

Documentación prueba Voicemod

Objetivo

El objetivo es construir una aplicación que permita lista las voces proporcionadas por el servidor y poder realizar las siguientes acciones sobre estas:

- 🌊 Buscar por nombre.
- 🌊 Filtrar por la categoría a la que pertenecen.
- 🌊 Ordenar de manera ascendente y descendente por nombre.
- 🌊 Seleccionar una voz al azar entre las disponibles.
- 🌊 Seleccionar una voz al pulsar sobre ella.
- 🌊 Añadir y eliminar la voz de una lista de favoritos.
- 🌊 Adaptación de la aplicación a entornos móviles.

Iniciar la aplicación

- 🌊 Descargar el repositorio de <https://github.com/borjalopezspain/voicemod>.
- 🌊 Acceder a la rama **Main**
- 🌊 Ejecutar el comando **Yarn Install** en la carpeta raíz del proyecto. Esto instalará los paquetes necesarios para la aplicación.
- 🌊 Una vez finalizada la instalación, ejecutar el comando **Yarn start** en la carpeta raíz. Este comando es un comando combinado que levantará el servidor de datos y el servidor de la aplicación.
- 🌊 Si queremos ejecutar la batería de tests unitarios, ejecutaremos el comando **Yarn test:unit** en la carpeta raíz.
- 🌊 Al emplear VUE-CLI como base, al guardar los cambios en un archivo recompila la aplicación y ejecuta una recarga de la página, ayudando así a que el desarrollo sea más fluido.

Tecnologías empleadas

- 🌊 HTML 5
- 🌊 CSS 3
- 🌊 SCSS
- 🌊 VUE.JS
- 🌊 TYPESCRIPT
- 🌊 JSON SERVER

Estructura de archivos

La aplicación se ha estructurado intentando seguir los principios SOLID y un patrón de diseño DDD, englobando las partes de la aplicación que realizan una función en una misma carpeta y separando el código en componentes para que sea más mantenible, escalable y reutilizable.



Para el manejo de las vistas, se ha empleado un sistema de rutas que, a pesar de no ser necesario para este proyecto al incluir sólo una vista, deja preparado el camino a una posible implementación de más pantallas que funcionen con rutas independientes.

Para englobar de manera general la aplicación, se ha creado un nivel superior a las vistas o layout, que contiene el header, el footer y el contenedor de rutas donde se cargan las vistas. Esto permite, en un futuro, crear más layouts para englobar partes generales e independientes de la aplicación como podría ser una pantalla para el login.

Cada componente o vista tiene sus propios estilos enfocados a este con scope.

Para estilos más generales se emplean archivos de estilo que se importan en el archivo general y variables de css y scss para que se puedan reutilizar en los archivos de estilo de los componentes.

En la carpeta 'models' se organizan las interfaces de cada 'dominio'.

Vistas y componentes

Header

El componente del header contiene los componentes que forman los filtros que nos permitirá manejar las voces de las diferentes listas:

- El componente de búsqueda permite filtrar la lista de voces comparando el texto introducido con los nombres de cada una de las voces.
- El componente de voz seleccionada, nos muestra de una manera rápida la voz que tenemos seleccionada, pudiendo desplazar el scroll a ella si pinchamos encima, añadirla o quitarla de favoritos o eliminarla de la selección.
- El componente de ordenación por categoría permite filtrar la lista de voces por la categoría a la que pertenecen.
- El componente de ordenación ordena la lista de voces por ascendente o descendente según en nombre.
- El componente de aleatorio selecciona una voz de la lista de voces disponible de manera aleatoria, haciendo scroll a ella si no se muestra en pantalla para una mayor usabilidad.

El header cambia dependiendo de la resolución de pantalla permitiendo tener un mayor control de los componentes y no saturar un único componente que lo contenga todo. Esto se logra gracias a la componetización de la aplicación que nos permite reutilizar los demás componentes de manera eficaz.

Home

Al tratarse de una aplicación con una única pantalla, la página principal es la que contiene los componentes que la forman. Esta contiene los componentes del contenido que cargan las voces, en este caso, la lista de voces y la lista de favoritos.

Footer

Se ha añadido un componente footer para mostrar información adicional.



Comunicación entre componentes

Para la comunicación entre componentes se ha empleado Vuex, que nos permite ejecutar acciones y modificar datos que pueden ser usados en toda la aplicación y por todos los componentes.

En Vuex, encontramos también las llamadas a los servicios, que gestionan la comunicación con el servidor.

Se ha pretendido una organización con modelo DDD. Cada módulo de Vuex pertenece a un 'dominio' de la aplicación. En este caso se ha empleado un único módulo para ello, pero se podrían emplear más módulos, segregando más la información y empleando un generador de módulo único para evitar la importación individual de cada módulo.

También se ha empleado Emit y Props para comunicación directa entre 'Padre' e 'Hijos'.

Arquitectura de la aplicación

Se ha utilizado Typescript, que nos permite tener un código más mantenible, eliminando errores al emplear una interfaz definida para el uso de objetos y clases. Para ello se ha definido una carpeta models, que estructura las interfaces por 'dominio' y las exporta a un archivo index usando la herramienta barrel para poder acceder de manera más fácil a estas en los lugares donde las necesitemos emplear.

Se ha empleado Vue Class Component que nos permite una mejor organización del código, facilita el uso de Typescript y usar los componentes como clases, heredando todas las propiedades de estas y permitiendo ser extensibles y reutilizables.

Se ha separado la lógica de servicios en una carpeta a parte estructurada por 'dominio', la cual se ha programado en forma de clase para poder aplicarle una interfaz de Typescript, evitando errores y permitiendo contener la información de las llamadas al servidor de manera más clara y reutilizable.

Para mantener la persistencia de datos, se ha empleado JSON Server. Esto nos permite tener una API REST falsa para simular la funcionalidad de un servidor, pudiendo leer, guardar, eliminar y editar datos de forma persistente.

Control de versiones

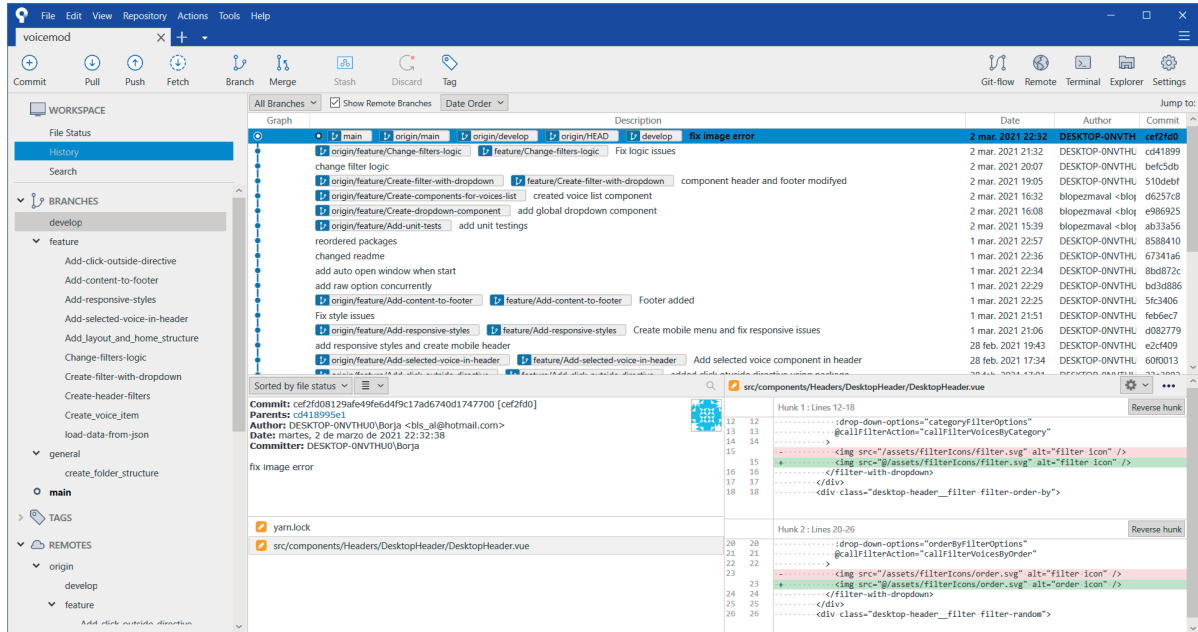
Los archivos de la aplicación se han subido a un repositorio público de github

'<https://github.com/borjalopezspain/voicemod>' y se ha utilizado la herramienta SourceTree para el control de versiones y desarrollo por ramas.






Capturas prueba Voicemod

Control de versiones con Sourcetree

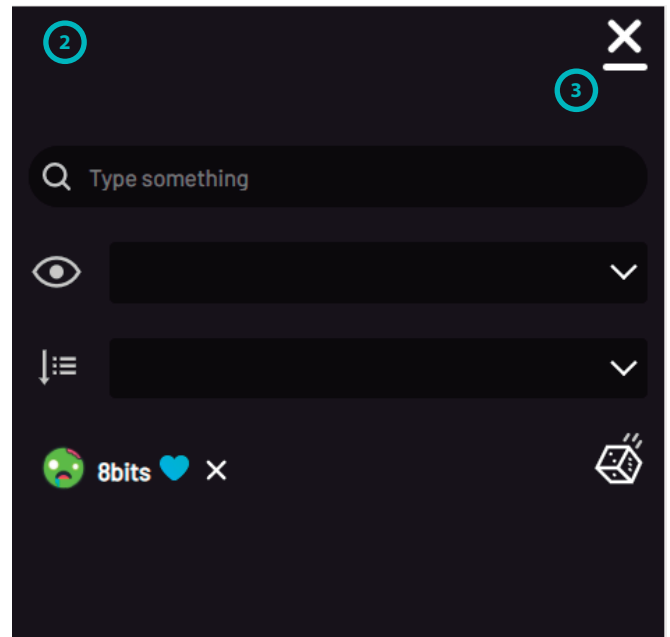
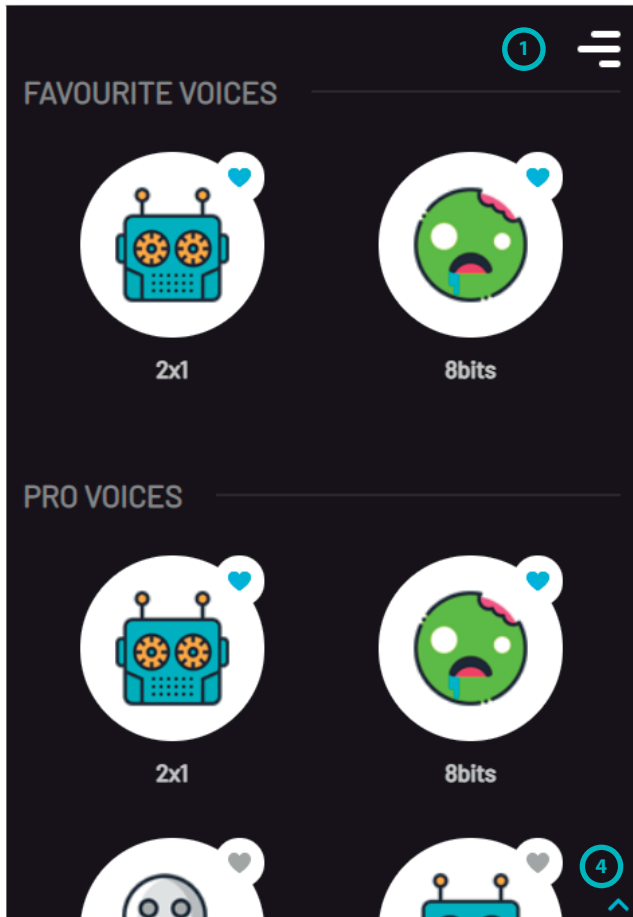


Vista general de la aplicación *desktop*



- ① Búsqueda por nombre
- ② Voz seleccionada. Desde este componente se pueden realizar las acciones de:
 -  Ir a la posición de la voz seleccionada haciendo click sobre el nombre de la voz.
 -  Añadir o eliminar la voz de la lista de favoritos.
 -  Eliminar la voz seleccionada.
- ③ Filtrar por categoría.
- ④ Ordenar por nombre ascendente y descendente.
- ⑤ Seleccionar una voz al azar. Si la voz no está en pantalla, hará scroll a su posición.
- ⑥ Listado de voces favoritas.
- ⑦ Listado de voces filtradas.
- ⑧ Voz seleccionada.
- ⑨ Al hacer *hover* sobre las voces, permite añadir o eliminar de la lista de favoritos haciendo click en el icono del corazón.
- ⑩ Botón *scroll to top*.





1 Botón para desplegar el menú.

2 Menú header en versión móvil.

3 Botón para cerrar el menú.

4 Botón *scroll to top*.