Documentación

El pasillo del saber

Cátedra: Prácticas Profesionalizantes

Establecimiento: E.E.T.P. N.º 478 “Dr. Nicolás Avellaneda”

Curso: Informática 6º año

Docente: Diego Gautero

Año: 2.022

Hardware

Índice

Introducción:

Se trata de un pasillo delimitado por andariveles con tres estaciones individuales compuestas, cada una, por una pequeña alfombra de goma EVA de 25cm x 25cm con un sensor, de tipo switch mecánico, integrado.

Cuando el participante ingrese en el pasillo, al pisar cada colchoneta, será detectado por los sensores que generarán el encendido de una luz testigo, indicando que se encuentra en el lugar correcto y expulsarán, además, una tanda de preguntas, del tipo “*multiple choice*”, que el participante visualizará en su teléfono móvil. Una vez respondidas las preguntas, el participante podrá continuar hasta la siguiente estación para repetir el proceso. Por último, encontramos un dispenser al final del pasillo, con una puerta trampa controlada por cerradura EM que, dependiendo del resultado obtenido (puntaje por respuestas correctas), se abrirá dando acceso a un premio estímulo.

En esta sección se mostrará la preparación y el armado de la parte del hardware de nuestro proyecto “El pasillo del saber”

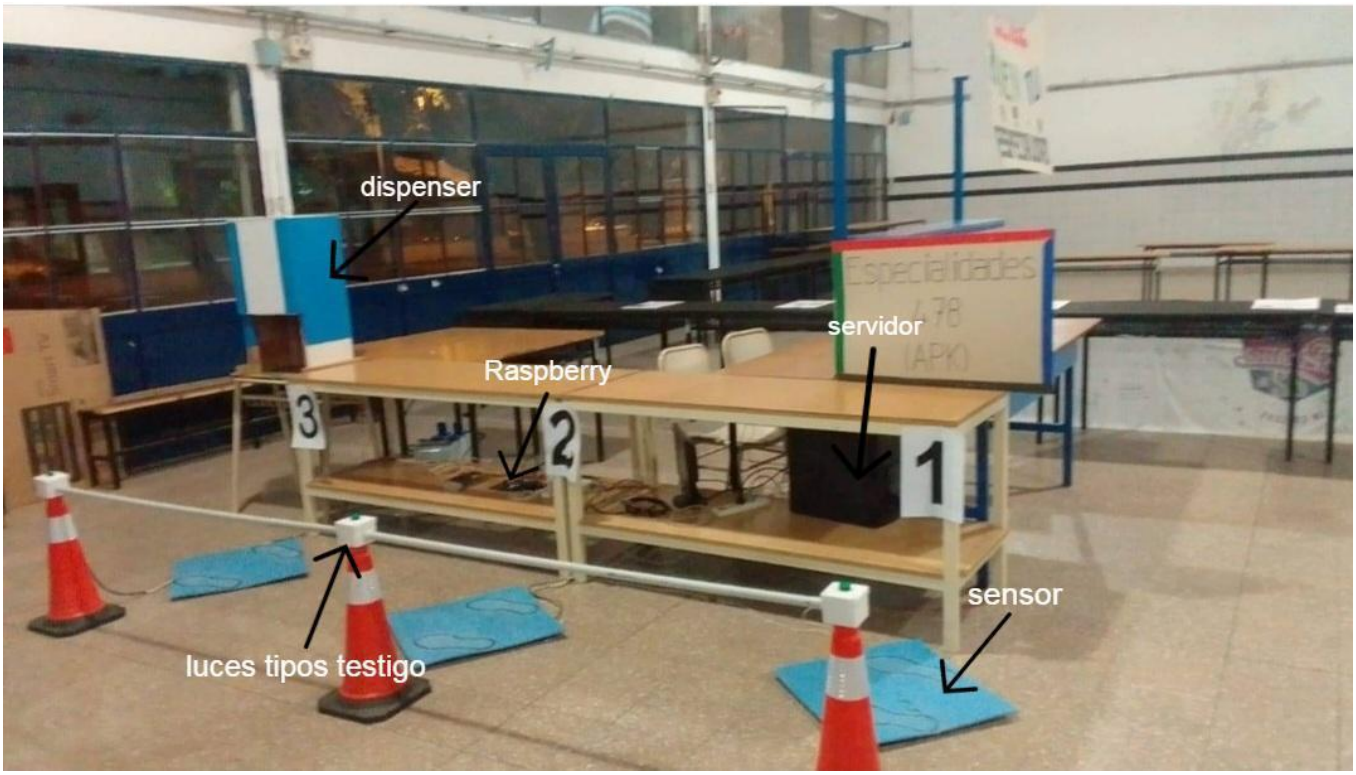


Imagen de “El pasillo del saber”.

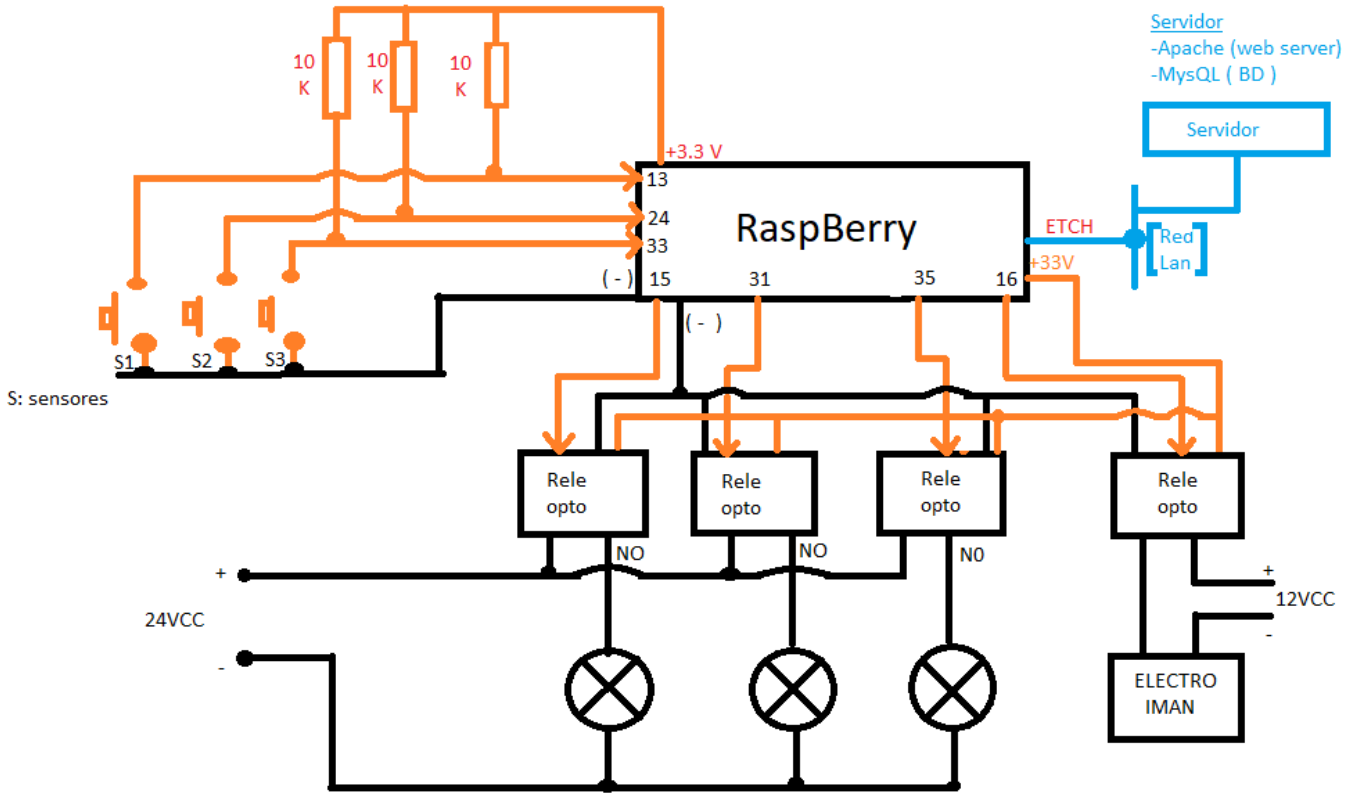


Diagrama funcional físico.

Listado componentes de Hardware:

- **Servidor:** Tipo PC
- **Raspberry Pi 3B+.**
- **Relés con acoplador óptico:** Módulo de 2 (dos) relés optoacoplados, 3.3V - High y Low (2 unidades).
- **Sensores:** Del tipo mecánico de manufactura artesanal
- **Luces testigo:** De tipo LED y 24 VCC.
- **Cerradura:** Tipo electromagnética, marca Zudem, 12 VCC.
- **Cableado sensores (entradas).**
- **Cableado de luces y cerradura EM (salidas).**
- **Cableado puerto GPIO.**

Componentes de Hardware:

Servidor:

Compuesto por una PC, en el mismo corren el servidor web, la Base de Datos, y la aplicación web.

Raspberry:

La Raspberry PI es un ordenador del tamaño de una tarjeta de crédito, que permite interactuar con el mundo físico mediante su puerto GPIO, que permiten generar o leer señales.

Utilizamos la Raspberry como puente entre la aplicación (interfaz lógica del sistema), y el mundo físico (sensores y actuadores).

Funcionamiento:

- Mediante sensores (entradas) capturamos la posición del participante y la Raspberry registra esta situación en la base de datos ubicada en el servidor.
- La aplicación consulta estos registros antes de tomar decisiones, como: encender/apagar las luces o la cerradura EM (las salidas). Además, la posición detectada también condiciona la tanda de preguntas y respuestas correspondiente a cada estación.

Raspberry PI 3.

Acoplador óptico:

Un **optoacoplador** es un componente electrónico que se utiliza como transmisor y receptor óptico (de luz), es decir, pueden transmitir de un punto a otro una señal eléctrica sin necesidad de conexión física ni cables (por el aire), mediante una señal luminosa.

Nosotros, usamos el relé con optoacoplador para aislar la Raspberry del manejo de salidas, cuyo amperaje podría dañarla.

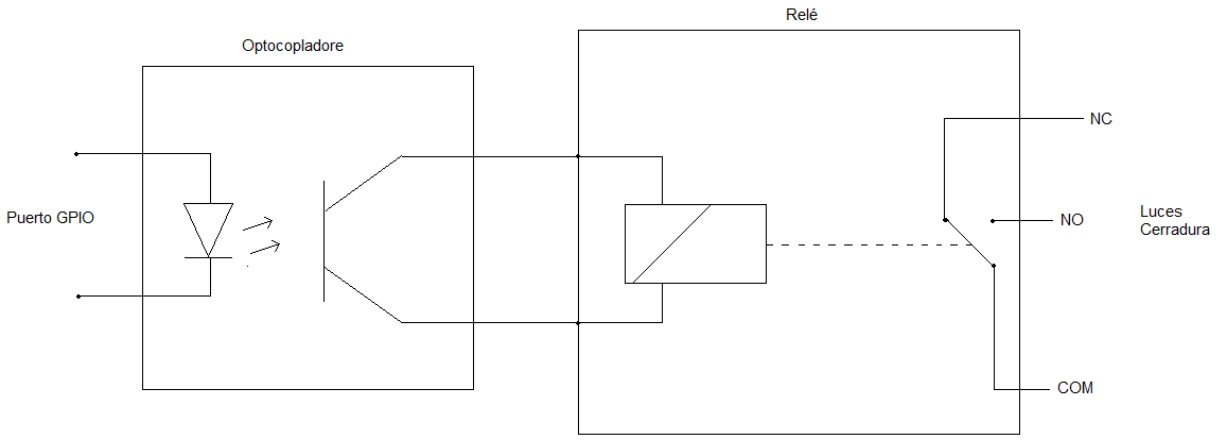


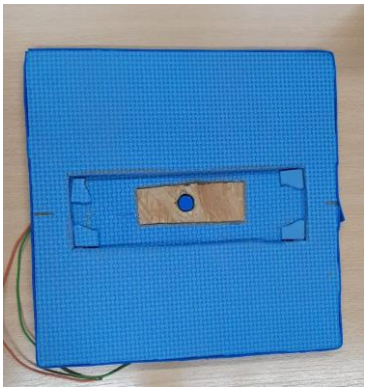
Diagrama de un relé optoacoplado.

Funcionamiento:

- El puerto GPIO de la Raspberry se conecta al optoacoplador, que al recibir una señal genera luz que excita un fototransistor que, a su vez, comanda un relé que es el que maniobra la salida (luz, EM, motor, etc.) con una lógica de “normal abierto” o “normal cerrado”, según corresponda.

Sensores:

Los sensores se encuentran conectados de forma directa a la Raspberry. Los sensores los utilizamos para poder conocer la estación en que se encuentra un participante.



En esta imagen se muestra el sensor al descubierto (botón: switch).

Esta sería la alfombra de goma EVA ensamblada para que los participantes se puedan subir.

En esta imagen se ve el switch utilizamos en el sensor

Aquí encontramos el sensor accionado (como se usa en el juego)

Luces:

Sirve para indicar al participante que se encuentra en el lugar correcto para que se disponga a recibir las preguntas a su teléfono celular.

Luz de ejemplo.

Cerradura electromagnética (EM):

Cerradura que trabaja con el principio de un electroimán y se mantiene cerrada mientras recibe energía. Se utiliza para dar acceso a un premio, solo en caso de que el *perfomance* del participante supere un mínimo de puntos luego de transitar por las estaciones de preguntas y respuestas.

Conexión/Funcionamiento:

- El negativo está conectado directamente en la cerradura electromagnética.
- El positivo está conectado al “COM” del relé y el “NC” (normal cerrado) va a la cerradura electromagnética.
- Con esta lógica, el suministro eléctrico es permanente y la cerradura permanece cerrada, hasta que el relé recibe un estímulo a través del puerto GPIO.

Cerradura electromagnética.

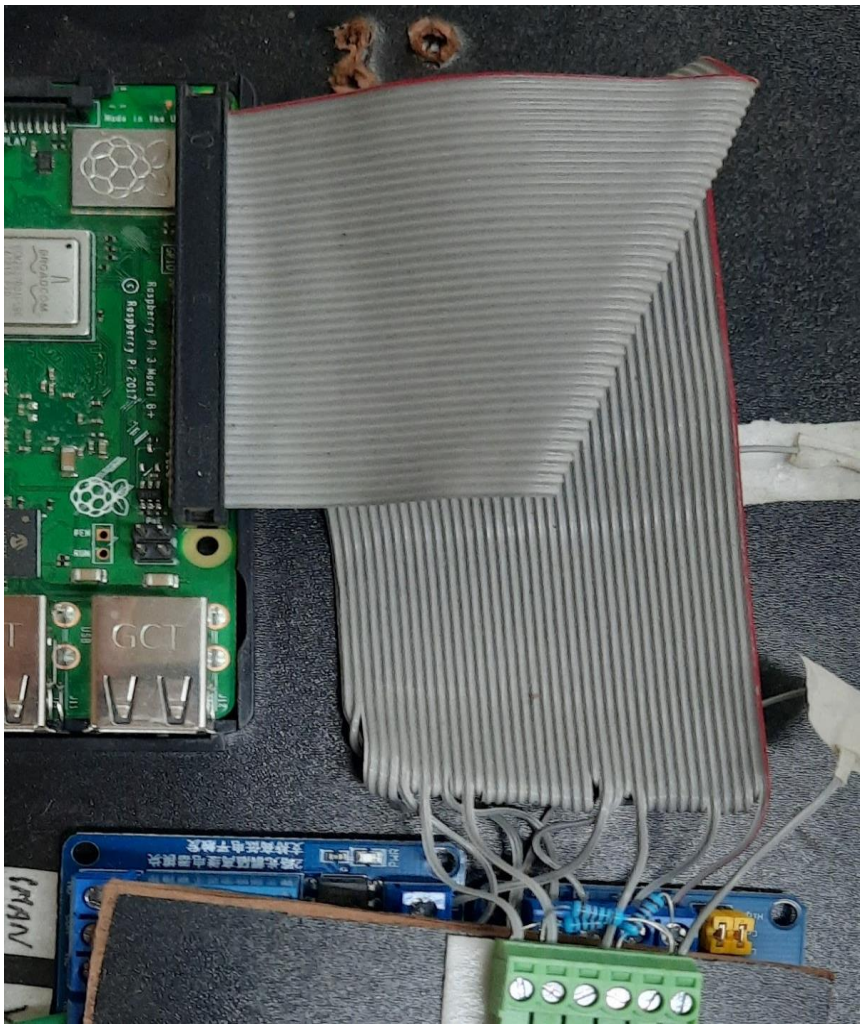
Cableado eléctrico (señales débiles):

El cableado de señales débiles está compuesto por tres partes. Estas son:

1. Borneras-Sensores (entradas)
2. Borneras-Luces/electroimán (salidas)
3. Puerto GPIO Raspberry-Borneras.

Importante:

Los sensores están referenciados mediante una resistencia de 10kOhm a 3.3V, generando un 1 lógico mientras no se encuentren presionados. Cuando detectan una persona, se accionan, cambiando la referencia a GND (-), equivalente a un 0 lógico.



Cableado eléctrico.

Cableado de sensores/luces/cerradura EM:

- **¿Qué tipo de cable utilizamos?**: El tipo de cable que utilizamos es: Cable bipolar paralelo multifilar
- **¿Por qué es bipolar paralelo?**: Se lo llama bipolar a los que tienen dos conductores eléctricos. Se lo nombra paralelo porque tiene dos cables unidos en paralelo
- **¿Qué medida es el cable ?:** La medida es 0,75 mm, pero esto es el diámetro interno (osea el diámetro del cobre).

Cable bipolar paralelo.

Pero, en este caso, utilizamos el cable para poder hacer el cableado de las luces testigo y, a su vez, conectarlo hacia la bornera.

Cableado puerto GPIO:

Para cablear desde el puerto hasta las borneras, reciclamos un cable IDE, utilizado originalmente para conectar el disco duro a la placa base de forma directa.

Cable IDE.

Este cable de 40 pines nos permite conectar la Raspberry (sus puertos GPIO) a las diferentes borneras (sensores y relés).

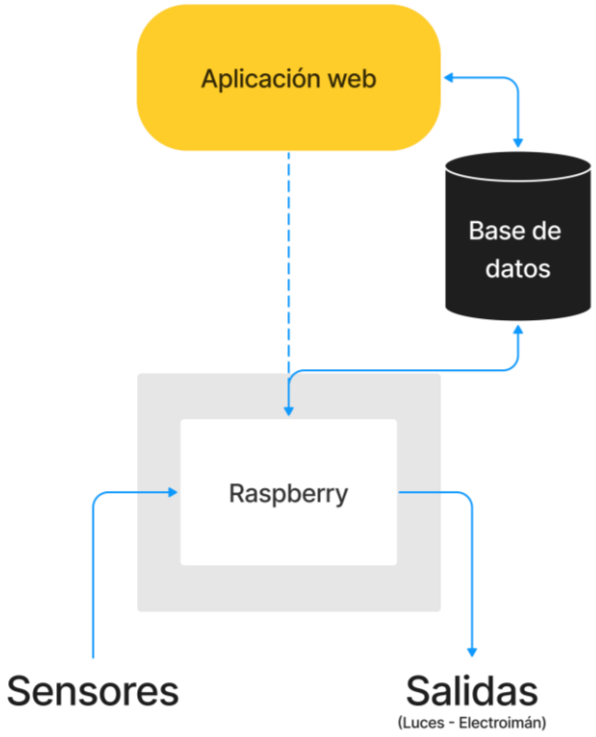
Software

Índice

Introducción:

Descripción del sistema: Se trata de un cuestionario de preguntas. Por medio de la interacción física-virtual, el usuario puede avanzar a través de tres etapas de diferente dificultad. Al finalizarlas, según la puntuación total obtenida, podrá recibir una recompensa.

Esta interacción física, recolectada por medio de sensores, es registrada por la Raspberry en la Base de Datos, y evaluada por la Aplicación web. A su vez, la Raspberry evalúa los cambios en la información de la BD para decidir las acciones a tomar (dar recompensas, apagar luces, entre otras.).



Mapa lógico del sistema.

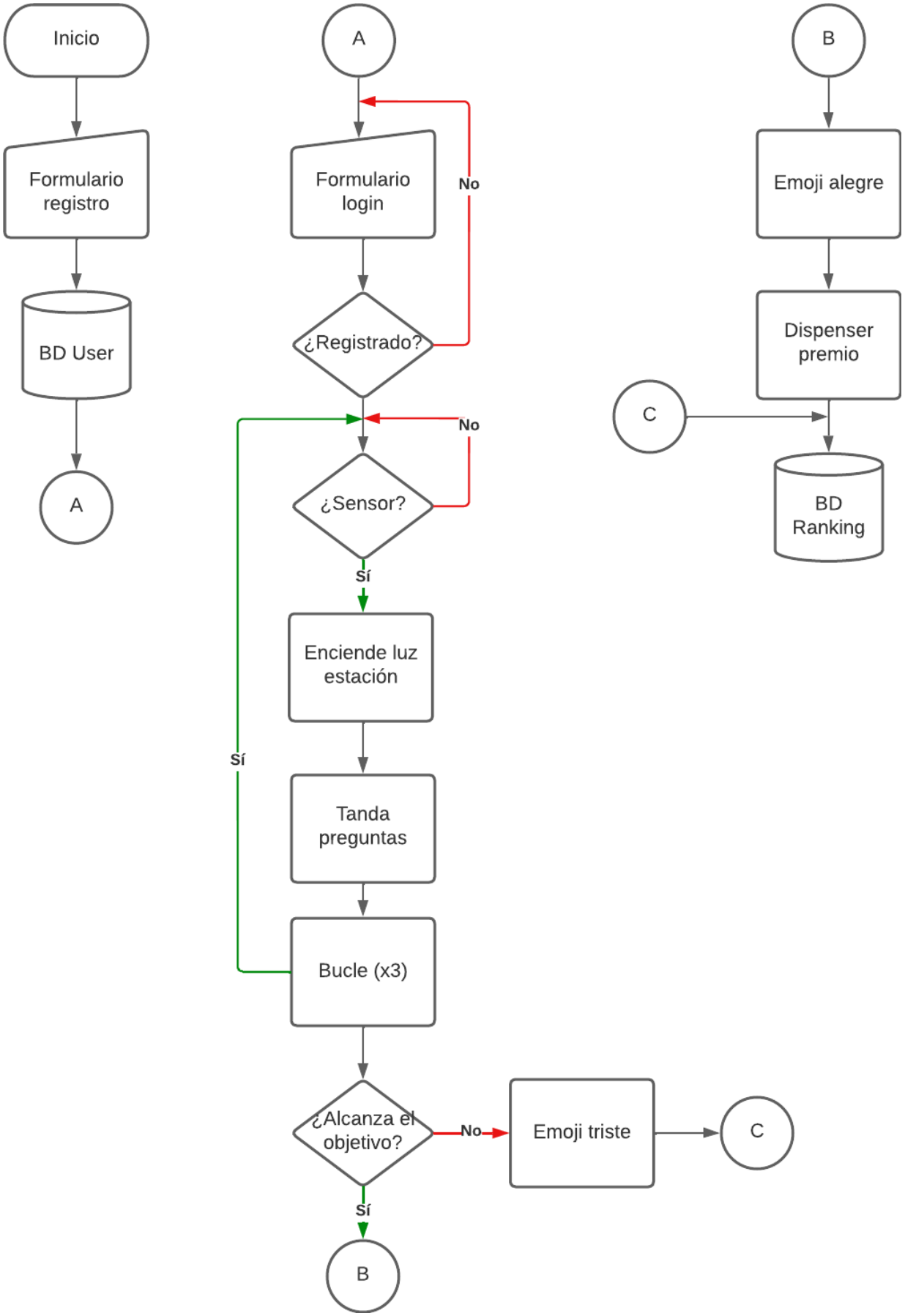


Diagrama de flujo general del sistema.

Necesidad y requisitos: Este proyecto inició a partir de una puesta de ideas para su aplicación práctica y desarrollo en el marco de la materia de Prácticas Profesionalizantes, en el ciclo 2.022. En esta puesta de ideas, desde la materia se enmarcó el criterio de selección en base a la integración de tres áreas fundamentales: Hardware, Software y Redes.

Objetivos: Partiendo de la premisa descripta anteriormente, el proyecto debía cumplir con los siguientes objetivos:

- **Integrar Hardware, Software y Redes.**
- **Interacción virtual/real:** El sistema debe proveer una interfaz que permita la interacción bidireccional entre el mundo físico y el virtual.
- **Espíritu lúdico:** Debe ser una actividad divertida, aunque este no sea su fin último.

Luego de esta introducción al sistema, la documentación se encuentra estructurada, en general, de la siguiente forma:

Arquitectura y diseño del sistema:

- Tecnologías
- Servidor
- Base de datos
    - Tabla
        - Atributos
    - Diagrama de Entidad-Relación
    - Usuarios del SGBD
- Módulos del sistema
    - Módulo
        - Descripción
        - Breve descripción de archivos
    - Diagrama del sistema

Tecnologías:

Para el desarrollo de la aplicación web, se estableció un servidor utilizando la estructura WAMP, es decir, un servidor Apache en una computadora con sistema operativo Windows, una base de datos MariaDB y PHP (WAMP=Windows, Apache, MariaDB, PHP).

- **XAMPP:** Un conjunto de soluciones de servidor web gratuitas, multiplataforma, de código abierto y libre (licencia GNU GPL). Algunas de las tecnologías que incluye, y son implementadas en el proyecto, son:
    - **Servidor HTTP Apache:** Se trata de un servidor web HTTP de código abierto y libre (Licencia Apache 2.0, no **copyleft**).
    - **MariaDB Server:** Un Sistema de Gestión de Base de Datos (SGBD, o DBMS por sus siglas en inglés) relacionales, de licencia GPLv2 (libre y de código abierto, a excepción de ciertos componentes) y derivado de MySQL.
        - **phpMyAdmin:** Herramienta de administración GUI para MySQL y MariaDB, en formato de aplicación web, bajo licencia GPLv2.
    - **PHP:** Lenguaje de programación interpretado de uso general, **procesado por el servidor**. Libre y de código abierto (no copyleft, bajo licencia PHP). Posee múltiples librerías incorporadas que, en este proyecto permiten: acceder y operar con el SGBD, generación aleatoria, funciones hash, entre otras.
- **Cliente de MariaDB:** Es la herramienta que el usuario de la BD puede utilizar para establecer una conexión e interactuar con el servidor de MariaDB desde el lado del cliente.
- **Tecnologías web básicas (HTML, CSS y JavaScript):** HTML o HyperText Markup Language (Lenguaje de Marcas de Hipertexto, en español), es aquella encargada de la interconexión y estructura fundamental de las páginas web, CSS o Cascading Style Sheets (Hojas de Estilo en Cascada) permite dar diseño a la página web y, JavaScript es normalmente utilizado para definir las interacciones y funciones concretas implementadas para permitir el contenido dinámico.
- **PuTTY:** Un cliente con múltiples protocolos para el acceso remoto, gratuito y de código abierto (bajo licencia MIT). En específico se utiliza “plink.exe” (PuTTY Link), un ejecutable que contiene la herramienta de conexión de línea de comandos, prescindiendo del resto de componentes de PuTTY.

Evaluación de tecnologías:

Las tecnologías utilizadas fueron en gran medida heredadas de los proyectos anteriores realizados en la cátedra, a excepción del Cliente de MariaDB y PuTTY.

Servidor:

El servidor HTTP Apache provee las funcionalidades de servidor, y permite el acceso al SGBD.

Se accede a través de la dirección IP 10.0.1.40. Por defecto, el puerto de acceso es el número 80.

Base de datos:

La base de datos del sistema es denominada “feria-db". Fue creada utilizando el SGBD (Sistema de Gestión de Base de Datos) MariaDB.

La base de datos no solo representa la locación donde es almacenada la información, también representa el punto de unión o interfaz en donde son conectadas la aplicación web y la Raspberry.

Tablas:

Consta de 5 tablas:

- **users**

  Tabla correspondiente a los usuarios que se registraron en el juego. Consta de 4 columnas: id-user, user-email, user-age y user-nick.

  Columnas:

  - id-user: Código de identificación único para cada usuario.
  - user-email: Dirección de correo electrónico del usuario.
  - user-age: Edad del usuario.
  - user-nick: Nick ingresado por el usuario.

- **journey**

Contiene información sobre el progreso actual de cada usuario. Consta de 5 columnas: id-user, journey-started, journey-step, journey-stage y journey-ended.

Columnas:

- id-user: Código de identificación único para cada usuario.
- journey-started: Hora de iniciado el recorrido.
- journey-step: Punto del trayecto en el circuito de preguntas (0-9).
- journey-stage: Número de etapa del circuito en donde se encuentra el usuario.
- journey-ended: Hora de finalizado el recorrido.

- **rankingmayores**

  Es una tabla de clasificación donde se encuentran únicamente los usuarios registrados como mayores de 15 años. Consta de 3 columnas: id-user, ranking-score y ranking-et.

  Columnas:

  - id-user: Código de identificación único para cada usuario.
  - ranking-score: Puntaje total en el juego.
  - ranking-et: Tiempo transcurrido (*elapsed time*, o et) total en el juego.

- **rankingmenores**

  Es una tabla de clasificación donde se encuentran los usuarios registrados como menores de 15 años. Consta de 3 columnas: id-user, ranking-score y ranking-et.

  Columnas:

  - id-user: Código de identificación único para cada usuario.
  - ranking-score: Puntaje total en el juego.
  - ranking-et: Tiempo transcurrido (*elapsed time*, o et) total en el juego.

- **signals**

  Es una tabla que registra las señales recibidas de los sensores. Contiene 3 columnas: id-record, signal-stage y signal-time.

  Columnas:

  - id-record: Código de registro único.
  - signal-stage: Número de sensor o etapa del circuito en donde se registró actividad.
  - signal-time: Fecha y hora de la creación del registro.

Diagrama de Entidad-Relación:

El siguiente es el diagrama de Entidad-Relación de la base de datos definida anteriormente:

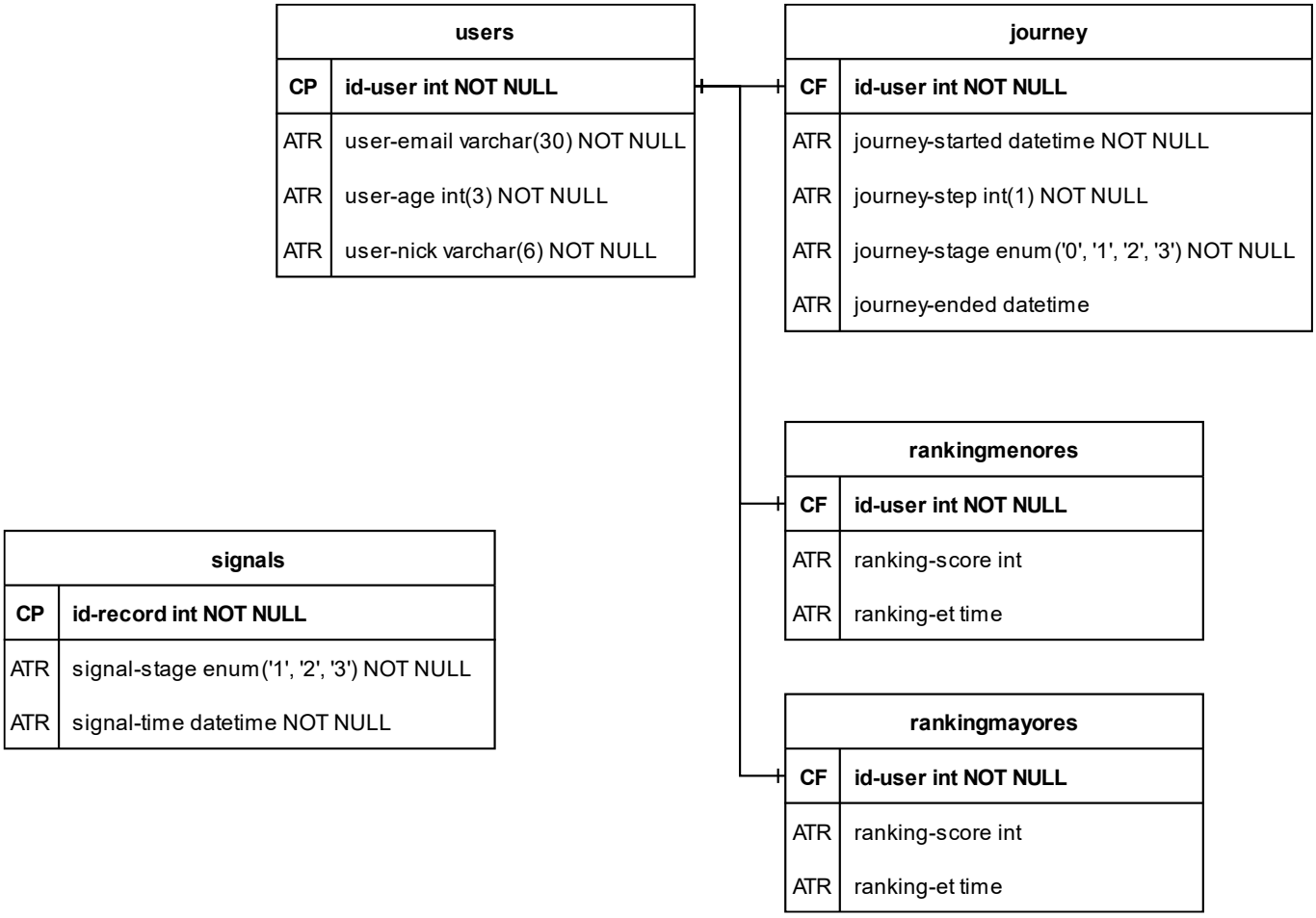


Diagrama de Entidad-Relación de feria-db.

Usuarios del SGBD:

Para el acceso al SGBD, fueron creados dos usuarios, diferenciados por sus permisos y por su dirección de acceso:

Usuarios del SGBD		
Nombre de usuario	**tester1**	**raspberry**
Nombre del servidor	localhost (local)	% (Cualquier servidor)
Contraseña	tester1	-r4spb3rry-
Privilegios globales (solo Datos)	SELECT, INSERT, UPDATE, DELETE, FILE	SELECT, INSERT

Usuarios del Sistema de Gestión de Bases de Datos creados.

Módulos de la aplicación:

El siguiente apartado describe la funcionalidad de cada módulo de la aplicación, así como una breve descripción de los archivos propios de estos.

Pero existen ciertos archivos que contienen funcionalidad aisladas, que pueden ser reutilizadas. Estos se encuentran en los directorios “*/header*”, y ya que son reutilizados, son descriptos a continuación de forma general.

Archivos:

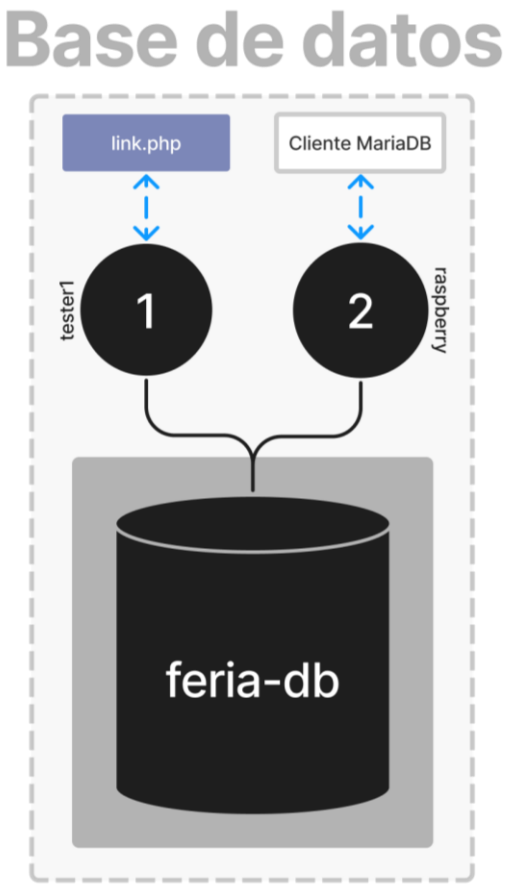
- **header/emoji.php:** Una función que decide el emoji que se mostrará según la cantidad de puntos.
- **header/link.php:** Contiene una función para conectarse a la base de datos, utilizando las credenciales del usuario (creado para editar los datos de las tablas de la BD).

De la misma forma, los archivos (CSSs) que contienen plantillas generales para el diseño de las páginas web, son descriptos a continuación:

Archivos:

- **normalize.css:** Como su nombre indica, establece valores por defecto para facilitar el diseño de las páginas.
- **messages.css:** Establece una plantilla general para los mensajes de error o notificación en las páginas.
- **fonts.css:** Contiene todas las fuentes de texto del proyecto, que se encuentran almacenadas localmente. Permite utilizarlas en otros archivos.

Por último, el acceso a la Base de datos se realiza por medio de dos procesos: link.php y el Cliente MariaDB, que utilizan el usuario tester1 y raspberry, respectivamente:



Accesos a la Base de Datos.

Acceso:

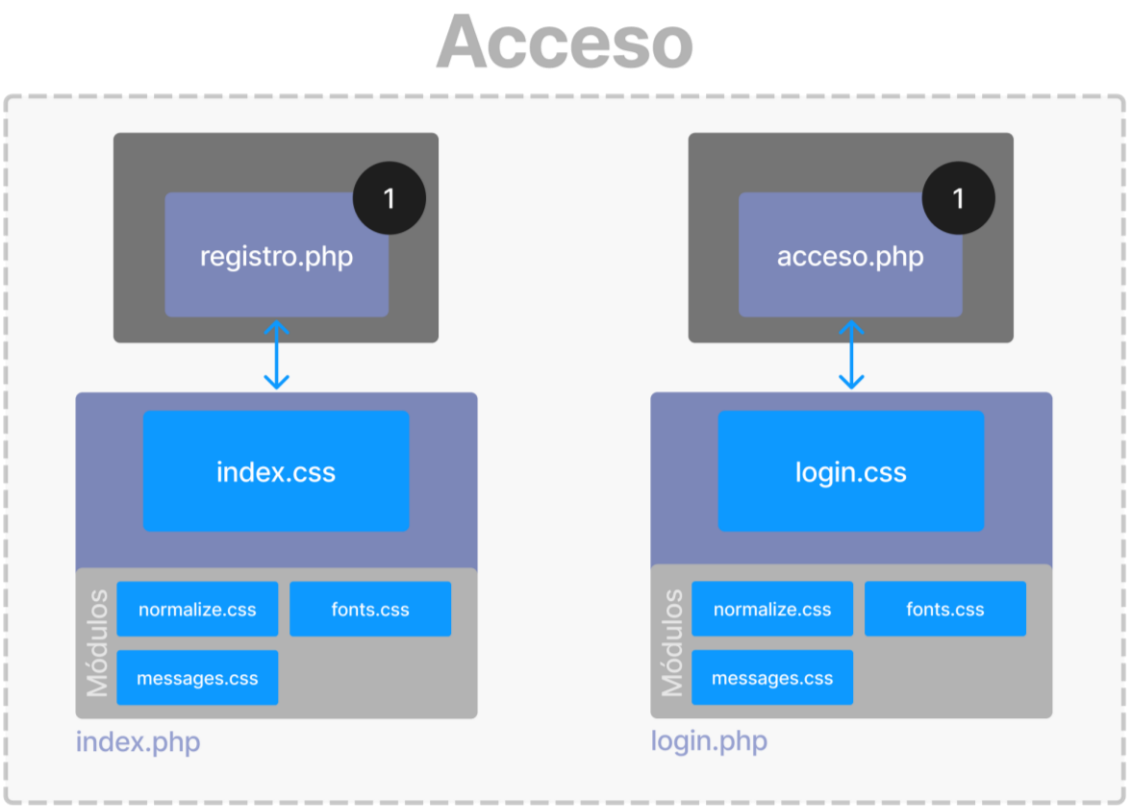


Diagrama de módulo Acceso.

Consta de 4 (cuatro) páginas: index.php, login.php, registro.php y acceso.php.

index.php se debe encontrar en un tablero aparte en el que cualquiera se pueda inscribir sin importar si tiene que esperar antes de jugar. Una vez llenado el formulario, este será enviado al servidor a través de la página registro.php.

login.php consta de un formulario en el que se debe escribir el nombre de usuario creado anteriormente para poder empezar a jugar. Este formulario es enviado a la página acceso.php donde se valida el usuario y, si ya está inscripto, le deja jugar. Si ya ha completado el juego, es redireccionado al ranking.

Archivos:

- **index.php:** Formulario donde se inscriben los usuarios. Tras pasar por la correspondiente validación, el usuario es devuelto a esta página.
- **login.php:** En esta página el usuario ingresa su nombre de usuario (su **nick**, escrito en index.php) para comenzar a jugar.

- **registro.php:** Permite registrar a los usuarios en la base de datos diferenciando según su edad y evitando posibles repeticiones.
- **acceso.php:** Obtiene las variables del formulario HTML, revisa si existe ese usuario en la base de datos, muestra un mensaje en caso de que no exista, lo redirige al ranking si ya jugó, sino sigue con verificar si es mayor o menor de 15 años. Luego, crea las variables que se utilizarán a lo largo del juego, y redirecciona a intermedio-page.php.
- **index.css:** Contiene el diseño gráfico para los elementos en index.php y login.php.

Q&A:

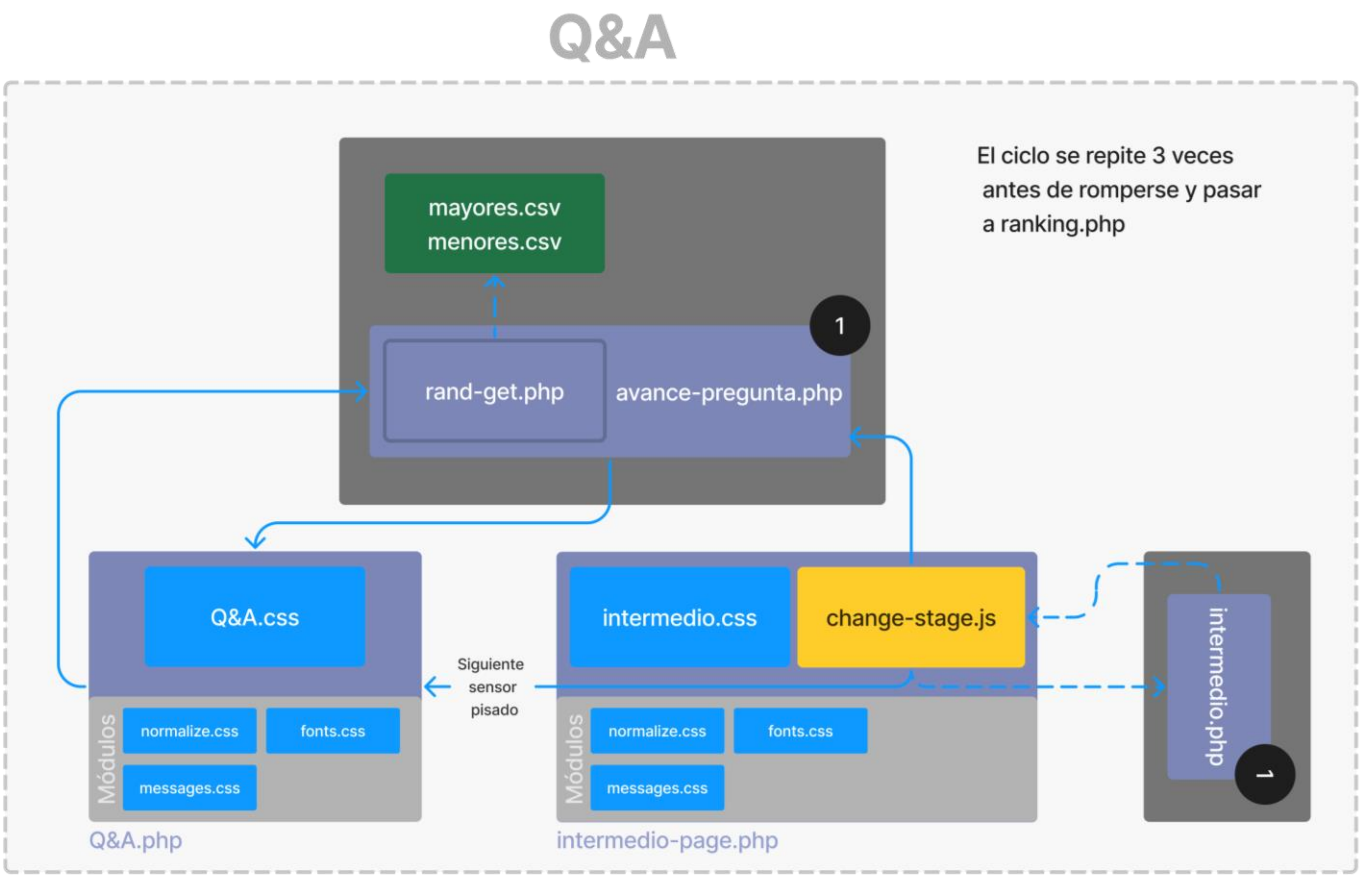


Diagrama de módulo Q&A.

La página acceso.php envía al usuario a la página intermedio-page.php , donde change-stage.js (por medio de ejecutar intermedio.php de forma asíncrona) se espera a que la etapa corresponda al número de pregunta actual, en otras palabras, que se encuentre presionado el sensor perteneciente a la siguiente etapa.

Una vez pisado el primer sensor, el usuario es redirigido a la página avance-pregunta.php , donde es seleccionada la pregunta de forma aleatoria y son almacenadas sus respuestas, entre otras validaciones. Luego, es redirigido a la página Q&A.php , una plantilla donde se presentan las preguntas y respuestas. Tras responder la pregunta, la respuesta es verificada en la página avance-pregunta.php, si es correcta, se calcula el puntaje, se registra el tema de la pregunta para que luego no se repita (banlist), se aumenta la racha y el contador de preguntas respondidas, se selecciona la nueva pregunta y se almacenan sus respuestas, y es redirigido a Q&A.php . El proceso descripto se repite tres veces (por cada pregunta), y entonces, se reinicia la banlist y es redireccionado a intermedio-page.php, donde deberá ubicarse sobre la próxima plataforma para continuar el juego (será direccionado a avance-pregunta.php).

Si el contador de preguntas respondidas es 0, 3 o 6 y la última etapa registrada no es la correspondiente (1ra, 2da y 3ra respectivamente), la página avance-pregunta.php redirigirá al usuario a la página intermedio-page.php hasta que estas correspondan.

Si el contador de preguntas respondidas es 9, o mayor, el usuario será redirigido a la página ranking.php.

Archivos:

- **Q&A.php:** Es una plantilla que se completa con la pregunta, respuesta y progreso en el juego del usuario. Según el tópico, la imagen de fondo cambia dinámicamente. Se utiliza la misma plantilla en todas las preguntas.
- **intermedio-page.php:** Página hecha para esperar a que el usuario pise el próximo sensor antes de pasar a la próxima pregunta, de la siguiente etapa.
- **avance-pregunta.php:** Se realizan las siguientes validaciones en orden:
    - Si la pregunta fue respondida, aumenta el contador de preguntas respondidas y actualiza el progreso en la base de datos.
    - Si la pregunta fue respondida correctamente, se aumenta la racha y se suman los puntos correspondientes. Devuelve la racha a 0 si la pregunta no fue correctamente respondida.
    - Registra el momento de finalización si ya respondió al menos 9 preguntas, esta redundancia, es en caso de que llegue a responder más de 9 veces por alguna razón. Luego recupera el momento de inicio, el cual compara con el de finalización, para obtener el tiempo transcurrido. Por último, se actualiza dicho tiempo transcurrido en el registro, para entonces enviar al usuario a la página ranking.php .
    - Selecciona una nueva pregunta sólo si la anterior fue respondida.
    - Compara el paso con la etapa actual para redireccionar al usuario a intermedio-page.php , si estos no corresponden.
- **rand-get.php:** Permite obtener una pregunta de forma aleatoria (pseudo-aleatoria), seleccionando el archivo CSV (mayores.csv o menores.csv) a utilizar según el rango de edad, evitando las temáticas anteriores por etapa.

- **intermedio.php:** Actualiza la etapa del usuario en la base de datos cuando pisa el siguiente sensor.
- **change-stage.js:** Ejecuta intermedio.php de forma asíncrona en intervalos de 6 segundos (6.000 milisegundos), y evalúa y redirecciona según los resultados obtenidos.
- **mayores.csv:** Tabla con los tópicos, sub-tópicos, preguntas, respuestas y número de respuesta correcta. Corresponde a los usuarios mayores de 15 años.
- **menores.csv:** Tabla con los tópicos, sub-tópicos, preguntas, respuestas y número de respuesta correcta. Corresponde a los usuarios menores de 15 años.
- **Q&A.css:** Contiene el diseño gráfico para los elementos en Q&A.php.
- **intermedio.css:** Contiene el diseño gráfico para los elementos en intermedio-page.php.

Ranking:

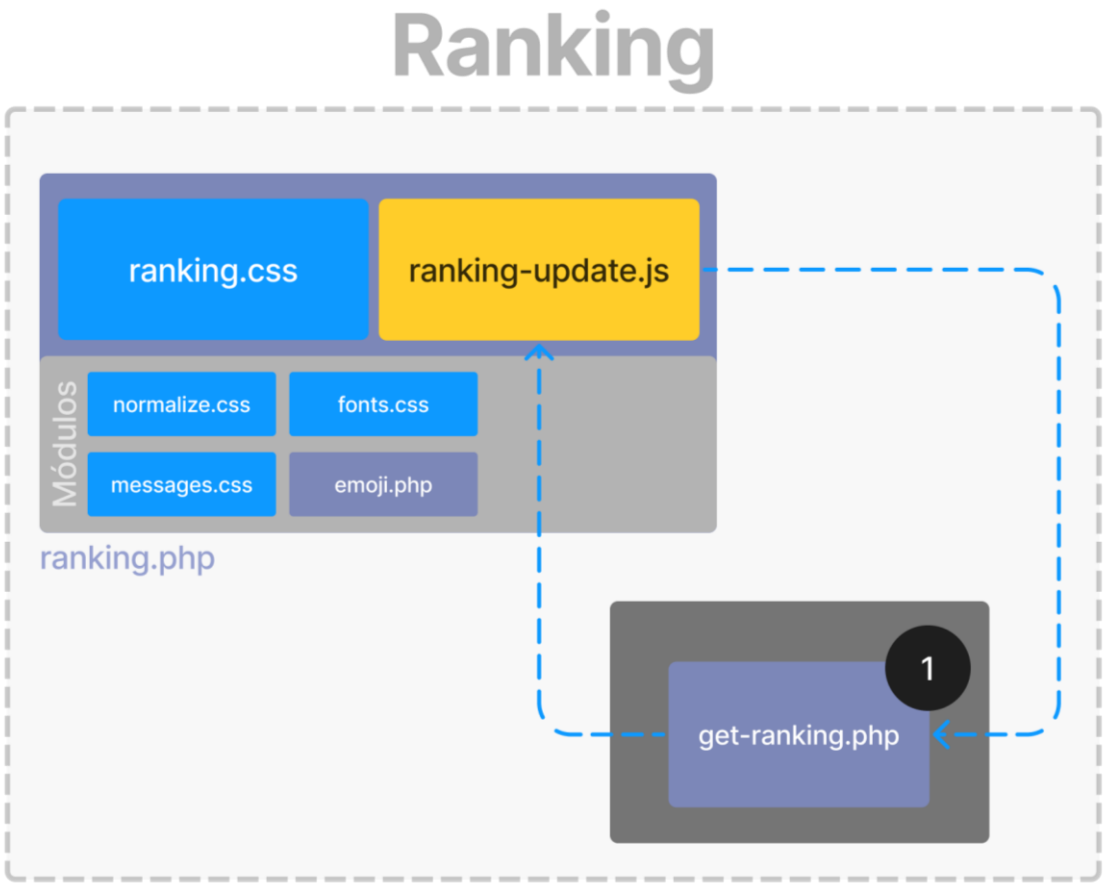


Diagrama de módulo Ranking.

En la página ranking.php, si el usuario llega desde avance-pregunta.php, o redireccionado desde acceso.php, se mostrará el puntaje del usuario, su tiempo y una tabla donde podrá comparar su puntaje con el de otras personas. Si por el contrario se ingresa de cualquier otra forma, sólo se mostrará la tabla de puntuaciones.

Archivos:

- **ranking.php:** En esta página el usuario podrá ver su puntaje, el tiempo que le tomó responder todo y/o una tabla donde podrá ver su puntaje comparado al resto de usuarios. Incluye un buscador por nombre de usuario.
- **get-ranking.php:** Evalúa la edad del último usuario que jugó, y según esta recupera la tabla de puntuaciones correspondiente. Se utiliza en ranking-update.js.
- **ranking-update.js:** Ejecuta get-ranking.php de forma asíncrona en intervalos de 10 segundos (10.000 milisegundos), y actualiza la tabla en ranking.php con la nueva información. Contiene la función utilizad para la búsqueda de nick en la tabla.
- **ranking.css:** Contiene el diseño gráfico para los elementos en ranking.php.

Progresión del jugador:

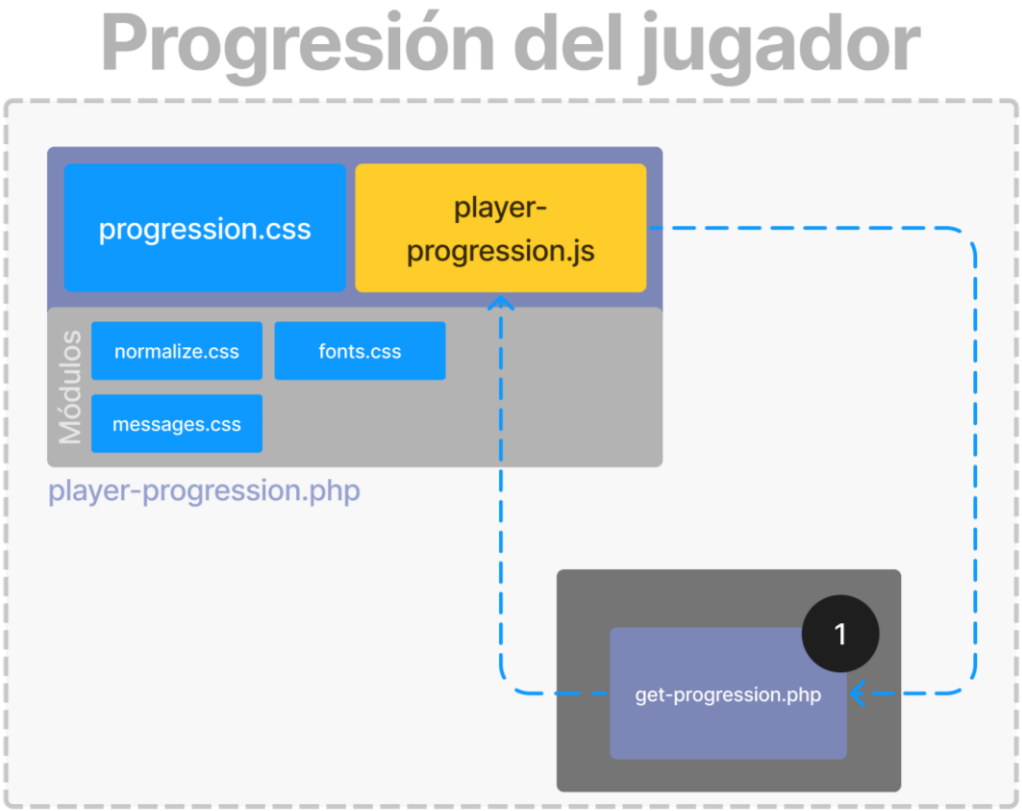


Diagrama de módulo Progresión del jugador.

Conforma un panel para la visualización pública del progreso y puntaje del jugador actual. Se accede por medio de player-progression.php, y se actualiza de forma constante por medio de player-progression.js (que ejecuta get-progression.php para obtener la información).

Archivos:

- **player-progression.php:** Es una pantalla que permite a los espectadores observar la información (id) y progreso (paso y puntaje) del jugador actual.
- **get-progression.php:** Toma la edad del jugador al que pertenece el último registro generado en journey , y recupera su id, paso y puntaje actual.
- **player-progression.js:** Ejecuta get-progression.php de forma asíncrona en intervalos de 3 segundos (3.000 milisegundos), y actualiza la visualización de progresión del jugador, como su información, en player-progression.php con la nueva información.
- **progression.css:** Contiene el diseño gráfico para los elementos en player-progression.php.

Admin (Control Remoto Electroimán):

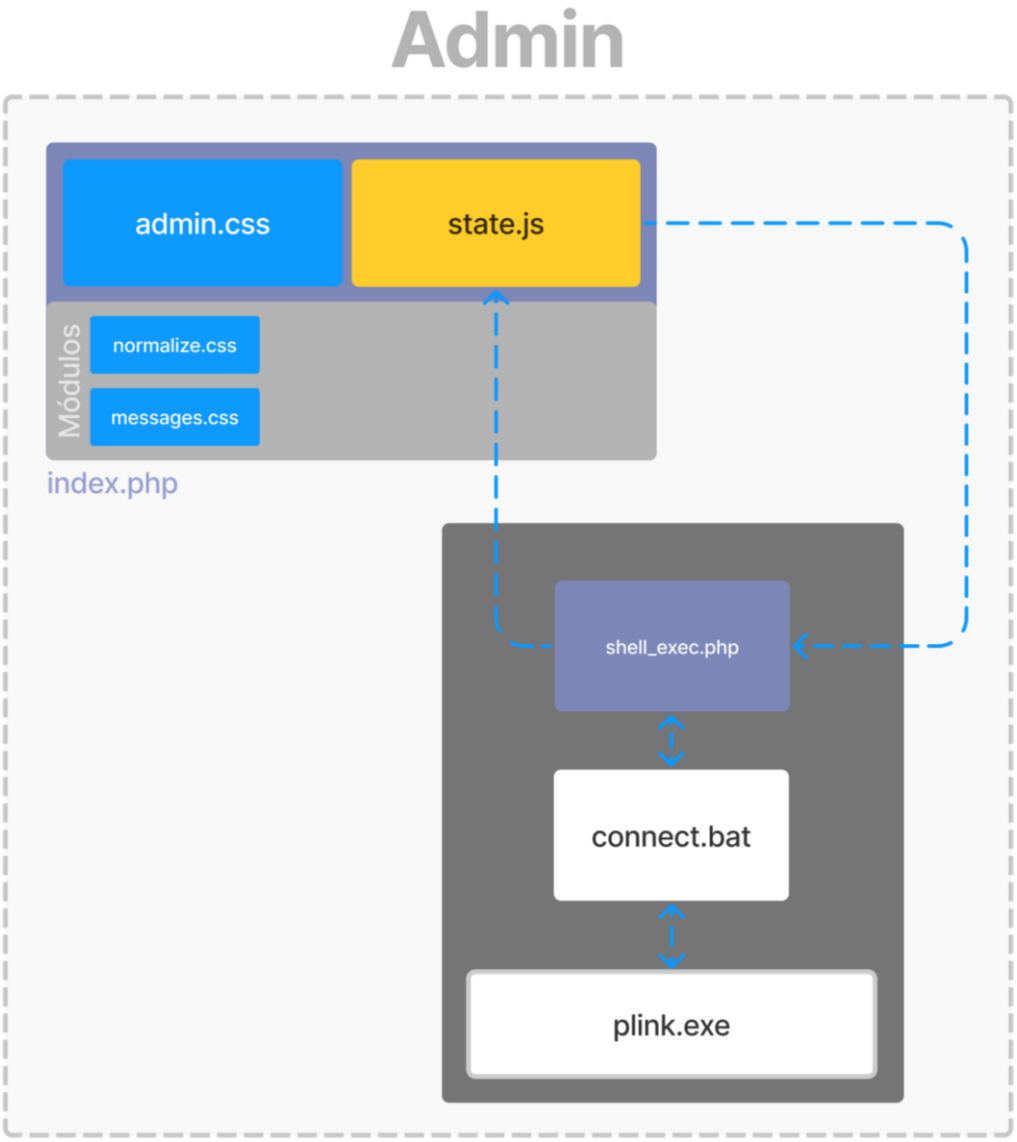


Diagrama de módulo Admin.

Permite desactivar el electroimán por un periodo de 3 segundos de forma remota. Consta de un botón en index.php, por el cual se inicia state.js (que ejecuta shell_exec.php), lo que desactivará el electroimán por 3 segundos. shell_exec.php, a su vez, inicia connect.bat por medio del CMD de Windows, el cual se conecta de forma remota a la Raspberry utilizando plink.exe, y activa raspberry-open.sh para desactivar el electroimán. Según el éxito o fracaso de la ejecución del CMD en shell_exec.ph, se mostrarán mensajes correspondientes (en index.php).

El control remoto del electroimán posee una raíz (carpeta de origen) diferente a la aplicación web. Se encuentra en “*IP/admin/*” (a diferencia del resto del proyecto, en “*IP/web/*”).

Archivos:

- **index.php:** Interfaz gráfica con un botón para activa state.js.

- **state.js:** Ejecuta shell_exec.php de forma asíncrona, evalúa y provee mensajes según el éxito o fracaso de la ejecución (de shell_exec.php).
- **shell_exec.php:** Ejecuta por medio del CMD de Windows connect.bat. Informa en caso de éxito o fallo de la operación.
- **connect.bat:** Ejecuta plink.exe para acceder a la Rasberry utilizando el protocolo SSH, utilizando su usuario y contraseña, y ejecutando raspberry-open.sh de forma remota.
- **plink.exe:** Componente de PuTTY que contiene las funciones para establecer conexiones remotas por medio de CLI de Windows (CMD).
- **admin.css:** Contiene el diseño gráfico para los elementos en index.php del módulo Admin.

Raspberry:

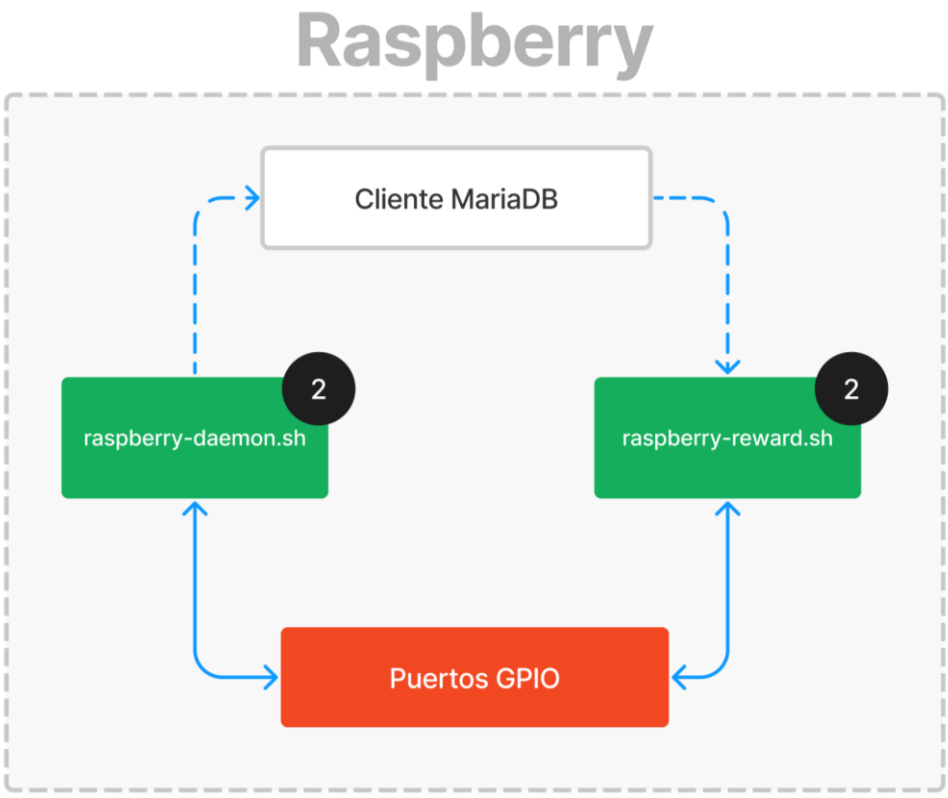


Diagrama de módulo Raspberry.

La lectura y escritura de los estados de los pines fue realizada utilizando un shell de Linux, BASH, más específicamente, en dos programas llamados raspberry-daemon.sh y raspberry-reward.php. Para las operaciones de lectura y escritura de la base de datos, fue utilizado el cliente de MariaDB.

Ambos scripts cuentan con dos modos:

- **Modo 1:** Permite operar de forma normal con la Raspberry.
- **Modo 2:** Permite realizar pruebas de depuración prescindiendo de la Raspberry.

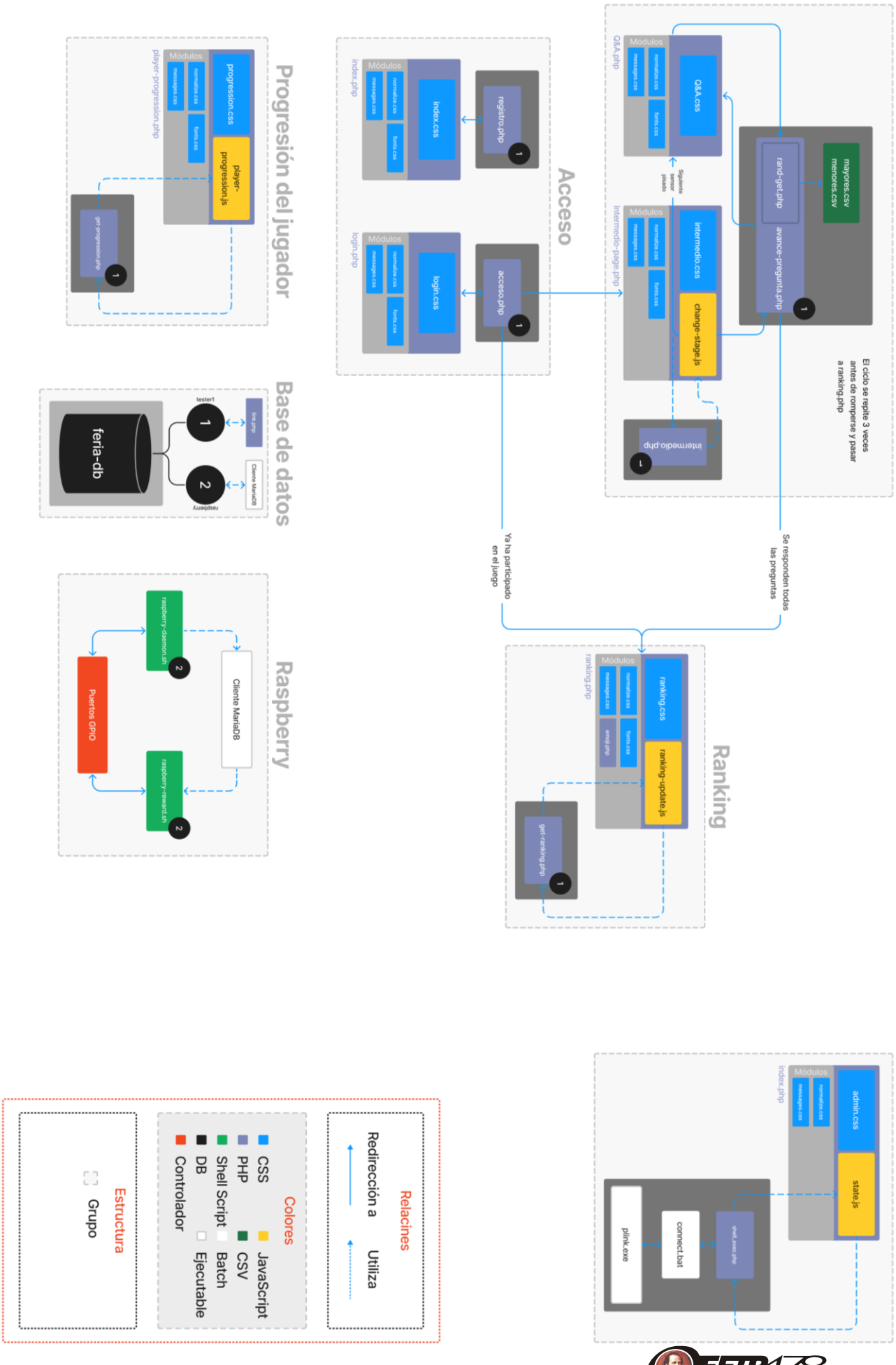
Este último es denominado modo de solo terminal, y fue implementado para posibilitar la simulación de los valores obtenidos o acciones ejecutadas normalmente por la Raspberry. Este modo cuenta con:

- Configuración de la dirección IP del servidor donde se encuentra la BD.
- Mensajes describiendo el accionar normal de la Raspberry según la situación.

Archivos:

- **raspberry-daemon.sh:** Es el programa que permite registrar el cambio de las entradas (las plataformas) en la base de datos “feria-db” y activar los dispositivos (luces testigo y electroimán).
    - **Modo de solo terminal:** El ingreso de entradas físicas es remplazado por una serie de comandos, que permiten disparar los eventos de creación de entradas en la base de datos. Estos son:
        - **1:** Registra la llegada a la etapa 1.
        - **2:** Registra la llegada a la etapa 2.
        - **3:** Registra la llegada a la etapa 3.
- **raspberry-reward.php:** Suelta automáticamente una bolsa de caramelos si el jugador actual alcanzo o superó la puntación necesaria.
    - **Modo de solo terminal:** Solo es necesario acceder. El programa se comportará de forma idéntica, con la excepción de que carece de las acciones ejecutadas en las salidas de la Raspberry, y que permite configurar la dirección IP del servidor.

Diagrama del sistema:



Para más información ver Anexo I: Descripción de archivos.

Redes

Índice

Introducción:

La red permite establecer la vía de comunicación entre los diferentes dispositivos que intervienen en el sistema: permite a los teléfonos celulares acceder a la aplicación de forma remota, a la Raspberry efectuar consultas en la Base de Datos y, al servidor, proveer su servicio a todos los dispositivos conectados a la red, establecida por medio del router.

Listado de dispositivos de la red:

- **Servidor**
  - IP del servidor: 10.0.1.40/24 - fija
- **Raspberry**
- **Teléfono/s**
  - IP del teléfono: 10.0.1.3/24 - 10.0.1.103/24 (cualquier dirección IP dentro del pool del router) – dinámica
- **Router**
  - IP de la red: 10.0.1.0/24
  - IP del router: 10.0.1.1/24 - fija

Dispositivos de la red:

Router:

Para la red utilizamos un router o enrutador hogareño, al cual le asignamos la dirección IP 10.0.1.1/24 y un pool de 101 usuarios a su DHCP. Su SSID es: El Pasillo Del Saber.

Servidor:

El servidor es una computadora con sistema operativo Windows y aplicación XAMPP. Se encarga tanto del procesamiento *backend*, como de la Base de Datos.

Raspberry:

La Raspberry en este caso se encarga del procesamiento de las señales y de enviar los datos a la BD.

Teléfono:

El teléfono fue prestado por los alumnos encargados de atender el puesto, este se conectó por Wi-Fi a la red del router para acceder a la página web. El dispositivo sólo requiere tener un navegador (preferiblemente Google Chrome) y acceso a red Wi-Fi.

Router:

En este diagrama, y en la realidad, nuestro enrutador se encontró desconectado de internet. Por un lado, para evitar el uso de páginas externas, y por otro, para darle autonomía al proyecto y poder presentarlo en varios lugares sin problema ni necesidad de conexión.

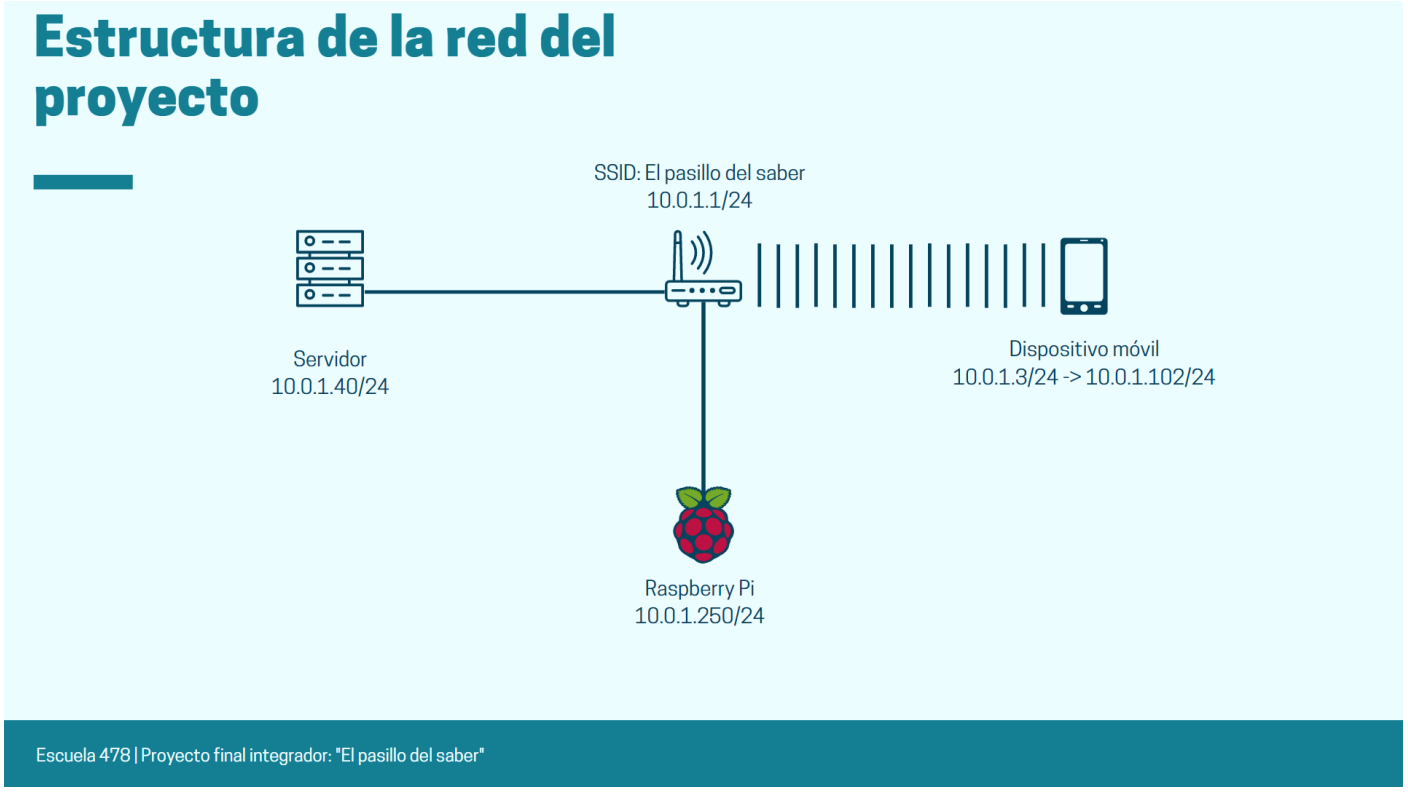


Diagrama de la red.

Posibles aplicaciones

de las tecnologías desarrolladas

Posibles aplicaciones:

A partir de la aplicación del proyecto, fue posible visualizar nuevas aplicaciones para las tecnologías desarrolladas:

- **Museo inteligente:** Utilizando las plataformas (interruptores) que registran la interacción física del usuario, sería posible distribuir los elementos de forma de posicionarlas delante de objetos importantes o representaciones históricas. Podrían iniciar el recorrido del museo, en el que, al posicionarse sobre una plataforma, inicie una narración sobre la historia de los objetos, y finalizarla, sea necesario deslazarse a la próxima para continuar.

Anexo I

Descripción de archivos

Índice

Descripción de archivos:

La siguiente es una descripción detallada archivos que contienen funciones del programa. Por lo tanto, archivos de diseño (como los CSS), quedan excluidos de este detalle. Las descripciones, siguen la siguiente estructura:

- Descripción de archivos
    - Tipo de archivo
        - Archivo: Descripción de archivo a detalle.

PHP:

acceso.php:

Al comenzar a leer el archivo, podemos ver que iniciamos una sesión, luego tomamos el nombre de usuario escrito en el formulario y lo guardamos en un formulario. Al final de la imagen se añada con `include` el código de una página con el fin de establecer las conexiones necesarias a la base de datos.

```php
<?php
  session_start();

  #Obtener variables desde el formulario en HTML, con el método POST.
  $usuario = $_POST['Usuario'];//$email = $_POST['email'];

  #Realizar, comprobar y almacenar credenciales de la conexión a la DBMS.
  #-------------------------------------------------------------------------
  include 'header\link.php';
  $link = connect();
```

Luego de esto, se realiza una validación de cuenta para verificar que el nombre de usuario insertado exista en la BD. Para esto, se crean 3 variables: `$select`, `$result` y `$validation`. La primera contiene la petición a la base de datos de un registro con el cual el usuario insertado corresponda, la segunda guarda la respuesta de la base de datos, y la tercera almacena el número de filas (o registros) devueltos por la base de datos.

En caso de que la validación falle (`$validation == 0`), se devolverá un mensaje de error, y será redireccionado a `login.php`.

```php
#Validación de cuenta.
  #-------------------------------------------------------------------------
```

```
  $select = "SELECT*FROM users WHERE `nick` = '$usuario'";//$select = "SELECT*FROM users
WHERE `user-email`='$email'";
  $result = mysqli_query($link, $select);
  $validation = mysqli_num_rows($result);


  if ($validation == 0) {
    #Notificar que primero se debe crear un usuario en el registro.
    $_SESSION['index_message'] = "Cuenta no registrada. Por favor, primero dirígete a la
terminal de registro.";
    header("Location: ../../web/login.php");
    die();
  }
```

Ahora creamos tres variables, las cuales realizan las siguientes acciones en orden: buscar la información de la petición anterior y almacenar de este registro el id y la edad del usuario. Una vez hecho esto, se comprueba si la persona ya ha participado preparando otra petición para buscar al mismo usuario en la tabla `journey` según su id, con tal de almacenar el resultado y el número de filas devueltas (si devuelve 0, significa que no hay registro de este usuario). Si no hay registro del usuario aún, se insertará en la tabla antes mencionada el identificador de usuario y luego se corroborará dicho registro.

Ahora que no hay duda de que existe un registro de este usuario, se verifica que no haya completado el juego aún, pues en el caso contrario, se lo enviará a la página `ranking.php`.

```
$user = mysqli_fetch_assoc($result);
  $id = $user['id-user'];
  $edad = $user['user-age'];

#-------------------------------------------------------------------------------------
------

  #Comprobar si la persona ya ha participado. Si es así, redirigirlo al ranking.
  #-----------------------------------------------------------------------------
  $select = "SELECT*FROM journey WHERE `id-user`='$id'";
  $result = mysqli_query($link, $select);
  $validation = mysqli_num_rows($result);

  if ($validation == 0) {
    #Insertar nuevo registro de progreso.
    $insert = "INSERT INTO journey (`id-user`) VALUES ('$id')";
    mysqli_query($link, $insert);
    $result = mysqli_query($link, $select);
  }

  $journey = mysqli_fetch_assoc($result);
```

```
  if ($journey['journey-step'] >= 9) {
    header("Location: ../../web/php/ranking.php");
    die();
  }
```

Con el siguiente código, se selecciona el nombre de la tabla clasificatoria correspondiente a la edad registrada del usuario.

```
if($edad < 15){
    $edad = "rankingmenores";
  }elseif($edad >= 15){
    $edad = "rankingmayores";
  }
```

Se valida que el usuario se haya creado en la tabla clasificatoria antes seleccionada. En caso de no lograrse, se mostrará un mensaje de error que recomienda hablar con un administrador.

```
#Validación del nick.
  $select = "SELECT*FROM $edad WHERE `id-user`='$id'";
  $result = mysqli_query($link, $select);
  $validation = mysqli_num_rows($result);

  if ($validation == 0) {
    #Notificar que hubo un error en el registro del nick.
    $_SESSION['index_message'] = "Nick de cuenta no creado. Por favor, contactarse con
un administrador.";
    header("Location: ../../web/");
    die();
  }
```

Con las siguientes líneas de código, se guardará en variables de sesión los datos de id y edad, necesarios para validar el usuario y que el juego funcione.

```
#Registro de variables del usuario.
  #-------------------------------------------------------------------------
  $_SESSION['usuario']['id'] = $user['id-user'];
  $_SESSION['usuario']['edad'] = $edad;
```

Ahora se define el paso del usuario, la etapa en la que se encuentra, la racha de preguntas y su puntaje, como ramas del arreglo multidimensional `$_SESSION` (una variable *superglobal*, por poder utilizarse en diferentes páginas). Por último, es redireccionado a la página intermedia de etapas (`intermedio-page.php`) para iniciar el recorrido.

```
#Registro de variables de sesión del usuario.
  #-------------------------------------------------------------------------
```

```
  $_SESSION['juego']['paso'] = $journey['journey-step'];
  $_SESSION['juego']['etapa'] = $journey['journey-stage'];
  $_SESSION['juego']['racha'] = 0;

  #Registro de variables de sesión del usuario.
  #-------------------------------------------------------------------------
  $_SESSION['tabla']['puntaje'] = 0;

  #Redirección para iniciar el desafío.
  #-------------------------------------------------------------------------
  header("Location: ../../web/php/intermedio-page.php");
```

avance-pregunta.php:

Nota de los desarrolladores

El orden en el que se ejecuta este código es MUY importante. Alterar con cautela.

Con el siguiente código, al provenir de responder preguntas, aumenta en uno la cantidad de preguntas respondidas, se almacena variables de forma local para facilitar su uso, y luego se actualiza la información del paso del jugador en la DB.

```
//Aumenta la cantidad de preguntas respondidas
if(isset($_POST['respuestas'])){
    $_SESSION['juego']['paso'] += 1;
    $id = $_SESSION['usuario']['id'];
    $paso = $_SESSION['juego']['paso'];
    $update = "UPDATE journey SET `journey-step` = '$paso' WHERE `id-user` = '$id'";
    mysqli_query($link,$update);
}
```

Esta parte permite procesar las respuestas, calcular el puntaje y almacenarlo en la base de datos. En caso de que la respuesta sea incorrecta (`$_POST['respuestas']` no sea igual a `$_SESSION['juego']['ans_c']`), se perderá la racha (`$_SESSION['juego']['racha']` se reiniciará a 0).

```
//Procesar respuesta, calcular puntaje y almacenarlo.
if(isset($_POST['respuestas']) && ($_POST['respuestas'] ==
$_SESSION['juego']['ans_c'])){
    ++$_SESSION['juego']['racha'];
    $_SESSION['tabla']['puntaje'] += ((10 * $_SESSION['juego']['etapa']) *
$_SESSION['juego']['racha']);

    $puntaje = $_SESSION['tabla']['puntaje'];
```

```
    $update = "UPDATE $edad SET `ranking-score` = '$puntaje' WHERE `id-user` = $id";

    mysqli_query($link,$update);
}else if(isset($_POST['respuestas']) && ($_POST['respuestas'] !=
$_SESSION['juego']['ans_c'])) {
    $_SESSION['juego']['racha'] = 0;
}
```

Con el siguiente bloque de código, se abre un condicional: una vez se respondan las 9 preguntas, se registra el tiempo actual en el registro del usuario perteneciente al momento de finalización del juego.

```
//Salida del bucle una vez responda nueve preguntas
if($_SESSION['juego']['paso'] >= 9){
    $id = $_SESSION['usuario']['id'];
    $update = "UPDATE journey SET `journey-ended` = current_timestamp() WHERE `id-user`
= '$id'";
    mysqli_query($link,$update);
```

Dentro del condicional anterior, recuperaremos el momento en que el usuario inicio el juego, y las fechas (de inicio y finalización) son almacenadas en variables.

```
    #Recuperación y comparación del tiempo transcurrido en el juego.
    #-------------------------------------------------------------------------------
    $select = "SELECT*FROM journey WHERE `id-user`='$id'";
    $result = mysqli_query($link, $select);
    $validation = mysqli_num_rows($result);

    $journey = mysqli_fetch_assoc($result);

    $first  = date_create($journey['journey-started']);
    $second = date_create($journey['journey-ended']);
```

Luego, se almacena una variable con la diferencia (o tiempo transcurrido) entre ambos (inicio y fin), a la cual se le da formato, y se registra en la BD. Una vez hecho esto, el tiempo transcurrido se almacena en la variable de sesión.

```
    $diff = $first->diff($second);
    $et = $diff->format( '%H:%I:%S' );

    $update = "UPDATE $edad SET `ranking-et` = '$et' WHERE `id-user` = $id";

    mysqli_query($link, $update);

    $_SESSION['tabla']['tiempo'] = $et;
```

Para terminar con el condicional, se redirecciona al jugador a la página de `ranking.php`.

```
#Redirección al ranking.
    #-------------------------------------------------------------------------
    header("Location: ../../web/php/ranking.php");
    die();

}
```

El siguiente bloque de código sirve para buscar nuevas preguntas en caso de que se haya respondido la pregunta anterior, se inicie el juego, o se esté pasando a la siguiente etapa.

```
//Recupera las preguntas sólo si se respondió la anterior
if(isset($_POST['respuestas']) || $_SESSION['juego']['paso'] == 0 ||
$_SESSION['juego']['paso'] == 3 || $_SESSION['juego']['paso'] == 6){
    include 'rand-get.php';
}
```

Como último condicional: al finalizar una etapa, se reinicia la lista de sub-tópicos usados y se redirecciona a `intermedio-page.php` (para poder cargar la siguiente etapa y seguir con las preguntas) o, lo envía de vuelta a `Q&A.php` en caso de que no se haya terminado la etapa.

```
//Redirecciona al intermedio y borra la banlist
if(($_SESSION['juego']['paso'] == 3 && $_SESSION['juego']['etapa'] == 1) ||
($_SESSION['juego']['paso'] == 6 && $_SESSION['juego']['etapa'] == 2)){
    unset($_SESSION['juego']['banlist']);
    header( "Location: ../../web/php/intermedio-page.php");
    die();
} else {
    header("Location: ../../web/php/Q&A.php");
}
?>
```

get-progression.php:

Establece que el contenido debe ser leído como parte de una aplicación y no como una página web(HTML).

```
header("Content-Type: application/json");
```

Es establecida la conexión con la base de datos.

```
    #Realizar, comprobar y almacenar credenciales de la conexión a la DBMS.
    #-------------------------------------------------------------------------
    include 'header\link.php';
```

```
    $link = connect();
```

Se recuperan y almacenan los datos del usuario en un array multidimensional. Luego se almacena el id en una variable separada.

```
    #Recolección y clasificación de las entradas del ranking.
    #-------------------------------------------------------------------------
    $select = "SELECT `id-user`, `journey-step` FROM journey ORDER BY `id-user` DESC
LIMIT 1";
    $result = mysqli_query($link, $select);
    $data['journey'] = mysqli_fetch_assoc($result);
    // print_r($data);
    $id = $data['journey']['id-user'];
```

En el siguiente bloque se busca y guarda la edad (`user-age`) del último usuario.

```
    #Comprobar la edad del último usuario para elegir ranking a mostrar.
    $select = "SELECT `user-age` FROM users WHERE `id-user` = '$id'";
    $result = mysqli_query($link, $select);
    $user = mysqli_fetch_assoc($result);
```

Se compara la edad del jugador con la división de edad para poder tomar los datos de la tabla de puntuaciones correcta (`rankingmenores` o `rankingmayores`). Luego, toda la información recolectada, que fue almacenada previamente en el array `$data`, es codificada en formato JSON, y escrita en la página.

```
#Recolección y clasificación de las entradas del ranking.
    #-------------------------------------------------------------------------
    if ($user['user-age'] < 15) {
        $select = "SELECT `ranking-score` FROM rankingmenores WHERE `id-user` = '$id'";
    }elseif($user['user-age'] >= 15){
        $select = "SELECT `ranking-score` FROM rankingmayores WHERE `id-user` = '$id'";
    }
    $result = mysqli_query($link, $select);
    $data['ranking'] = mysqli_fetch_assoc($result);

    echo json_encode($data); // Escribir contenido de la tabla en formato JSON, que será
enviado a la página de ranking.
?>
```

get-ranking.php:

Establece que el contenido debe ser leído como parte de una aplicación y no como una página web(HTML).

```
header("Content-Type: application/json");
```

Es establecida la conexión con la base de datos.

```
    #Realizar, comprobar y almacenar credenciales de la conexión a la DBMS.
    #-------------------------------------------------------------------------
    include 'header\link.php';
    $link = connect();
```

Se obtiene el registro del último usuario, y se almacena su ID.

```
    #Comprobar último usuario.
    $select = "SELECT `id-user` FROM journey ORDER BY `id-user` DESC LIMIT 1";
    $result = mysqli_query($link, $select);
    $user = mysqli_fetch_assoc($result);
    $id = $user['id-user'];
```

Luego, es comprobada la edad del usuario para saber a qué ranking enviarlo.

```
    #Comprobar la edad del último usuario para eleguir ranking a mostrar.
    $select = "SELECT `user-age` FROM users WHERE `id-user` = '$id'";
    $result = mysqli_query($link, $select);
    $user = mysqli_fetch_assoc($result);
```

Se verifica la edad del usuario y se solicitan los datos requeridos del ranking correspondiente, ordenados en principio de manera descendente por el puntaje y, de forma secundaria, de manera ascendente por el tiempo transcurrido. Luego, la información recolectada, es almacenada en el array `$val`, codificada en formato JSON, y escrita en la página.

```
    #Recolección y clasificación de las entradas del ranking.
    #-------------------------------------------------------------------------
    if ($user['user-age'] < 15) {
        $select = "SELECT `ranking-nick`, `ranking-score`, `ranking-et` FROM
rankingmenores ORDER BY `ranking-score` DESC, `ranking-et` ASC";
    }elseif($user['user-age'] >= 15){
        $select = "SELECT `ranking-nick`, `ranking-score`, `ranking-et` FROM
rankingmayores ORDER BY `ranking-score` DESC, `ranking-et` ASC";
    }
    $result = mysqli_query($link, $select);
    $val = mysqli_fetch_all($result, MYSQLI_BOTH);
    echo json_encode($val); // Escribir contenido de la tabla en formato JSON, que será
enviado a la página de ranking.
?>
```

intermedio.php:

Para comenzar, es iniciada la sesión, se establece la conexión con la base de datos y se comunica el tipo de contenido al interprete.

```php
<?php
    session_start();

    #Realizar, comprobar y almacenar credenciales de la conexión a la DBMS.
    #-------------------------------------------------------------------------
    include 'header\link.php';
    $link = connect();

    header("Content-Type: application/json"); // Advise client of response type
```

En el siguiente bloque de código, se almacena las variables de sesión necesarias en variables locales para trabajarlas con mayor facilidad, y se obtiene el último registro de la tabla de señales.

```php
    #Validación de requisitos para el pasaje de etapa.
    #-------------------------------------------------------------------------
    $etapa = $_SESSION['juego']['etapa'];
    $paso = $_SESSION['juego']['paso'];

    $signals = mysqli_query($link, "SELECT * FROM signals ORDER BY `id-record` DESC
LIMIT 1");
    $signal = mysqli_fetch_array($signals);

    $id = $_SESSION['usuario']['id'];
```

El siguiente bloque contiene una serie de validaciones de requisitos para poder pasar de etapa, esta consta en realizar las 3 preguntas correspondientes y pasar a la siguiente fase, hasta llegar a la página final, el ranking.

```php
    if (($paso == 0 && $_SESSION['juego']['etapa'] == 1) || ($paso == 3 &&
$_SESSION['juego']['etapa'] == 2) || ($paso == 6 && $_SESSION['juego']['etapa'] == 3)){
        echo "Q&A";
        die();
    }
    if ($paso == 0 && $signal['signal-stage'] == 1){
        mysqli_query($link, "UPDATE journey SET `journey-stage` = '1', `journey-step` =
'$paso' WHERE `id-user` = '$id'");
        $_SESSION['juego']['etapa'] = 1;
        echo $_SESSION['juego']['etapa'];
        die();
    }
```

```
    if ($paso == 3 && $signal['signal-stage'] == 2){
        mysqli_query($link, "UPDATE journey SET `journey-stage` = '2', `journey-step` =
'$paso'  WHERE `id-user` = '$id'");
        $_SESSION['juego']['etapa'] = 2;
        echo $_SESSION['juego']['etapa'];
        die();
    }
    if ($paso == 6 && $signal['signal-stage'] == 3){
        mysqli_query($link, "UPDATE journey SET `journey-stage` = '3', `journey-step` =
'$paso' WHERE `id-user` = '$id'");
        $_SESSION['juego']['etapa'] = 3;
        echo $_SESSION['juego']['etapa'];
        die();
    }
    if ($paso == 9){
        echo 'No';
        die();
    }
?>
```

player-progression.php:

En este documento lo que se muestra es la trayectoria que lleva el usuario en pantalla aparte, según el jugador vaya avanzando en el juego, en un monitor o TV se mostrara el progreso que lleva.

```
<div class="id">
                <span id="id"></span>
            </div>
            <div class="data">
                <span>Puntaje</span>
                <h1 id="score"></h1>
            </div>
        </div>
        <div class="contenedor">
            <div class="subcontenedor">
                <div class="step easy nonctive" id="step-3"><span class="text" id="3-
text">3</span></div>
                <div class="step easy nonctive" id="step-2"><span class="text" id="2-
text">2</span></div>
                <div class="step easy nonctive" id="step-1"><span class="text" id="1-
text">1</span></div>
            </div>
            <div class="subcontenedor">
                <div class="step medium nonctive" id="step-6"><span class="text" id="6-
text">6</span></div>
                <div class="step medium nonctive" id="step-5"><span class="text" id="5-
text">5</span></div>
```

```
                <div class="step medium nonctive" id="step-4"><span class="text" id="4-
text">4</span></div>
            </div>
            <div class="subcontenedor">
                <div class="step hard nonctive" id="step-9"><span class="text" id="9-
text">9</span></div>
                <div class="step hard nonctive" id="step-8"><span class="text" id="8-
text">8</span></div>
                <div class="step hard nonctive" id="step-7"><span class="text" id="7-
text">7</span></div>
            </div>
        </div>
    </div>
    <script src="../../web/javascript/player-progression.js"></script>
```

Q&A.php:

Aquí se muestran las preguntas y respuestas recibidas de `avance-pregunta.php`, y permite enviar la respuesta. Posee un fondo de pantalla que cambia de forma dinámica según la temática de la pregunta, y también una barra de progresión que muestra la cantidad de preguntas respondidas.

```
<body background="<?PHP echo "../img/Q&A-backgrounds/" . $_SESSION['juego']['sub'] .
".jpg";?>">
    <div class="transparent-background"></div>
    <h1><?php echo $_SESSION['juego']['question'];?></h1>
    <form method="POST" action="avance-pregunta.php">
        <div class="respuestas">
            <label class="respuesta"><input name="respuestas" id="R1" type="radio"
value=1><?php echo $_SESSION['juego']['ans1'];?></label>
            <label class="respuesta"><input name="respuestas" id="R2" type="radio"
value=2><?php echo $_SESSION['juego']['ans2'];?></label>
            <label class="respuesta"><input name="respuestas" id="R3" type="radio"
value=3><?php echo $_SESSION['juego']['ans3'];?></label>
        </div>
        <input class="btn-submit" type="submit">
    </form>
    <progress id="q" max="10" value="<?php echo $p+1?>" class="barra <?php echo $e ?>" >
<?php echo $p?>0% </progress>
</body>
```

Nota: `$e` pretendía cambiar el color de la barra según la dificultad. No fue posible con lo que intentamos

rand-get.php:

Primeramente, es creada una variable que referencia al archivo de preguntas y se le asigna el nombre de un archivo de prueba.

```
<?php
$file = "file.csv";//dirección del archivo csv
```

Luego de eso, se asignan variables al usuario, al último sensor pisado, y a la cantidad de preguntas que lleva respondidas.

```
$age = $_SESSION['usuario']['edad'];
$stage = $_SESSION['juego']['etapa'];//último sensor pisado por el usuario
$pos = $_SESSION['juego']['paso'];//cantidad de preguntas respondidas
```

A continuación, se selecciona un archivo de preguntas según la edad del jugador, es decir, si es mayor o menor de 15 años.

```
if($age == "rankingmenores"){                //Se selecciona el archivo csv a utilizar
según la edad del usuario
    $file = "menores.csv";
}elseif($age == "rankingmayores"){
    $file = "mayores.csv";
}
```

Ahora, se establece el rango de preguntas a seleccionar según su dificultad:

- Fáciles

```
//Easy
if($stage==1){
    $min = 0;
    $max = 29;

}//Medium
```

- Medias

```
}//Medium
elseif($stage==2){
    $min = 30;
    $max = 59;

}//Hard
```

- Difíciles

```
}//Hard
elseif($stage==3){
    $min = 60;
    $max = 89;
}
```

A continuación, se busca una pregunta cuyo tópico no haya aparecido antes para evitar la repetición.

```
//Busca la pregunta de un tópico que no haya sido respondido aún en la etapa
do{
    $question_index = rand($min,$max);
    //mapea el archivo
    $csv = array_map('str_getcsv', file($file,FILE_SKIP_EMPTY_LINES));
    $keys = array_shift($csv);
    //transforma el archivo a un array multidimensional
    foreach ($csv as $i=>$row) {
        $csv[$i] = array_combine($keys, $row);
    }
    $tema = $csv[$question_index]["tópico"];
}while(in_array($tema,$banlist) == True);
```

Ahora, se almacena la pregunta obtenida de forma aleatoria, junto con su index, sus respuestas posibles, respuesta correcta, y se añade el tópico de la pregunta a la lista de tópicos a evitar (banlist).

```
//se almacena el índice de la pregunta
$_SESSION['juego']['i'] = $question_index;
//se almacena la pregunta
$_SESSION['juego']['question'] = $csv[$question_index]["pregunta"];
//se almacenan las respuestas
$_SESSION['juego']['ans1'] = $csv[$question_index]["respuesta1"];
$_SESSION['juego']['ans2'] = $csv[$question_index]["respuesta2"];
$_SESSION['juego']['ans3'] = $csv[$question_index]["respuesta3"];
//se almacena la respuesta correcta
$_SESSION['juego']['ans_c'] = $csv[$question_index]["respuesta_correcta"];
$_SESSION['juego']['sub'] = $csv[$question_index]["subtópico"];
$_SESSION['juego']['banlist'][] =$tema;
?>
```

ranking.php:

En este documento se verá el ranking en la página final, primero es iniciada la sesión y recuperado el puntaje, para luego decidir el emoji a mostrar.

```
<?php session_start();
    if(isset($_SESSION['usuario']['id'])){
```

```
        $puntaje = $_SESSION['tabla']['puntaje'];
    }
?>
```

Aquí se muestran los puntos y tiempo transcurrido en la partida del usuario, junto a un emoji por haber terminado el juego.

```
            <?PHP
                if(isset($_SESSION['usuario']['id'])){
                    include 'header/emoji.php';
                    ?><span class="puntos"><b>Puntos:</b> <?PHP echo
$_SESSION['tabla']['puntaje'];?></span>
                    <span class="puntos"><b>Tiempo:</b> <?PHP echo
$_SESSION['tabla']['tiempo'];?></span> <?PHP
                }else{
                    ?><img src="../../web/img/school-icon.svg"><?PHP
                }
            ?>
```

Debajo de lo anteriormente mencionado, podemos ver una tabla que contiene la posición en la que quedo el jugador, su nick, el puntaje y, finalmente, el tiempo transcurrido en el juego. Tanto los suyos, como lo de los demás jugadores que ya hayan pasados.

```
            <input class="input-100" type="text" placeholder="Buscar por nick.."
id="search-input" onkeyup="myFunction(this.value, 'ranking')">
            <div style="overflow: auto; max-height: 400px;">
                <table id="ranking">
                    <tr>
                        <th class="table-header">Posición</th>
                        <th class="table-header">Nick</th>
                        <th class="table-header">Puntaje</th>
                        <th class="table-header">Tiempo</th>
                    </tr>
                </table>
                <!--<tr><td>Sin registros.</td></tr>-->
            </div>
```

Para finalizar este documento, lo que se hará es destruir la sesión del jugador, para que el mismo usuario no siga accediendo, y otro usuario pueda jugar.

```
    <?PHP
    #Cierre de sesión.
    #-------------------------------------------------------------------------
        session_unset();
        session_destroy();
        setcookie("PHPSESSID", "", time()-1000,"/", "127.0.0.1",false,false);
```

```
    ?>
```

registro.php:

Primero, se inicia la sesión, luego se declaran y definen variables con los datos recuperados a través del método POST.

```
<?php
  session_start();

  #Obtener variables desde el formulario en HTML, con el método POST.
  $email = $_POST['email'];
  $edad = $_POST['edad'];
  $letras = $_POST['nick'];//
  $nums = $_POST['nums'];//
  $nick = $letras.$nums;//se arma el nickname
```

Se establece la conexión con la BD.

```
  #Realizar, comprobar y almacenar credenciales de la conexión a la DBMS.
  #-------------------------------------------------------------------------
  include 'header\link.php';
  $link = connect();
```

A continuación, se busca en la base de datos registros de la cuenta, para validar la existencia de la cuenta.

```
  #Validación de cuenta.
  #-------------------------------------------------------------------------
  $select = "SELECT*FROM users WHERE `user-email`='$email'";
  $result = mysqli_query($link, $select);
  $validation = mysqli_num_rows($result);
```

En el caso de que el usuario no exista, se procederá a guardar estos datos en la DB. En el caso de que sí exista, se mostrará un mensaje en pantalla.

```
  if ($validation == 0) {
    #Insertar nuevo usuario.
    $insert = "INSERT INTO users (`user-age`, `user-email`, `nick`) VALUES ('$edad',
'$email','$nick')";
    mysqli_query($link, $insert);
    $result = mysqli_query($link, $select);
  } else {
    $_SESSION['index_message'] = "Cuenta ya registrada.";
  }
```

Luego de que el jugador se registre, se evaluará su edad para saber a que preguntas y ranking asignarlo.

```
  $user = mysqli_fetch_assoc($result);
  $id = $user['id-user'];
  $edad = $user['user-age'];

#-------------------------------------------------------------------------------------
------

  if($edad < 15){
    $edad = "rankingmenores";
  }elseif($edad >= 15){
    $edad = "rankingmayores";
  }
```

Al registrarse, se le pidió al usuario que ingrese un nick, y aquí revisamos en la base de datos si este ya está utilizado o no.

```
  #Comprobar si el nick está utilizado.
  #-------------------------------------------------------------------------
  $select = "SELECT*FROM $edad WHERE `ranking-nick`='$nick'";
  $result = mysqli_query($link, $select);
  $validation = mysqli_num_rows($result);
```

Si no hay registros de ese nombre de usuario, se insertará ese nombre en la base de datos y estará libre para jugar. Si, por el contrario, ese nombre ya está registrado, mostramos un mensaje en pantalla.

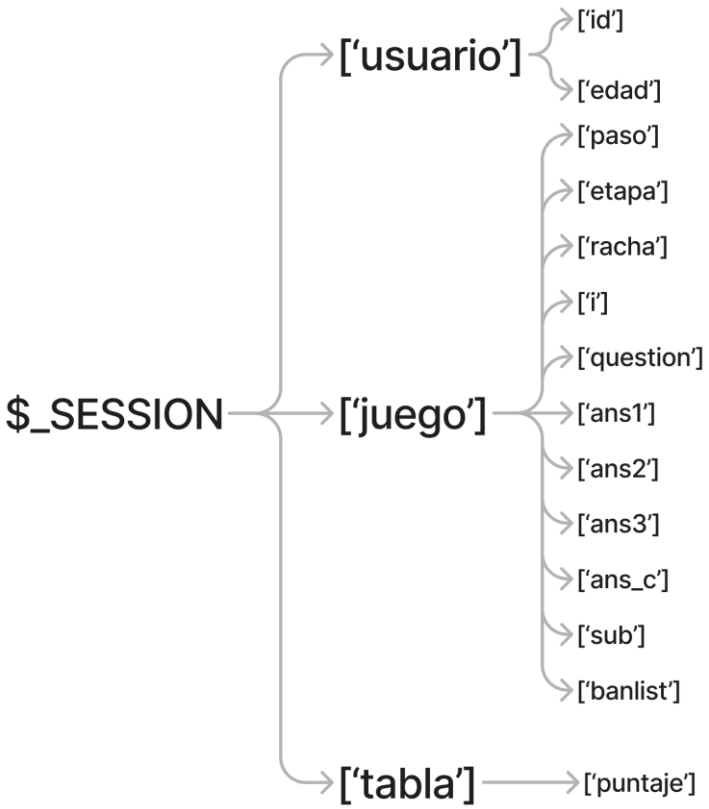
```
  if ($validation == 0) {
    #Insertar nick y ID de usuario al ranking correspondiente.
    $select = "SELECT*FROM $edad WHERE `id-user`='$id'";

    $result = mysqli_query($link, $select);
    $validation = mysqli_num_rows($result);

    if ($validation == 0) {
      $insert = "INSERT INTO $edad (`id-user`, `ranking-nick`) VALUES ('$id', '$nick')";
      mysqli_query($link, $insert);
    }
  } else {
      $_SESSION['index_message'] = "Nick ya registrado.";
  }
```

Una vez completado este proceso y, en vista que el dispositivo usado para registrarse no es el mismo que el utilizado para jugar, enviaremos al usuario de vuelta a la página de registro.

```
  #Redirección para evitar caminos indeseados.
  #-------------------------------------------------------------------------------
  header("Location: ../../web/");
?>
```

Mapa de variables de sesión:

El siguiente mapa, utilizado durante el desarrollo para facilitar la comprensión y seguimiento de la estructura creada y la información almacenada, contiene la ramificación de la variable `$_SESSION`.



Mapa de la variable de sesión.