementación de los formularios de login, registro y contacto. Esto implica el diseño y creación de cuentas de usuario y autenticación, así como el envío de los mensajes a través del formulario de contacto (ver sección [Páginas del sistema](#)).

En la sección [Componentes básicos del sistema](#) se especificó que es necesario realizar las traducciones para el sistema de internacionalización de Angular, de modo que la aplicación pueda ser utilizada por personas que no entiendan español. Para ello, habría que proporcionar el documento de internacionalización que genera Angular de la aplicación a un traductor, lo cual supondría un desembolso adicional. Una vez traducido, Angular es capaz de cambiar el idioma del texto según las necesidades.

Por último y como objetivo principal de futuras fases de desarrollo, sería interesante aumentar el rango de ejercicios de la plataforma con el desarrollo de ejercicios ya existentes en la versión original de VIRTRA-EL y la inclusión de nuevos ejercicios.

## 5. Bibliografía

Lista de enlaces y figuras referenciadas a lo largo del documento.

### 5.1. Referencias

- [1] Página oficial de Angular, <https://angular.io/>
- [2] Página oficial de Ionic, <https://ionicframework.com/>
- [3] Compatibilidad de Angular con navegadores, <https://angular.io/guide/browser-support>
- [4] Compatibilidad de Ionic con navegadores, <https://ionicframework.com/docs/reference/browser-support>
- [5] Licencia de Angular, <https://angular.io/license>
- [6] Licencia de Ionic, <https://ionicframework.com/docs>
- [7] Licencia MIT, <https://opensource.org/licenses/MIT>
- [8] Comparativa de rendimiento de Angular, React y Vue, <https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/>
- [9] Página oficial de TypeScript, <https://www.typescriptlang.org/>
- [10] Comparación entre Ionic y React Native por parte de Ionic Framework, <https://ionicframework.com/resources/articles/ionic-vs-react-native-a-comparison-guide>
- [11] Lista de aplicaciones desarrolladas con Angular, <https://www.madewithangular.com/>
- [12] Página de Microsoft Office Home, <https://www.office.com/apps?auth=2>
- [13] Página de Delta, <https://es.delta.com/eu/es>
- [14] Página del banco Santander Brasil, <https://www.santander.com.br/>
- [15] Página de la revista Forbes, <https://www.forbes.com/#55864a3f2254>
- [16] Página de Blender video, <https://video.blender.org/videos/local>
- [17] Lista de aplicaciones desarrolladas con Ionic, <https://ionicframework.com/customers>
- [18] Página de 86400, <https://www.86400.com.au/>
- [19] Página de Sworkit, <https://sworkit.com/>
- [20] Página de Instant Pot, <https://instantpot.com/>
- [21] Página de Untappd, <https://untappd.com/>
- [22] Documentación de TypeScript, <https://www.typescriptlang.org/docs/home.html>
- [23] Tutorial básico de Angular, <https://angular.io/start#create-a-new-project>
- [24] Tutorial de Angular "Tour of Heroes", <https://angular.io/tutorial>
- [25] Documentación de Ionic, <https://ionicframework.com/docs>
- [26] Documentación sobre componentes Ionic, <https://ionicframework.com/docs/components>
- [27] Documentación sobre la carga dinámica de componentes Angular, <https://angular.io/guide/dynamic-component-loader>
- [28] What is Scrum?, <https://www.scrum.org/resources/what-is-scrum>
- [29] ¿Qué son los tableros Kanban?, <https://www.atlassian.com/es/agile/kanban/boards>
- [30] Web de React.js, <https://es.reactjs.org/>
- [31] Web de Vue.js, <https://vuejs.org/>

## 5.2. Figuras y tablas

- [Figura 1.](#) Diagrama de Gantt del proyecto
- [Figura 2.](#) Tabla de presupuesto
- [Figura 3.](#) WordList Exercise
- [Figura 4.](#) SemanticSeries Exercise
- [Figura 5.](#) ClassifyObject Exercise
- [Figura 6.](#) Pyramids Exercise
- [Figura 7.](#) Rendimiento del ejercicio LogicalSeriesExercise en la versión original y en la nueva
- [Figura 8.](#) Sistema de asignación de terapeutas
- [Figura 9.](#) Medallas obtenidas
- [Figura 10.](#) Tabla comparativa de metodologías
- [Figura 11.](#) Ejemplo de tablero Kanban
- [Figura 12.](#) Tabla comparativa de tecnologías
- [Figura 13.](#) Fases de desarrollo del proyecto
- [Figura 14.](#) Tablero Kanban Fase de desarrollo 0
- [Figura 15.](#) Tablero Kanban Fase de desarrollo 1
- [Figura 16.](#) Tablero Kanban Fase de desarrollo 2
- [Figura 17.](#) Tabla de planificación temporal del proyecto
- [Figura 18.](#) Leyenda arquitectura del sistema
- [Figura 19.](#) Arquitectura del sistema
- [Figura 20.](#) Diagrama de navegabilidad
- [Figura 21.](#) Home page
- [Figura 22.](#) Project page
- [Figura 23.](#) Members page
- [Figura 24.](#) Exercises page
- [Figura 25.](#) Contact page
- [Figura 26.](#) Register page
- [Figura 27.](#) Access page
- [Figura 28.](#) Session page
- [Figura 29.](#) Header component modo escritorio
- [Figura 30.](#) Header component versión móvil
- [Figura 31.](#) Footer component
- [Figura 32.](#) UserHeaderMenu modo escritorio
- [Figura 33.](#) UserHeaderMenu modo móvil
- [Figura 34.](#) Diagrama de la implementación de los ejercicios en la versión original
- [Figura 35.](#) Diagrama de la implementación de los ejercicios en la nueva versión
- [Figura 36.](#) Diagrama del funcionamiento de la página Session
- [Figura 37.](#) Assistant component
- [Figura 38.](#) Diagrama del sistema de comunicación por eventos
- [Figura 39.](#) Diagrama de comunicación entre frontend y backend
- [Figura 40.](#) Fase INTRO WordListExercise
- [Figura 41.](#) Fase COUNTDOWN
- [Figura 42.](#) Fase READ WordListExercise
- [Figura 43.](#) Fase WRITE WordListExercise
- [Figura 44.](#) Fase REPEAT WordListExercise
- [Figura 45.](#) Fase END
- [Figura 46.](#) Fase INTRO LogicalSeriesExercise
- [Figura 47.](#) Fase DEMO LogicalSeriesExercise
- [Figura 48.](#) Fase EXERCISE LogicalSeriesExercise
- [Figura 49.](#) Fase INTRO SemanticSeriesExercise
- [Figura 50.](#) Fase EXERCISE SemanticSeriesExercise
- [Figura 51.](#) Fase INTRO PositionsExercise
- [Figura 52.](#) Fase BUILDING PositionsExercise
- [Figura 53.](#) Fase ANSWER PositionsExercise
- [Figura 54.](#) Fase REP\_TRY\_AGAIN PositionsExercise



[Figura 55.](#) Fase NEXT\_BUILDING PositionsExercise  
[Figura 56.](#) Fase NEXT\_BUILDING\_INTRO PositionsExercise  
[Figura 57.](#) Fase ERR\_TRY\_AGAIN PositionsExercise  
[Figura 58.](#) Fase ERR\_1 PositionsExercise  
[Figura 59.](#) Fase ERR\_2 PositionsExercise  
[Figura 60.](#) Fase ERR\_3 PositionsExercise  
[Figura 61.](#) Fase ERR\_4 PositionsExercise  
[Figura 62.](#) Fase INTRO DirectNumbersExercise  
[Figura 63.](#) Fase DEMO\_INTRO DirectNumbersExercise  
[Figura 64.](#) Fase WATCH DirectNumbersExercise  
[Figura 65.](#) Fase ANSWER\_DEMO DirectNumbersExercise  
[Figura 66.](#) Fase CORRECT\_ANSWER DirectNumbersExercise  
[Figura 67.](#) Fase ERR\_1 DirectNumbersExercise  
[Figura 68.](#) Fase DEMO\_END DirectNumbersExercise  
[Figura 69.](#) Fase ANSWER DirectNumbersExercise  
[Figura 70.](#) Fase NEXT DirectNumbersExercise  
[Figura 71.](#) Fase INTRO NumbersAndVowelsExercise  
[Figura 72.](#) Fase DEMO\_INTRO NumbersAndVowelsExercise  
[Figura 73.](#) Fase WATCH NumbersAndVowelsExercise  
[Figura 74.](#) Fase ANSWER\_DEMO NumbersAndVowelsExercise  
[Figura 75.](#) Fase CORRECT\_ANSWER NumbersAndVowelsExercise  
[Figura 76.](#) Fase ERR\_1 NumbersAndVowelsExercise  
[Figura 77.](#) Fase DEMO\_END NumbersAndVowelsExercise  
[Figura 78.](#) Fase ANSWER NumbersAndVowelsExercise  
[Figura 79.](#) Fase NEXT NumbersAndVowelsExercise  
[Figura 80.](#) Fase INTRO1 PyramidsExercise  
[Figura 81.](#) Fase INTRO2 PyramidsExercise  
[Figura 82.](#) Fase DEMO PyramidsExercise  
[Figura 83.](#) Fase DEMO\_END PyramidsExercise  
[Figura 84.](#) Fase EXERCISE PyramidsExercise  
[Figura 85.](#) Fase EXERCISE\_CHANGE PyramidsExercise  
[Figura 86.](#) Fase INTRO WordListExerciseAlternative  
[Figura 87.](#) Fase WRITE WordListExerciseAlternative  
[Figura 88.](#) Fase INTRO2 WordListExerciseAlternative  
[Figura 89.](#) Fase EXERCISE WordListExerciseAlternative  
[Figura 90.](#) Fase INTRO MouseDialogExercise  
[Figura 91.](#) Fase EXERCISE MouseDialogExercise  
[Figura 92.](#) Fase INTRO1 IntroductionExercise  
[Figura 93.](#) Fase INTRO4 IntroductionExercise  
[Figura 94.](#) Fase INTRO5 IntroductionExercise  
[Figura 95.](#) Fase QUESTION1 IntroductionExercise  
[Figura 96.](#) Fase END IntroductionExercise  
[Figura 97.](#) Fase INTRO InstrumentalQuestionnaireExercise  
[Figura 98.](#) Fase QUESTION1 InstrumentalQuestionnaireExercise  
[Figura 99.](#) Fase INTRO ClassifyObjectsExercise  
[Figura 100.](#) Fase DEMO1 ClassifyObjectsExercise  
[Figura 101.](#) Fase DEMO2 ClassifyObjectsExercise  
[Figura 102.](#) Fase DEMO\_EXERCISE ClassifyObjectsExercise  
[Figura 103.](#) Fase DEMO\_END ClassifyObjectsExercise  
[Figura 104.](#) Fase EXERCISE\_INTRO ClassifyObjectsExercise  
[Figura 105.](#) Fase MEMORIZE ClassifyObjectsExercise  
[Figura 106.](#) Fase ANSWER ClassifyObjectsExercise  
[Figura 107.](#) Fase SCORE ClassifyObjectsExercise  
[Figura 108.](#) Fase NEXT ClassifyObjectsExercise



## 6. Anexo

### 6.1. Cómo ejecutar la aplicación

1. Importar la base de datos que se encuentra en la carpeta “db/”.
2. Levantar la base de datos con el comando “mongod” en windows, si MongoDB está instalado como servicio en Linux, podemos levantar el servicio con el comando “sudo service mongod start”.
3. Lanzamos el backend con el comando “npm run dev” desde la carpeta “backend/”.
4. Lanzamos el frontend con el comando “ionic serve” desde la carpeta “frontend/”.
5. Conectarse a “localhost:8100” para acceder a la aplicación.
6. Si se desea añadir más usuarios a la base de datos, se deben añadir a la colección “users” con el “id” deseado, la “session” actual (la primera es la 1) y el “currentExercise” (siendo 0 el primer ejercicio de cada sesión).

### 6.2. Cómo añadir nuevos ejercicios y sesiones

Para añadir nuevas sesiones (suponiendo que nos encontramos en “frontend/src/app/”):

- Editar el fichero “services/sessions.service.ts”, añadiendo un nuevo elemento al array de sesiones incluyendo:
  - El id de la sesión de tipo de tipo number.
  - Un título para la sesión de tipo string.
  - Una descripción para la sesión de tipo string.
  - Un array con los ejercicios para la sesión (id, duration, repetitions).

Se pueden utilizar el resto de sesiones incluidas en el fichero como referencia.

Para añadir nuevos ejercicios (suponiendo que nos encontramos en “frontend/src/app/”):

- Crea o usa los componentes que necesites y agrúpalos en un solo componente situado en “components/exercises/”.
  - La clase del componente principal debe de tener los siguientes atributos:
    - private userId: number
    - private exerciseAttributes: ExerciseAttributes
  - Incluye el objeto “exerciseManager” en la clase y añade el siguiente código en el constructor:

```
exerciseManager.exerciseInfo.subscribe( data => {
  this.userId = data.userId;
  this.exerciseAttributes = data.attributes;
});
```

- Añade el siguiente código en la función que se llama cuando el ejercicio finaliza:

```
exerciseManager.notifyEnd({
  id: this.exerciseAttributes.id,
  success: true
});
```

- Si necesitas modificar el código del asistente, se puede hacer así:

```
exerciseManager.notifyAssistant({
  show: (true/false),
  title: 'Un título',
  description: 'Una descripción'
});
```

- Si necesitas almacenar los resultados del ejercicio, crea un atributo de tipo “Score”.

- Para enviar al backend los resultados del ejercicio utiliza “ExerciseResultsService” de la siguiente forma:

```
this.exerciseResultsService.addResult(this.userId,
this.exerciseAttributes.id, this.score).subscribe(res => {});
```

- Para generar una medalla con los resultados obtenidos, genera una puntuación final para el ejercicio entre 0 y 10, almacénala en “this.score.finalScore” y envíala de la siguiente forma:

```
this.medalsService.createMedal(this.userId,
this.exerciseAttributes.id, this.score.finalScore).subscribe(res
=> {});
```

- Y para recibir la medalla:

```
this.medalsService.getMedal(this.userId,
this.exerciseAttributes.id).subscribe(res => {});
```

- Incluye la clase del componente en las secciones “declarations: [...]” y “entryComponents: [...]” en “pages/session/session.module.ts”.
- Edita el fichero “services/exercises.service.ts” añadiendo un nuevo elemento en el array de ejercicios incluyendo:
  - El id del ejercicio de tipo number, este id debe coincidir con el id especificado en el array de ejercicios de la sesión.
  - La clase del componente con el ejercicio completo.
  - Un título para el ejercicio de tipo string.

Se pueden utilizar otros ejercicios como referencia a la hora de desarrollar nuevos ejercicios.

# Agradecimientos

Gracias a Carlos Rodríguez Domínguez y María José Rodríguez Fórtiz por ayudarme en el desarrollo de este proyecto de fin de grado.









