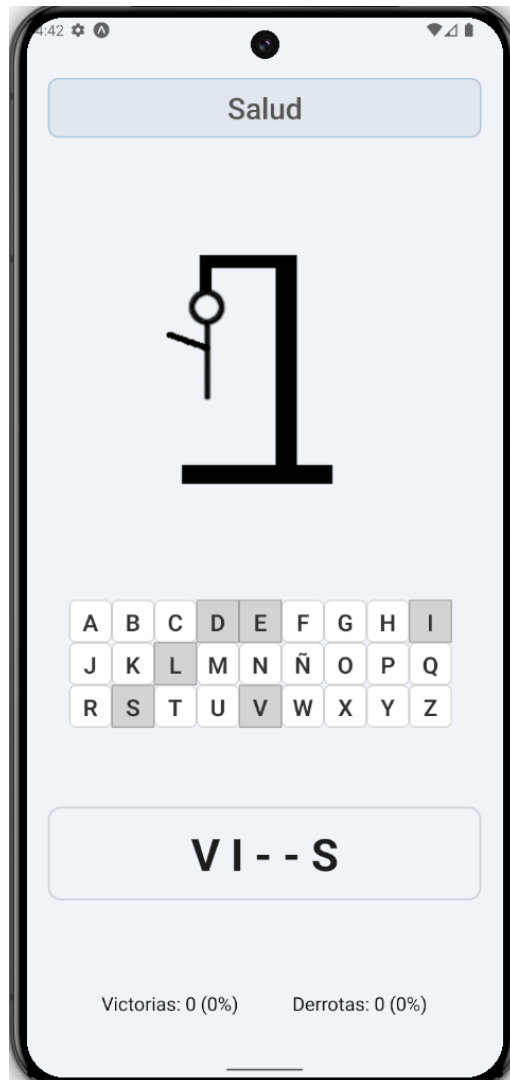


PROYECTO 4

EL AHORCADO



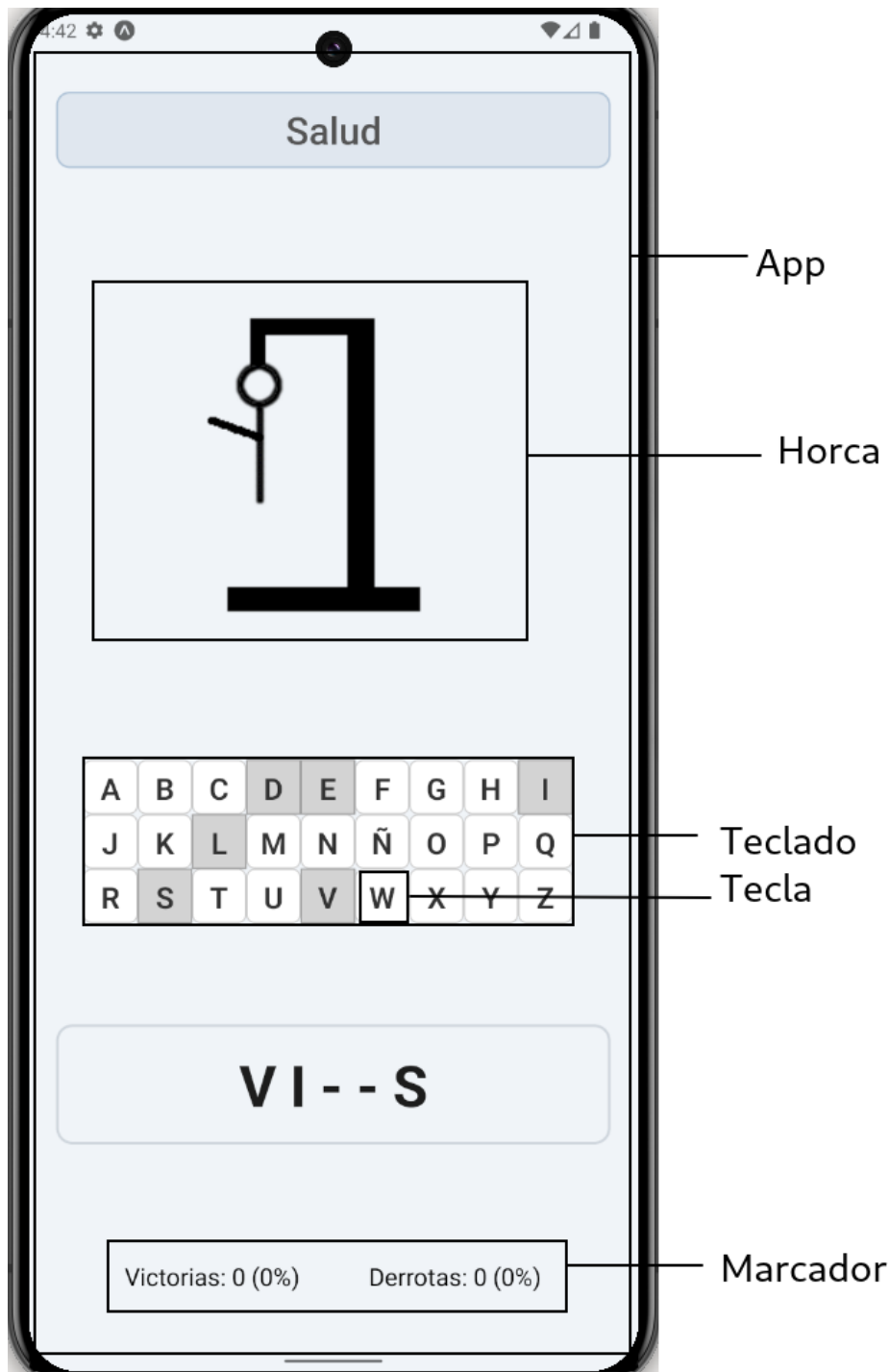
- Número de personas: 2
- Tutorial: 10
- Lenguaje: JavaScript
- Tipo de proyecto: guiado por componentes

• **¿Qué hace la aplicación?** Este proyecto es un sencillo juego del ahorcado, en el que hay que adivinar una palabra elegida aleatoriamente entre varias categorías

Carpetas del proyecto

- **assets:** Se copiarán en ella las imágenes necesarias
- **data:** Se copiará aquí el archivo **palabras.json** con las categorías y palabras del juego
- **components:** Se programarán aquí los componentes del proyecto
- **helpers:** Se pondrá aquí un archivo con funciones auxiliares para programar el juego

Estructura de componentes



helpers/Funciones.ts

Este archivo define y exporta los siguientes elementos, que se utilizan en **App.tsx** para programar el juego

- **PALABRAS:** Es una constante en la que se carga el archivo **palabras.json**
- **getCategorias():** Devuelve la lista de todas las categorías que hay en el objeto **PALABRAS**

Ayuda: Usa la función **Object.keys**, que sirve para obtener una lista con todas las claves de un objeto.

- **generarPalabraAleatoria:** Es una función que elige una categoría aleatoria (de entre todas las categorías que hay en el objeto **PALABRAS**), y de la lista de palabras de esa categoría, elige una palabra aleatoria. Devuelve un objeto con dos claves:

- **categoría:** Es la categoría aleatoria elegida
- **palabra:** Es la palabra aleatoria elegida

Ayuda: Usa la función **Math.random** para generar un número aleatorio entre $[0,1)$, multiplícalo por **N** para obtener un número aleatorio entre $[0,N)$ y por último, usa **Math.floor** para redondearlo a entero.

- **generarDisplayInicial(palabra):** Es una función que recibe una palabra y devuelve un **string** formado por tantos guiones (-) como letras tiene la palabra recibida.
- **realizarIntento(palabra,display,letra):** Es una función que recibe la palabra que hay que adivinar, el display (la lista de guiones y letras descubiertas por el usuario) y la letra que intenta el usuario. El objetivo de esta función es devolver un **string** con el display actualizado tras ver si la letra está en la palabra que hay que acertar. Para ello, se parte de un **string** vacío y se rellenará su contenido de esta forma:
 - Se recorren todas las letras de **display**
 - Si la letra recorrida ha sido descubierta, esa letra se añade tal cual al resultado
 - Si la letra recorrida no ha sido descubierta, se comprueba si la letra de la correspondiente posición en **palabra** coincide con **letra**
 - Si es así, se añade **letra** al resultado
 - Si no es así, se añade un guión al resultado

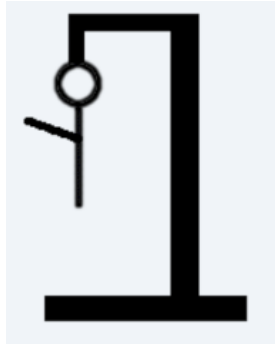
La función devolverá un objeto con dos claves:

- **actualizado:** Vale true si el nuevo display es distinto al recibido
- **display:** Es el **string** con el resultado que se ha formado en esta función

components/Horca.ts

Este componente representa la imagen de la horca que se va formando conforme se pierden vidas

Diseño



Props

- **vidas:** Es el número de vidas

Estilos

- imagen de la horca:
 - o anchura: **70%**
 - o se mantendrá la relación de aspecto

Indicaciones

Se insertará un **Image** que mostrará una foto, según el número de vidas. Con todas las vidas la foto es **ahorcado_6.png** y se va bajando hasta llegar a 0 vidas, que muestra **ahorcado_0.png**

components/Tecla.tsx

Este componente es una tecla que solo se puede pulsar una vez. Cuando la tecla ha sido pulsada, ya no se puede volver a pulsar.

Diseño



Props

- **letra:** Es la letra que se muestra en la tecla
- **pulsarLetra:** Es la función que se ejecuta cuando se pulsa por primera vez la tecla. Es una función que recibe como parámetro un **string** y se le deberá pasar la letra que se muestra en la tecla.
- **usada:** Es un **boolean** que vale **true** si la tecla ya se ha pulsado y no puede volver a pulsarse.

Estilos

- Contenedor:
 - Tamaño **40x40**
 - Ancho de borde: **1**
 - Padding **5**
 - Radio del borde **6**
 - Color del borde **#ccc**
 - Color de fondo **#fff**
- Contenedor deshabilitado (se pone cuando la tecla ya ha sido pulsada)
 - Mismas características que el anterior, excepto:
 - Color del borde **#999**
 - Color de fondo: **#d3d3d3**
- Texto de la letra:
 - Centrado en el contenedor
 - Tamaño de letra **22**
 - Color **#333**
 - Peso (**fontWeight**) de **600**

Indicaciones

Se muestra el texto de la letra en el contenedor indicado, de forma que al pulsarlo por primera vez se llamará a la función del prop **pulsarLetra** pasándole como parámetro la letra que se muestra en la tecla. Si la tecla ya ha sido usada, no se podrá pulsar el botón y además se mostrará el estilo deshabilitado

components/Teclado.tsx

Este componente es la colección de todas las teclas.

Diseño



Props

- **pulsarLetra:** Es la función que se ejecuta cuando se pulsa una tecla. Dicha función recibe un **string** con la tecla pulsada.
- **letrasUsadas:** Es un **string** que contiene (concatenadas una tras otra) todas las letras que han sido pulsadas. Por ejemplo, si se han pulsado las teclas V, P, S, dicho **string** será "VPS"

Estilos

- contenedor
 - o dispone en fila sus elementos y coloca una tras otras las teclas, bajando de renglón automáticamente al llegar al extremo derecho
 - o padding horizontal **20**
 - o alineación centrada

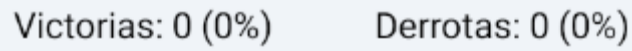
Indicaciones

Se añadirán al contenedor todas las teclas. Se puede hacer "a mano", escribiendo las 27 teclas una tras otra, o hacer una función auxiliar que devuelva la lista de las letras (insertar la Ñ tras la N) y usar **map** para transformar dicha lista en una lista de etiquetas **Tecla**

components/Marcador.tsx

Este componente es el marcador donde se ven los datos estadísticos de las partidas jugadas.

Diseño



Victorias: 0 (0%) Derrotas: 0 (0%)

Props

- **victorias:** Es el número de victorias
- **derrotas:** Es el número de derrotas

Estilos

- contenedor
 - o Dispone en fila sus elementos, repartiendo entre ellos el espacio libre
- Texto
 - o Tamaño de letra 18

Indicaciones

Se pueden crear variables locales para almacenar la cantidad total de partidas y los correspondientes porcentajes. Se dispondrá en una fila los dos textos y se mostrarán en ellos todos los datos.

components/App.tsx

Este componente es el componente principal, en el que se realiza la partida.

Estilos

- Contenedor principal
 - Ocupa toda la pantalla
 - Reparte todo en columnas, dejando la misma cantidad de espacio libre entre sus elementos
 - Padding **20**
 - Color de fondo **#f0f4f8**
- Texto donde se muestra la categoría elegida
 - Alineación del texto centrada
 - Tamaño de letra **28**
 - Color de texto **#555**
 - Color de fondo **#e0e7ef**
 - Padding vertical **8**
 - Padding horizontal **16**
 - Radio del borde **16**
 - Peso del texto: **500**
 - Color del borde **#b0c4d8**
 - Anchura del borde **1.5**
- Texto donde se muestra el display
 - Alineación centrada
 - Tamaño de letra **42**
 - Color **#1A1A1A**
 - Espacio entre letras (**letterSpacing**): **10**
 - Negrita
 - Padding vertical **14**
 - Padding horizontal **24**
 - Radio del borde **12**
 - Anchura del borde **2**
 - Color del borde **#d0d7de**

Variables de estado (todas van con su setter)

- **vidas**: Cantidad de vidas
- **display**: Es un **string** con los guiones y letras de la palabra oculta que ve el usuario
- **categoría**: Es un **string** con la categoría elegida aleatoriamente
- **palabra**: Es un **string** con la palabra que hay que adivinar
- **letrasUsadas**: Es un **string** que empieza vacío y en él se van concatenando las letras que se van pulsando en el teclado.
- **victorias**: Es el número de partidas ganadas
- **derrotas**: Es el número de partidas perdidas

Funciones auxiliares

Se programarán en este componente estas funciones: (usará funciones de **Funciones.ts**)

- **inicializarPartida:** Usa la función **generarPalabraAleatoria** para obtener un objeto que contiene una categoría y palabra elegida aleatoriamente, y con ellas, rellena todas las variables de estado. Además, pone **6** vidas, inicializa el display con la función **inicializarDisplay** y las letras usadas las pone vacías ""

Además, se usará **useEffect** de esta forma para llamar a **inicializarPartida** cuando se cargue la app

```
useEffect(inicializarPartida, [])
```

*Cuando se pone **useEffect** pasándole como segundo parámetro una lista vacía [], se llama a la función **inicializarPartida** una sola vez, cuando se carga el componente.*

- **victoria:** Incrementa el contador de victorias y muestra una ventana emergente indicando al jugador que ha ganado. Tendrá un botón "Nueva partida", que al ser pulsado llamará a **inicializarPartida**
- **derrota:** Incrementa el contador de derrotas y muestra una ventana emergente indicando al jugador que ha perdido y cuál era la palabra correcta. Tendrá un botón "Nueva partida", que al ser pulsado llamará a **inicializarPartida**
- **pulsarLetra(letra):** Aquí es donde realmente se juega. Para ello, esta función recibe una letra (que es la jugada del usuario) y hace lo siguiente:
 - Añade a **letrasUsadas** dicha letra
 - Usa **realizarIntento** para obtener un objeto que nos indica si la letra estaba en la palabra que hay que adivinar y el display actualizado.
 - Si la letra estaba en la palabra, se actualizará el display y se mirará si el usuario ha acertado la palabra, llamando a **victoria** en ese caso
 - En caso contrario, se restará una vida al usuario y se mirará si quedan vidas, llamando a **derrota** en ese caso

Indicaciones

El componente dispone verticalmente, con los estilos comentados:

- Un **Text** con la categoría
- Un **Horca** con el número de vidas
- Un **Teclado** pasándole a sus props **pulsarLetra** y **teclasUsadas**
- Un **Text** con el display
- Un **Marcador** con la cantidad de victorias y derrotas