

Memoria Proyecto Final DAW

IES La SENIA |

Pintorrea

ruben saiz serrano

2020

Índice

1. **Introducción2**
2. **Idea de negocio4**
3. **Proyectos existentes2**
4. **Tecnologías utilizadas2**
   1. **Frontend2**
   2. **Backend2**
5. **Introducción**

Este proyecto se trata de una web en la que su producto principal es un juego de dibujar y adivinar en el que hay dos tipos de roles:

* La encargada de dibujar la palabra que la web le sugiera.
* La encargada de adivinar que está dibujando el dibujante.

Los usuarios pueden registrarse y acceder con su propia sesión de esta forma pueden puntuar después de cada partida que jueguen.

Para poder jugar con los demás usuarios debes entrar en una de las salas disponibles e invitarlos.

Las siguientes paginas tendrán los siguientes contenidos:

* Estructura de la aplicación web.
* La idea de negocio.

**4. Tecnologías utilizadas**

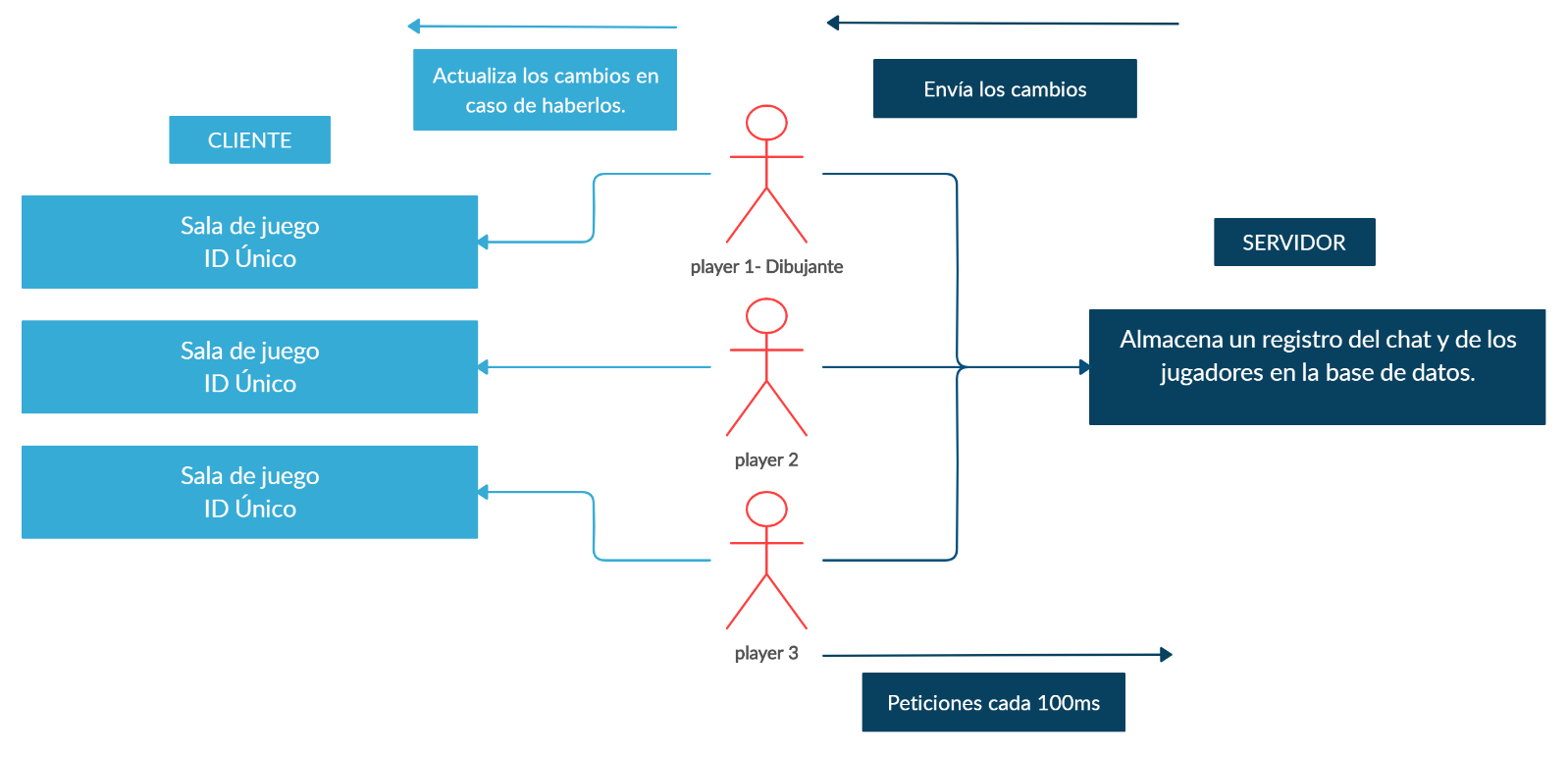
En este proyecto he querido mantener una consistencia a la hora de elegir los lenguajes de programación que iba a utilizar, quería que fuera robusto, rápido y que tuviera una gran base de usuarios activos, por eso he optado por utilizar nada mas que Javascript.

Se que existen muchos entornos de programación como pueden ser LAMP que se compone de Linux, Apache, MYSQL y Php. Pero en mi caso veo mucho mas limpio de mantener el código cuando esta en el mismo lenguaje y también por supuesto nos quitamos de encima cualquier dolor de cabeza en las posibles futuras comunicaciones entre las distintas api’s de cada lenguaje.

Una de las dificultades que he tenido a la hora de desarrollar esta aplicación es enviar y recibir datos entre los usuarios en tiempo real y además que estén conectados a una sala.

Estas son dos de las ideas que he tenido para desarrollar esta funcionalidad:

La primera opción es **HTTP/REST.** Hacer una transferencia de datos por Ajax en la que cada uno de los clientes estará haciendo peticiones al servidor durante cortos plazos de tiempo. El cliente le hará llegar sus cambios, y el servidor después de almacenarlos le devolverá todos los cambios que hay en ese momento.



Esto es extremadamente pesado tanto para el servidor como para el cliente ya que con una petición cada 100 ms en una partida que puede durar por ejemplo 10 minutos estamos hablando de 600 peticiones en una sola partida. Aún encima en cada una de las peticiones el servidor está haciendo mínimo una consulta a la base de datos.

Esto también requiere una lógica extra ya que tienes que evitar duplicidades en las respuestas a los clientes.

La segunda opción es **WEB SOCKETS**, que según la Wikipedia, “*es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único****socket****TCP. Está diseñada para ser implementada en navegadores y servidores****web****, pero puede utilizarse por cualquier aplicación cliente/servidor*”.

Gracias a que tiene una comunicación bidireccional no nos hace falta en el cliente estar preguntando constantemente al servidor por los cambios, sino que establecemos una conexión a un canal con el servidor y en cuanto haya cambios se dispara un evento.

Ahora el cliente y el servidor solo accederían a los datos solo cuando los necesiten, pues bien, es evidente que para resolver este problema he escogido la segunda opción.

Pero esto no termina aquí ya que implementar web sockets de manera nativa es bastante complicado, así que después de una investigación por internet me he decantado por una librería llamada **socket.io**. Las razones son que es bastante fácil de utilizar y no necesitas tanto código como de manera nativa.

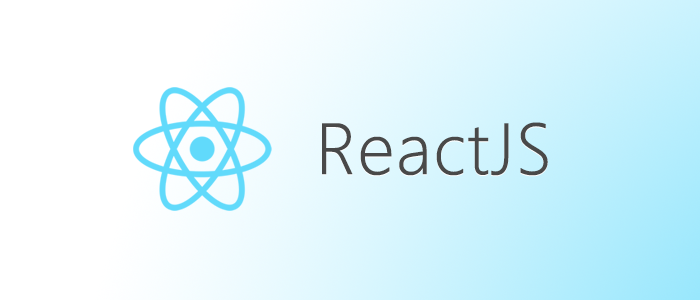
La distribución de conexiones que tengo creadas es la siguiente:

**4.1 Frontend**

Las tecnologías utilizadas en el frontend son:

* [React js](https://es.reactjs.org/)
* [SASS](https://sass-lang.com/)

Ahora paso a explicarlas una a una.

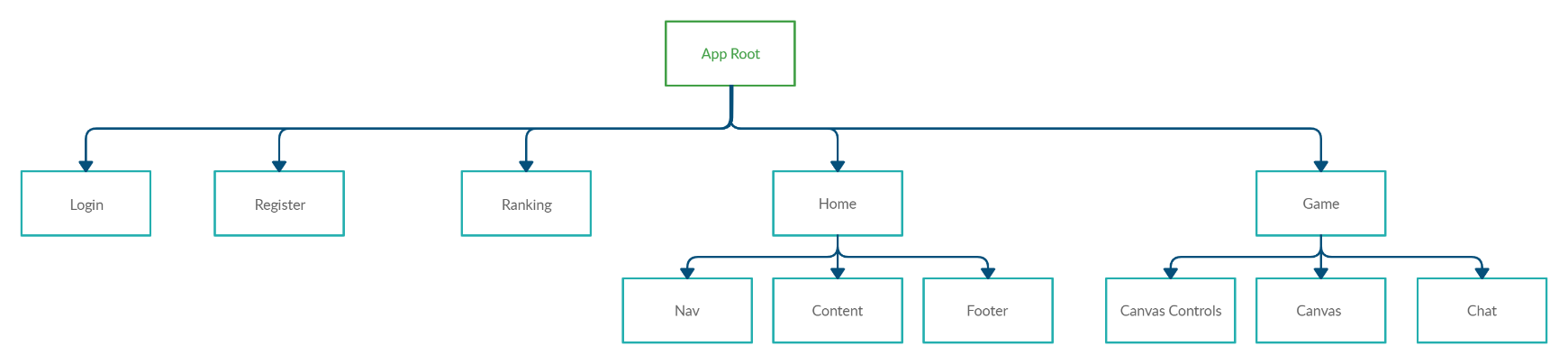
**REACT**

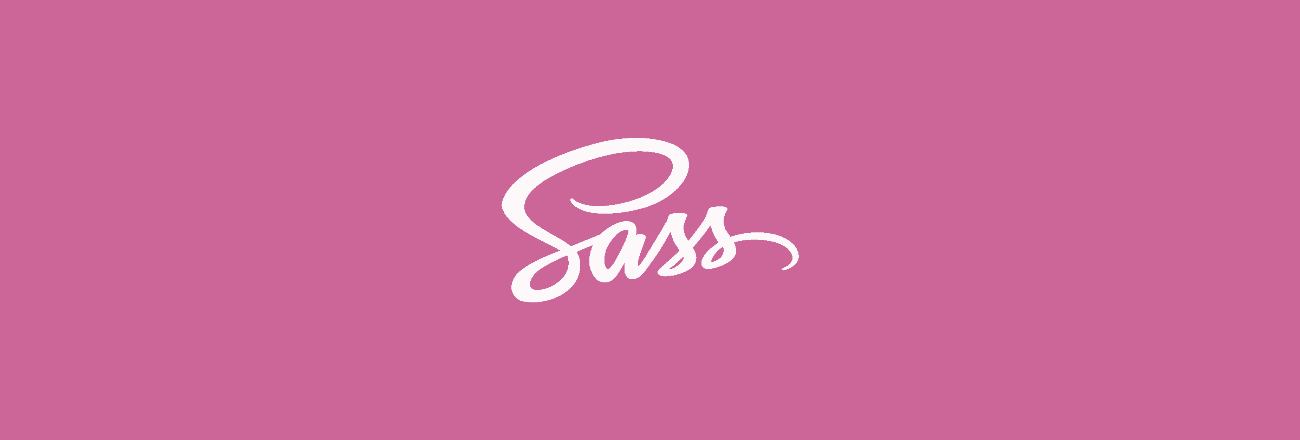
En el frontend he optado por utilizar principalmente **React**, es un framework desarrollado por Facebook. Se encarga de darle interactividad al DOM de una manera mas eficiente que javascript puro ya que se apoya de lo que ellos llaman “Virtual Dom”, gracias a esto puede cambiar el estado de los nodos del DOM con menos manipulaciones.

Ya había utilizado recientemente así que no ha sido muy difícil decantarme por esta tecnología porque en mi opinión es mejor que javascript puro, estos son mis principales argumentos del porqué.

* Es de Facebook, es open source y Facebook lo utiliza en sus sistemas. Gracias a esto hay una gran comunidad de desarrolladores actualmente utilizando React.
* Utiliza el javascript más nuevo, últimas versiones de EcmaSript. Gracias a esto no caes en un estado de confort y aprendes cosas nuevas.
* Lenguaje JSX. Aunque tenga una apariencia a HTML esta es una extensión de javascript para escribir de una forma más elegante los componentes renderizados por el DOM.



La jerarquía de componentes de esta aplicación es la siguiente:

**SASS**

SASS es un preprocesador de estilos que trabaja con una sintaxis bastante parecida y te permite utilizar controles de flujo, bucles, funciones… Esto es una gran ventaja respecto a css porque gracias a esto me ha ayudado a recortar duplicidades en mi código y esto a la larga se agradece porque introducir cambios me será mucho más sencillo.

**SOCKET IO CLIENT**

Que es socket io?