

# Universidad Tecnológica Nacional

Instituto Nacional Superior del Profesorado Técnico



## SISTEMA INFORMATIVO PARA LA RECEPCIÓN DEL HOSPITAL DE CLÍNICAS

Trabajo final de Seminario 3.601

Charras, Sebastian Horacio

Legajo 153.115

[sebastian.charras@alu.inspt.utn.edu.ar](mailto:sebastian.charras@alu.inspt.utn.edu.ar)

Fecha de presentación

Febrero 2024

Docentes de la cátedra

Prof. Dr. Diego P. Corsi

Prof. Matías García



*A mis padres*



## **Resumen**

En el presente trabajo se muestran las problemáticas, el desarrollo y los resultados de la implementación del Sistema Informativo para la Recepción del Hospital de Clínicas. Este sistema busca que los pacientes estén informados sobre qué puestos se encuentran libres para ser atendidos y que también cuenten con una trivia que les permita tener una espera más amena.

Debido a la necesidad de saber si es posible pasar a un puesto para recibir atención o si por el contrario hay que esperar, contar con una pantalla que indique los puestos libres resulta algo conveniente en muchas instituciones, desde bancos y hospitales hasta locales de venta al público. En este caso, los principales beneficios son la disminución del tiempo perdido y simplificación del trabajo, ya que de no haber una pantalla con esta información, un empleado debería indicar en voz alta a los pacientes cada vez que un puesto se libere.

Se realizó un extenso proceso de selección para elegir qué tecnologías usar para este desarrollo, dicho proceso se encuentra documentado en este trabajo al igual que las características del sistema producido y los resultados de su implementación.

## **Abstract**

In this text you will be able to read about the problems, the development and the results of the implementation of the Informative System for the Reception of the Clinics Hospital. This system is meant to bring information to the patients regarding the available spots that they can use to receive attention and also brings them a trivia that can make the wait more entertaining.

Due to the necessity to know if it is possible to go to a spot in order to receive attention or if it is necessary to wait, having a screen that marks which spots are free is convenient in many institutions, from banks and hospitals to retail businesses. In this case, the main benefits are the improvement in the use of time and the simplification of the work, since if there was no screen with this information, an employee would have to indicate to the patients each time a spot becomes available.

An extensive selection process was performed to choose which technologies to use for this development, that process is documented in the present work as well as the characteristics of the produced system and the results of its implementation.



## **Agradecimientos**

Agradezco a mis padres, que me ayudaron a poder emprender mi trayecto de formación académica.

Agradezco a mis compañeros, con los cuales fuimos avanzando en las diversas asignaturas, dándonos ayuda cuando lo necesitábamos.

Y Agradezco a mis docentes, que me llevaron a descubrir nuevas ramas de conocimiento.



# Índice

<b>1</b>	<b>Introducción</b>	<b>13</b>
<b>2</b>	<b>Estado del arte</b>	<b>13</b>
<b>3</b>	<b>Definición del problema</b>	<b>15</b>
<b>4</b>	<b>Solución propuesta</b>	<b>16</b>
4.1	Determinación de metodologías y tecnologías a utilizar	16
4.1.1	Metodologías	17
4.1.2	Lenguaje de programación del lado del cliente	18
4.1.3	Framework para CSS	20
4.1.4	Framework o biblioteca frontend	21
4.1.5	Lenguaje de programación y framework backend	23
4.1.6	Base de datos	25
4.1.7	Testing	26
4.2	Análisis de requisitos	27
4.2.1	Puestos	28
4.2.2	Trivia	28
4.3	Especificaciones del sistema	28
4.4	Análisis del sistema	28
<b>5</b>	<b>Resultados</b>	<b>29</b>
<b>6</b>	<b>Conclusiones</b>	<b>31</b>
6.1	Selección de metodologías y tecnologías	37
6.1.1	Selección de metodologías	37
6.1.2	Selección de lenguajes frontend	38
6.1.3	Selección de framework para CSS	39
6.1.4	Selección de frameworks o bibliotecas frontend	41
6.1.5	Lenguajes y frameworks backend	42
6.1.6	Selección de Sistema Gestor de Base de Datos	43
6.1.7	Selección de framework para testing	45
6.2	Manual de uso	47
6.3	Manual de instalación	50
6.3.1	Requisitos	50
6.3.2	Iniciar minikube	50
6.3.3	Aplicar configuraciones de kubernetes	51
6.3.4	Hacer accesible el sistema	52



## Índice de figuras

1	Hospital con pantalla para distribuir pacientes . . . . .	13
2	Colectivo con pantalla que muestra información relevante . . . . .	14
3	Preguntados . . . . .	15
4	Gráfico de calificaciones para metodologías . . . . .	18
5	Gráfico de calificaciones para lenguajes frontend . . . . .	20
6	Gráfico de calificaciones para frameworks CSS . . . . .	21
7	Uso de las principales tecnologías desde 2016 hasta 2021 . . . . .	22
8	10 Frameworks Web más usados en 2021 . . . . .	22
9	Gráfico de calificaciones para frameworks y bibliotecas frontend . . . . .	23
10	Gráfico de calificaciones para lenguajes y frameworks backend . . . . .	24
11	Gráfico de calificaciones para Sistemas Gestores de Base de Datos . . . . .	26
12	Gráfico de calificaciones de frameworks para hacer testing del código . . . . .	27
13	Interfaz de la pantalla principal del sistema . . . . .	29
14	Interfaz del editor de puesto . . . . .	30
15	Interfaz del editor de puestos avanzado . . . . .	30
16	Interfaz del editor de trivias . . . . .	30
17	Interfaz de la pantalla principal del sistema . . . . .	47
18	Interfaz del editor de puesto . . . . .	48
19	Interfaz del editor de puestos avanzado . . . . .	48
20	Interfaz del editor de trivias . . . . .	49
21	Interfaz del dashboard de Kubernetes . . . . .	51
22	Contenido de un archivo de configuración de kubernetes . . . . .	51



## 1. Introducción

En los hospitales con frecuencia es necesario esperar para ser atendidos en la sección de recepción. Esto hace que en caso de no avisarse a los pacientes qué puestos se encuentran libres, estos esperen más tiempo que el necesario y así este se use de manera ineficiente.

En lugar de requerir que haya una persona anunciando en voz alta cuando se libera un puesto, en la actualidad suelen usarse pantallas que indican los puestos libres y que en ocasiones pueden también dar información adicional a manera de recreación para los pacientes.

El presente trabajo busca eficientizar el uso del tiempo en la recepción para internación del Hospital de Clínicas al proveer una lista de los puestos libres que se actualice de forma automática, y a su vez mostrar una trivia con el propósito de volver más amena la espera.

A lo largo del trabajo podrán observarse las implementaciones realizadas para otras instituciones de sistemas similares (Estado del Arte), cuales son los detalles de las problemáticas que se busca resolver (Definición del problema), que características posee la solución planteada (Solución propuesta), los resultados y las conclusiones.

En el anexo se podrá ahondar en el análisis que se realizó a la hora de elegir las principales tecnologías y metodologías a usar. A su vez allí también se encuentran las instrucciones de uso y de instalación del sistema.

## 2. Estado del arte

En la actualidad se utilizan sistemas que cumplen al menos uno de los dos principales objetivos del proyecto en cuestión, ya sea indicar información relevante o brindar un medio para volver más ameno el momento.

Por ejemplo, en algunos hospitales se usa una pantalla que muestre una lista con los puestos y los números asignados a las personas para indicarles que deben ir a un puesto determinado (fig. 1). Esto puede también ser acompañado por un video.

**Figura 1**

*Hospital con pantalla para distribuir pacientes*



*Nota:* Extraída de Calle (s.f.)

Ahora bien, la implementación de pantallas para mostrar información en hospitales no siempre ha generado resultados positivos. En los casos en que no fueron acompañadas por avisos sonoros, aquellos que sufrían de impedimentos visuales ya no fueron capaces de saber cosas como dónde debían

pasar a consultar ni a qué sala les correspondía entrar. Las soluciones en ocasiones fueron pedir ayuda a auxiliares, atención del paciente u otros, quitando así la posibilidad de ser independientes a estos pacientes (Calle, s.f.).

La utilización de pantallas no se limita a los hospitales únicamente, sino que otros sectores las han implementado para mostrar determinada información. Por ejemplo, en algunos locales de comida rápida pueden verse pantallas en la zona donde se retiran los pedidos que indican a quienes están esperando que ya se está elaborando o que ya se puede retirar su orden. Para esto último pueden usar un número indicado en el tique de compra y otro para el puesto, lo cual es similar al caso de los hospitales en que se muestra la sala a la cual pasar y el número asignado al turno.

De hecho, incluso en algunos colectivos pueden verse pantallas que muestran contenido recreacional para volver más ameno el viaje o que incluso permiten ver información relevante sobre este. Por ejemplo, se instalaron hace años pantallas LED en algunos colectivos que ofrecen información sobre los recorridos de las líneas (fig. 2) y anuncian las paradas más importantes (La Capital, 2015).

**Figura 2**

*Colectivo con pantalla que muestra información relevante*

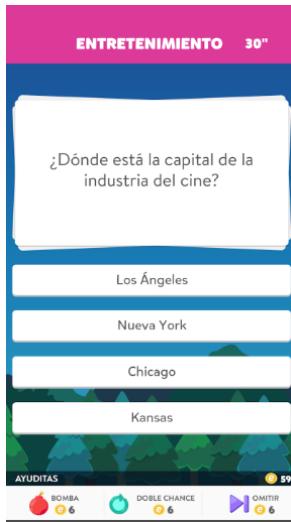


*Nota:* Extraída de La Capital (s.f.)

Dado que se desea usar una trivia para el proyecto en cuestión, se buscó información de un videojuego basado en dicho formato. Preguntados es una trivia usada por millones de personas en el mundo, en especial en Latinoamérica, que permite competir contra amigos. Se contesta entre cuatro posibles respuestas sobre una temática dada (fig. 3). Contiene algunas ayudas, como por ejemplo una bomba para eliminar dos respuestas erróneas y también una opción para cambiar de pregunta. (MINUTOUNO, 2014). Contiene preguntas que pertenecen a categorías como Entretenimiento, Deportes, Arte, Historia, Literatura, Geografía, Ciencia y Tecnología (afp, 2014).

**Figura 3**

*Preguntados*



*Nota:* Extraída de Etermax (s.f.)

Los ejemplos de sistemas vigentes actualmente muestran algunas formas en que se resolvieron problemáticas similares a las que debió enfrentar el proyecto actual. Elementos como la trivia y las indicaciones para distribuir pacientes son necesarios para resolverlas, y también la provisión de una salida sonora para el sistema demostró ser necesaria para así mejorar su accesibilidad.

### 3. Definición del problema

Antes de desarrollar el sistema planteado, existían múltiples problemas en el sector de espera para internación del Hospital de Clínicas.

En primer lugar, los pacientes no sabían a ciencia cierta cuándo un puesto se encontraba libre, y al no existir un sistema informático que se los indicara, era necesario que se avise en voz alta. Esto no resultaba eficiente ni conveniente, por lo que quedaba clara la necesidad de implementar un sistema que, mediante una pantalla y una salida de audio, indique a los pacientes cuando puedan pasar a algún puesto.

Además, al tratarse de una espera para internación, la situación no resultaba amena para los pacientes, por lo que proveer un medio para amenizar el momento mediante la pantalla mencionada anteriormente podía resultar conveniente.

Por último, al contar con un edificio de grandes proporciones, no es fácil saber específicamente ciertas cosas como, por ejemplo, la ubicación de los distintos sectores de atención. Podía aumentarse el conocimiento de aspectos importantes como este mediante el uso de la pantalla para dar información que beneficie a los pacientes adicional a la de los puestos libres.

De esta forma, con una pantalla, un sistema de audio y una computadora podían atenderse los tres problemas utilizando un sistema que indique los puestos libres, información sobre temas relevantes y una trivia referida a esos temas para volver más amena la espera.

## **4. Solución propuesta**

Para dar una solución a los problemas vistos, se necesitó contar con una lista de puestos libres que sólo muestre aquellos a los cuales los pacientes puedan pasar. En caso de que ningún puesto esté libre, se optó por mostrar un único mensaje que lo indique. Cuando un puesto se libera u ocupa, la lista se actualiza y esto se refleja tanto en la pantalla como mediante una salida de audio, en caso de encontrarse habilitada.

Para mostrar la trivia se eligió usar una parte de la pantalla, en la que aparezca un enunciado con unas posibles respuestas. Una vez terminado el tiempo, se indica la respuesta correcta y aparece un texto que brinda la explicación.

### **4.1. Determinación de metodologías y tecnologías a utilizar**

A la hora de determinar las metodologías y tecnologías que se usarán en el desarrollo del sistema, se debieron tener en cuenta ciertos factores que permitieran distinguir las mejores opciones de entre las demás. Para ello se prestó atención a los factores enumerados a continuación:

Para elegir metodologías:

- Mejoramiento de la calidad
- Tiempo necesario para el desarrollo
- Aplicabilidad
- Capacidad de adaptación a cambios de requerimientos

Para elegir tecnologías:

- Sencillez de uso
- Grado de adopción
- Rendimiento
- Mantenibilidad
- Dificultad de aprendizaje
- Aplicabilidad

Para determinar qué metodologías y tecnologías se usarán, se seguirá un criterio que permita ver cuáles son las más ventajosas. Para ello se utilizará el de suma y ponderación cualitativas. Scriven (1991) explica lo siguiente sobre este criterio:

Utiliza símbolos como pesos, por ejemplo: E = esencial, \* = extremadamente valioso, # = muy valioso, + = valioso, | = marginalmente valioso y 0 = sin valor. La evaluación se realiza en tres pasos:

1. Construcción de la lista de atributos: Se establece una lista de atributos para los productos a evaluar. Luego se le otorga a cada uno de los atributos un peso de referencia máximo (en forma de símbolo). Se eliminan los atributos que obtienen el peso 0, pues se los considera irrelevantes.

2. Calificación de los atributos de cada producto: Se les asigna a los atributos de cada

producto un símbolo que no puede exceder el peso de referencia máximo correspondiente. Por ejemplo, si el peso de un atributo es #, se lo puede calificar como #, +, | o 0, pero no \*. Si un producto obtiene una calificación menor que E en algún atributo cuyo peso de referencia es E, el producto se elimina. Si todos los productos obtienen la misma calificación en algún atributo, el atributo se elimina.

3. Construcción del ranking: Para cada producto, se cuenta cuántos símbolos de cada tipo obtuvo, y según esos totales se ordenan los productos. A veces hace falta comparar más detalladamente algún par de productos. Por ejemplo: 3\*, 4#, 2+ y 1| es sin duda mejor que 2\*, 5#, 2+ y 1|, pero no está claro, a priori, si es mejor que 2\* y 8#. (p. 166)

#### 4.1.1. Metodologías

Para realizar el desarrollo del proyecto se deberán elegir algunas metodologías a seguir que permitan asegurar la calidad del sistema, el cumplimiento de objetivos y el uso eficaz del tiempo. A continuación se explican algunas metodologías de entre las cuales se elegirán aquellas que se van a usar.

- Kanban
- Test Driven Development
- Extreme Programming
- Desarrollo en cascada

El proceso Kanban consta de tres partes principales: el tablero, la lista y la tarjeta. El tablero contiene el flujo de trabajo, la lista contiene una serie de tarjetas, y la tarjeta es un producto que debe crearse o una tarea que debe realizarse. Las tareas se van trasladando a distintas listas conforme se vayan realizando. Las tarjetas pueden indicar en su reverso cosas como *checklists* o su fecha límite para ser completadas. Se asignan prioridades a las tarjetas y se logra organizar el trabajo estableciendo cuáles son las cosas que deben hacerse con mayor urgencia (Ries, 2022).

Por otra parte, los usuarios de la metodología denominada Test Driven Development ponen el énfasis en confeccionar tests unitarios para cada parte del código antes de realizar su implementación. Esta metodología se compone de 3 pasos que se repiten constantemente: escribir un test que falle, realizar la implementación para que se apruebe dicho test, y por último refactorizar esa implementación (Singh, 2021, pp. 19-20).

Extreme Programming (XP) es una metodología ágil enfocada en entregar el sistema que se necesita, cuando se lo necesita (Wells, 1999).

XP promueve que en un proyecto haya una retroalimentación, respeto, comunicación, simplicidad y coraje. La comunicación entre miembros de un equipo que usa esta metodología debe ser fluida, permitiendo que se pueda indicar incluso cuáles son aquellos aspectos del proyecto que pueden llegar a traer problemas a futuro. En lugar de planificar a detalle cada parte del sistema con mucha antelación, se prefiere tener una buena idea general del mismo y recién ahondar en los detalles de cada parte cuando sea el momento apropiado. La implementación se caracteriza por ser simple, buscando pasar los tests confeccionados para asegurarse del buen funcionamiento del sistema. En lugar de entregar el proyecto sólo después de haber implementado todos los requerimientos del cliente, se le puede brindar acceso a una versión anterior, que implemente parte de dichos requerimientos y

que permita empezar a usarla más rápido, gracias a haber sido desarrollada incrementalmente. Las funcionalidades principales son las elegidas para implementarse en los componentes iniciales del sistema, y los cambios realizados se integran de manera continua, almacenándose en el repositorio de proyecto incluso varias veces al día (Cluster, 2019).

La metodología de desarrollo en cascada consta de distintas fases secuenciales, define de manera clara los procesos del proyecto y permite determinar de antemano los requerimientos para su realización. Es el método tradicional para hacer proyectos, caracterizado por su baja capacidad de adaptación al cambio en los requerimientos. Las fases que lo componen son las de requisitos, análisis, diseño, implementación, prueba, despliegue y mantenimiento (Instituto Europeo de Posgrado, s.f.).

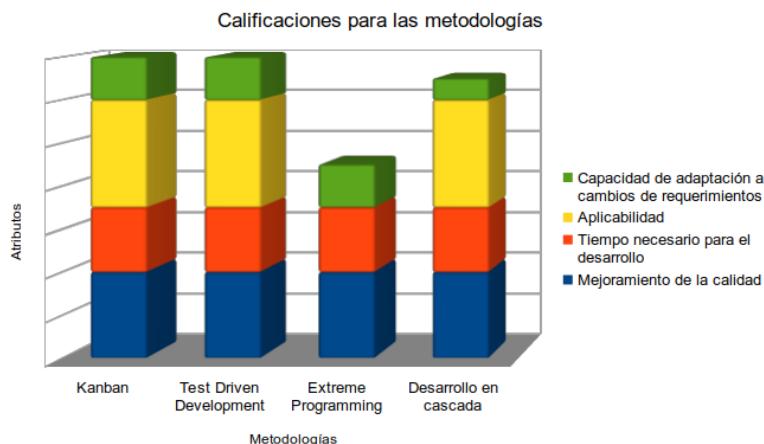
Entre las ventajas del desarrollo en cascada se encuentran que mantiene el trabajo organizado, define claramente los objetivos desde sus fases iniciales y permite identificar la fase en la cual se ha cometido algún error. Entre su desventajas están la obligatoriedad de terminar una fase para poder pasar a la siguiente, la demora en encontrar fallos y la dificultad para dividir en fases ordenadas el desarrollo de los proyectos de mayor tamaño (Risso, 2022).

En cuanto a la aplicabilidad, la metodología XP es la única de entre las destacadas previamente que no puede utilizarse ya que requiere contar con un equipo compuesto por más de una persona para poder realizar la programación por pares, la cual ha sido descripta como una de las doce prácticas originales de XP (Agile Alliance, s.f.).

Al asignar pesos como calificaciones a los atributos a evaluar, y luego hacer lo mismo a los atributos de cada metodología, se llegó a la distribución que se muestra en la fig. 4.

**Figura 4**

*Gráfico de calificaciones para metodologías*



Como se detalla en el Anexo I, Kanban y Test Driven development fueron las metodologías elegidas para el desarrollo del sistema.

#### 4.1.2. Lenguaje de programación del lado del cliente

Para el desarrollo del lado del cliente (frontend) se eligió entre los siguientes lenguajes de programación:

- JavaScript

- TypeScript

Hablando sobre JavaScript, MDN Web Docs (2022) detalla lo siguiente:

JavaScript (JS) es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js®, Apache CouchDB y Adobe Acrobat. JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional).

Por otra parte, el lenguaje TypeScript es un superconjunto de JavaScript, orientado a objetos y compilado (Shubel, 2022). Se diferencia de JavaScript en que permite especificar los tipos de datos que se pasan en el código, pudiendo reportar errores cuando los tipos de datos no coinciden en el momento de compilación (W3Schools, 2023).

Por su sencillez de uso se lo considera un lenguaje fácil de aprender. En cuanto a su grado de adopción, JavaScript es el lenguaje de programación más popular del mundo (W3Schools, 2023b).

En cuanto al rendimiento de JavaScript, un uso incompetente del mismo lleva a un empeoramiento de la velocidad del sistema. Es un lenguaje que permite escribir código muy ineficiente. Sin embargo, cada vez los motores para este lenguaje son más rápidos, al punto de que en la actualidad se encuentra bien optimizado y ya no es necesario preocuparse por cada línea de código (Wojnar, 2014). Usar JavaScript puede enriquecer la experiencia, pero excederse en su uso afecta negativamente a la velocidad de un sitio web (van Schie, s.f.).

Al ser un superconjunto de JavaScript, el código, sea este eficiente o no, puede dar un rendimiento similar usando TypeScript, por lo que sus ventajas e inconvenientes son similares en este apartado.

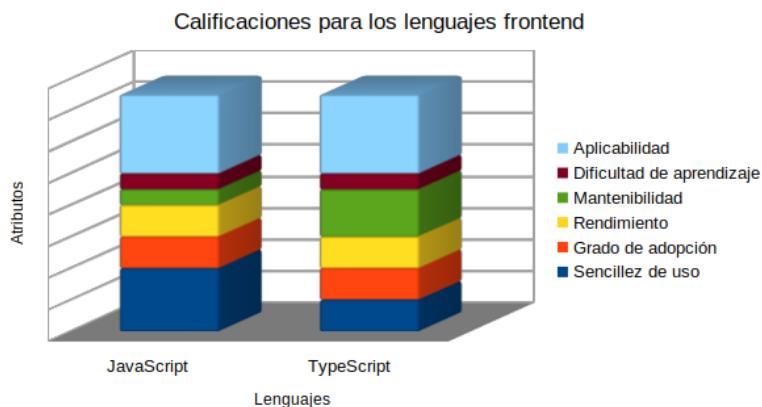
Al hacer un boceto, podría comenzarse a partir de un lienzo en blanco o de una hoja con guías y reglas. Al hacer un proyecto en JavaScript, se parte desde dicho lienzo en blanco, mientras que en TypeScript el tipado estático brinda ayuda adicional, similar a las guías y reglas. TypeScript es una gran herramienta para crear código que sea mantenable (Hao's learning log, 2018).

En cuanto a la dificultad de aprendizaje, si ya se cuenta con conocimientos de JavaScript, TypeScript no es muy difícil de aprender (Adams, 2022). Si no se cuenta con los fundamentos de programación necesarios esto puede ser un desafío, mientras que en otros casos puede ser algo extremadamente sencillo (Darling, 2022). Ahora bien, sigue siendo más fácil de aprender JavaScript que TypeScript (Njoku, 2022).

Al asignar pesos como calificaciones a los atributos a evaluar, y luego hacer lo mismo a los atributos de cada tecnología, se llegó a la distribución que se muestra en la fig. 5.

**Figura 5**

*Gráfico de calificaciones para lenguajes frontend*



Como se detalla en el Anexo I, luego de realizar el proceso de selección se determinó que JavaScript sería el lenguaje a usar para desarrollar el sistema.

#### 4.1.3. Framework para CSS

En cuanto a los frameworks para CSS, se eligió entre las siguientes opciones:

- Bootstrap
- Tailwind

Bootstrap permite crear sitios web rápidos y responsivos, contando con una caja de herramientas poderosa, extensible y con gran cantidad de características. Permite usar clases para personalizar la apariencia de componentes, por ejemplo modificando su posicionamiento, tamaño, colores y efectos (Bootstrap team y sus contribuidores, 2023).

Tailwind permite crear sitios web modernos rápidamente sin dejar el HTML. Permiten crear diseños a través de clases directamente en el maquetado. Es de bajo nivel, dando acceso a personalizar los componentes al grado de lograr que incluso conservando la misma paleta de colores y escalado de tamaño, puedan tener una apariencia completamente diferente. Además es pequeño, lo que lleva a que el cliente solo deba descargar archivos de poco tamaño (Tailwind Labs Inc., 2023b).

Bootstrap es fácil de aprender. Hay una gran cantidad de información en internet para verlo (A., 2022). Su sencillez en el uso, excelente documentación y soporte de la comunidad también son algunos puntos a su favor, y ha llegado a ser el framework para CSS más usado (Smith, 2021). Ahora bien, el código puede ser difícil de mantener y estar desorganizado, lo cual es una clara desventaja (Maliborski, 2015).

Tailwind permite crear diseños responsivos, y es sencillo de escribir y mantener código creado con este framework (Fitzgerald, 2022). También es fácil de aprender para quienes tengan un conocimiento previo de CSS (Abba, 2022). Aunque puede ser motivo de preocupación la mantenibilidad del código escrito en Tailwind, la realidad es que incluso leer líneas escritas en versiones antiguas puede ser simple y claro (Davies, 2023).

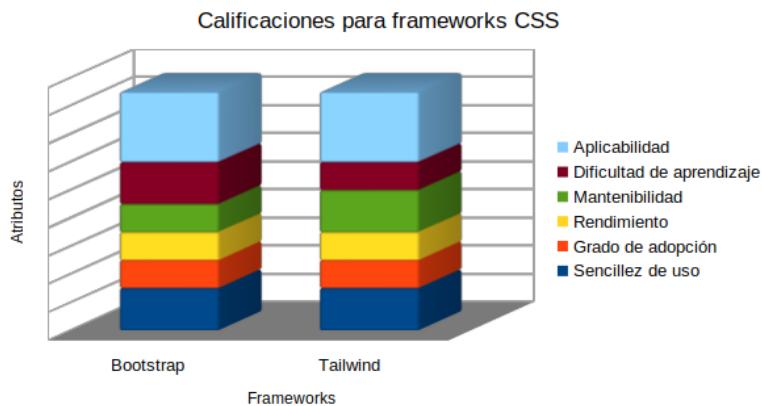
Ahora bien, largas líneas de nombres descriptivos de clases de Tailwind pueden volverse difíciles de comprender, por lo que en producción pueden traer complicaciones para ser mantenidas (Zhang,

2022). Este framework está enfocado en el rendimiento y busca usar el menor tamaño posible de archivos CSS generando solo aquel código que de verdad se usa en el proyecto (Tailwind Labs Inc., 2023a).

Al asignar pesos como calificaciones a los atributos a evaluar, y luego hacer lo mismo a los atributos de cada tecnología, se llegó a la distribución que se muestra en la fig. 6.

**Figura 6**

*Gráfico de calificaciones para frameworks CSS*



Como resultado del proceso de selección detallado en el Anexo I, se determinó que el framework Tailwind sería la opción elegida para el desarrollo del sistema.

#### 4.1.4. Framework o biblioteca frontend

Para desarrollar el lado del cliente (frontend) del sistema también se debió elegir un framework o biblioteca. Las opciones que se consideraron fueron las siguientes:

- Angular
- React
- Vue.js

Angular es usado por millones de desarrolladores, cuenta con diversas herramientas y permite entregar aplicaciones web con confianza (Google, 2023a). Los proyectos que lo utilizan pueden ser eficientes y sofisticados, y se caracterizan por ser aplicaciones de una sola página. Provee inyección de dependencias, para lo cual se utilizan los tipos de TypeScript (Google, 2023b). Además es fácil para aprender (W3Schools, 2023a).

React es una biblioteca para crear interfaces de usuario web y nativas, construyéndolas a partir de componentes. Con React se puede usar código y marcado para crear los componentes, siendo estos funciones de JavaScript. Esto contribuye a que sean fáciles de crear, mantener y eliminar (Meta Open Source, 2023).

Vue.js es un framework versátil para JavaScript con un buen rendimiento (You, 2023). Se usa para crear interfaces de usuario de manera práctica y rápida. Su curva de aprendizaje es la más sencilla en comparación con las otras dos opciones. Puede empezar a usarse de manera progresiva, por ejemplo

al migrar proyectos. Mientras React se centra más en un enfoque orientado a la programación en JavaScript, Vue.js está más enfocado en HTML y en los sistemas de plantillas (Manz, 2023).

Más de dos millones de sitios web están usando React. Los componentes en React son fáciles de depurar. Gracias a que usa el Virtual DOM, solo se actualizan los objetos que hayan cambiado, lo cual lleva a que el renderizado sea más veloz. (Galik, 2023).

Diferentes encuestas muestran resultados de uso distintos de cada una de estas tecnologías. La encuesta de Stack Overflow que mide el uso de las tecnologías más relevantes (fig. 7) pone en primer lugar a React, segundo a Angular y tercero a Vue.js; mientras que otra encuesta de Jet Brains enfocada en los frameworks web (fig. 8) ubica a React en el primer puesto, Vue.js en el segundo y Angular en el cuarto. El uso de Vue.js en particular puede verse, por ejemplo, en grandes compañías como Adobe y Netflix (Maciel, 2023). En cuanto a la cantidad de estrellas en GitHub, Vue.js está en primer lugar, seguido por React y luego por Angular (Daityari, 2023).

**Figura 7**

*Uso de las principales tecnologías desde 2016 hasta 2021*

Usage of Top Technologies from 2016 to 2021

Source: 2022 Stack Overflow Survey (16,085 Respondents)



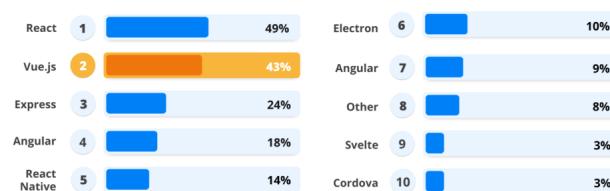
Nota: Extraída de Stack Overflow Survey (2022)

**Figura 8**

*10 Frameworks Web más usados en 2021*

Top 10 Most-used Web Frameworks in 2021

Source: Jet Brains Survey of 183 countries/regions



Nota: Extraída de Jet Brains Survey of 183 countries/regions (2021)

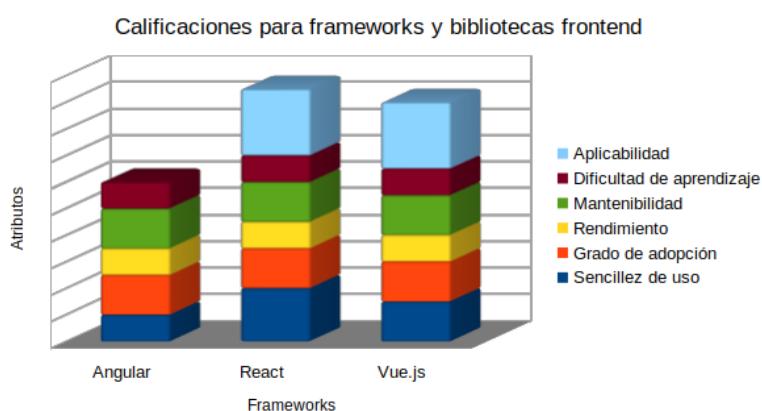
En Angular se usa una arquitectura basada en componentes que al ser fácilmente reemplazables permiten mantener el código de una manera más simple, y el basarse en TypeScript también contribuye a que esta tarea sea más amena. (AltexSoft, 2022). Por otra parte, Vue.js cuenta con su Composition API que permite manejar el estado y la lógica en componentes de una forma flexible y modular, lo que mejora su mantenibilidad (Theodosiou, 2023).

Para quienes tienen conocimiento de JavaScript, React será más fácil y productivo, mientras que Vue resultará una mejor opción para principiantes (Digitalya, 2021). Vue.js es especialmente sencillo de usar para aplicaciones más pequeñas, pero cuando los proyectos se vuelven más complejos, depender únicamente de plantillas se torna cada vez más difícil, por lo que React resulta más fácil de usar para aplicaciones más grandes (Stefanowicz, s.f.).

Al asignar pesos como calificaciones a los atributos a evaluar, y luego hacer lo mismo a los atributos de cada tecnología, se llegó a la distribución que se muestra en la fig. 9.

**Figura 9**

*Gráfico de calificaciones para frameworks y bibliotecas frontend*



Como resultado del proceso de selección detallado en el Anexo I, se determinó que React sería la opción elegida para el desarrollo del sistema.

#### 4.1.5. Lenguaje de programación y framework backend

Para el desarrollo del lado del servidor (backend) se eligió entre los siguientes lenguajes de programación y frameworks:

- Java y Spring
- Node.js® y Express.js

Java es un lenguaje de programación y plataforma para computación, lanzado inicialmente por Sun Microsystems en 1995 (Oracle, 2022). Algunas de sus características son su orientación a objetos, portabilidad, rendimiento y soporte para concurrencia mediante el uso de múltiples hilos de ejecución (JavaTpoint, 2023).

Spring es el framework para desarrollo de aplicaciones para Java empresarial más usado. El objetivo de quienes lo utilizan es obtener código fácilmente testeable, reusable y con un rendimiento elevado. Es modular y da soporte a la inyección de dependencias (tutorialspoint, 2023).

Node.js® es un entorno de ejecución de JavaScript orientado a eventos asíncronos que permite crear aplicaciones de red escalables. Contrastó con otros modelos de concurrencia basados en los hilos de Sistema Operativo los cuales suelen ser difíciles de usar y relativamente ineficientes, en su lugar usa un bucle de eventos (Open JS Foundation y contribuidores de Node.js®, 2023).

Express.js es un framework para Node.js® diseñado para construir aplicaciones web y que se convirtió en el más usado para este entorno de ejecución. Es minimalista, rápido y muy fácil de usar (Kinsta Inc., 2022).

Java es fácil de aprender y depurar ya que fue diseñado para también ser fácil de usar (IBM, 2023). Esto es así aún siendo un lenguaje más complejo que JavaScript (Eboka, 2022). Datos de encuestas de Stack Overflow muestran que Java ya no es el lenguaje más usado aunque permanece entre los más populares, pasando de un 41,1 % de uso en 2019 a un 33,27 % en 2022 (Palmer, 2022). En cuanto al rendimiento, Java es bastante rápido pero utiliza mucha memoria (Akki, 2020).

Spring es muy usado para el desarrollo profesional con Java debido a su sencillez de uso, flexibilidad, documentación y soporte de la comunidad (Flexiple Tech India Private Limited, 2023). Ahora bien, su curva de aprendizaje es empinada, por lo que resulta difícil para los desarrolladores relativamente inexpertos (adservio, 2022). Al usar inyección de dependencias facilita el mantenimiento del código (Turing, 2023).

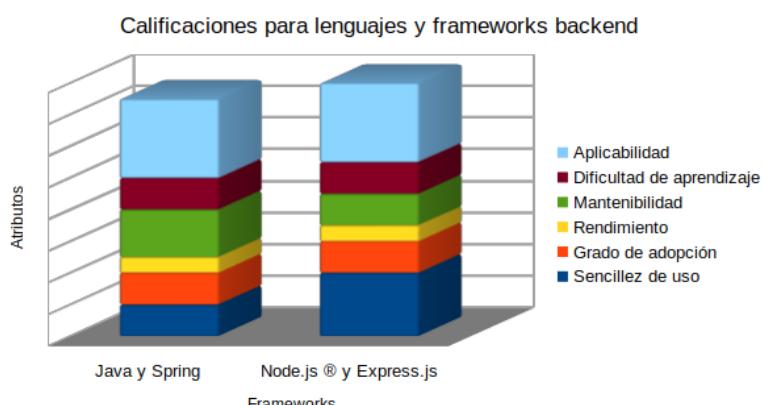
Express.js permite manejar pedidos usando funciones modulares, pequeñas y mantenibles. Es fácil de aprender ya que utiliza JavaScript, lo que permite usarlo sin necesitar aprender otro lenguaje (Azam, 2021).

Node.js® es considerado el framework más usado por los desarrolladores en todo el mundo, según una encuesta del 2022 (Vailshery, 2022). El uso de funciones anónimas puede hacer que el código sea poco mantenible, por lo que es conveniente evitarlas y también estructurar las funciones callbacks (Reed, 2015). Incluso para un desarrollador Javascript Junior, Node.js® se considera una tecnología que requiere menos esfuerzo y tiempo de aprendizaje (Kumar, 2023).

Al asignar pesos como calificaciones a los atributos a evaluar, y luego hacer lo mismo a los atributos de cada tecnología, se llegó a la distribución que se muestra en la fig. 10.

**Figura 10**

*Gráfico de calificaciones para lenguajes y frameworks backend*



Como resultado del proceso de selección detallado en el Anexo I, se determinó que Node.js® junto con Express.js serían las opciones elegidas para el desarrollo del sistema.

#### 4.1.6. Base de datos

Para el desarrollo de la base de datos se eligió entre los siguientes sistemas gestores:

- MongoDB
- MySQL
- PostgreSQL

MongoDB es una base de datos basada en documentos escalable y flexible. Su modelo de documentos es simple de aprender y de usar, de hecho los datos se almacenan en documentos con un formato similar a JSON (MongoDB, 2023). Es considerada la base de datos moderna más usada por el quinto año consecutivo (MongoDB, Inc., 2023).

MySQL es el sistema gestor de base de datos de código abierto más popular. Utiliza el modelo relacional basado en tablas y utiliza archivos físicos optimizados para la velocidad. Para brindar acceso a las bases de datos se usa el lenguaje SQL. Este sistema es muy rápido y fácil de usar (Oracle, 2023).

PostgreSQL es un sistema gestor de base de datos objeto-relacional con un buen rendimiento y confiabilidad (The PostgreSQL Global Development Group, 2023b). Intenta mantener conformidad con el estándar SQL sin que ello contradiga características tradicionales ni lleve a malas decisiones en cuanto a su arquitectura. De hecho contiene una gran cantidad de características que buscan ayudar a los desarrolladores a construir aplicaciones (The PostgreSQL Global Development Group, 2023a).

MongoDB puede manejar grandes cantidades de datos sin estructurar mucho más rápido que MySQL y PostgreSQL, pero tiene un alto consumo de memoria e incluso puede tener algún problema ocasional de velocidad en entornos en la nube (Smallcombe, 2023). En este sistema cada documento se almacena en la base de datos junto a objetos de datos relacionados y encapsulados juntos, lo que permite extraer datos con una única operación. De esta forma MongoDB aumenta el rendimiento, la velocidad y la sencillez de mantenimiento (Manthena, 2023).

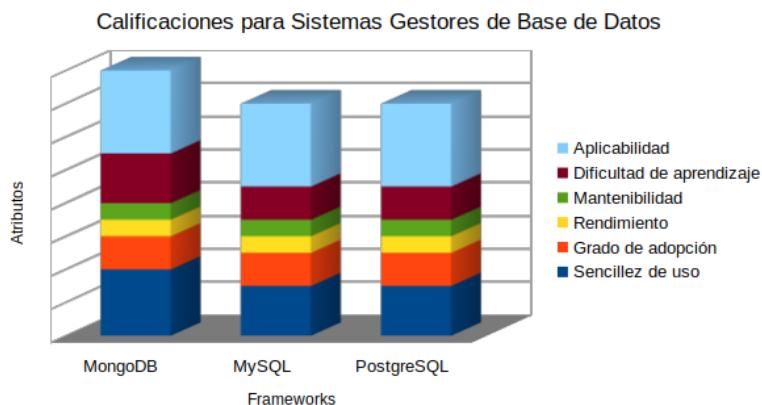
Se considera que MySQL es confiable y mantenible, lo que permite que los administradores de bases de datos no tengan que perder tiempo solucionando problemas de rendimiento y de tiempo de inactividad (DBQuest, Inc., 2012). El lenguaje SQL, usado por MySQL y PostgreSQL para acceder a las bases de datos es fácil de aprender, pudiendo tomar unas pocas semanas en caso de contar con conocimientos de programación previos (Thinkful, Inc., 2023).

PostgreSQL tiene bajos costos operativos, es fácil de usar y de mantener (Washington, 2022). Su cuota de mercado es de 17,32 %, mientras que la de MySQL es de 44,68 % (Lyon, 2023).

Al asignar pesos como calificaciones a los atributos a evaluar, y luego hacer lo mismo a los atributos de cada tecnología, se llegó a la distribución que se muestra en la fig. 11.

**Figura 11**

*Gráfico de calificaciones para Sistemas Gestores de Base de Datos*



Como resultado del proceso de selección detallado en el Anexo I, se determinó que MongoDB sería la opción elegida para el desarrollo del sistema.

#### 4.1.7. Testing

Para el testing del proyecto se eligió entre los siguientes frameworks:

- Jasmine
- Jest
- JUnit
- Mocha

Jasmine es un veloz framework para desarrollo guiado por comportamiento que permite realizar testing de código escrito en JavaScript. Puede usarse para realizar los tests tanto para el navegador como para Node.js ®. Tiene una sintaxis sencilla para poder escribir fácilmente los tests (Jasmine, 2023b). En este framework se escriben specs, que contienen expectativas que prueban el estado del código (Jasmine, 2023c). Solo ejecuta las specs en paralelo cuando se cuenta con por lo menos la versión 5.0 del paquete de NPM Jasmine y se pasa el argumento de línea de comandos correspondiente (Jasmine, 2023a).

En cuanto a Jest, este framework está centrado en la simplicidad y tiene un alto rendimiento gracias a que los tests se ejecutan de forma paralela. Es usado por muchas personas, llegando a encontrarse en más de 3 millones de repositorios públicos de GitHub (Meta Platforms, Inc. y afiliados, 2023).

JUnit es un framework de testing unitario para Java. Es simple, por lo que toma menos tiempo (Tutorials Point, 2023). Provee a los desarrolladores anotaciones, assertions y test runners para poder crear y correr las pruebas unitarias y así contribuir a mejorar la calidad del código (Gaba, 2023). Los tests en JUnit son ejecutados rápidamente, lo que permite correr una gran suite de pruebas en un corto período de tiempo (Code Intelligence, 2023).

Mocha es un framework de testing para JavaScript repleto de características que corre tanto en Node.js® como en el navegador, haciendo que el testing asíncrono sea simple. Los tests corren de

manera serializada, lo que lleva a que los reportes sean precisos, relacionando las excepciones sin capturar con los casos de testeo correctos (OpenJS Foundation and Mocha contributors, 2023). Tiene una sintaxis simple y es veloz (Ieremenko, 2023).

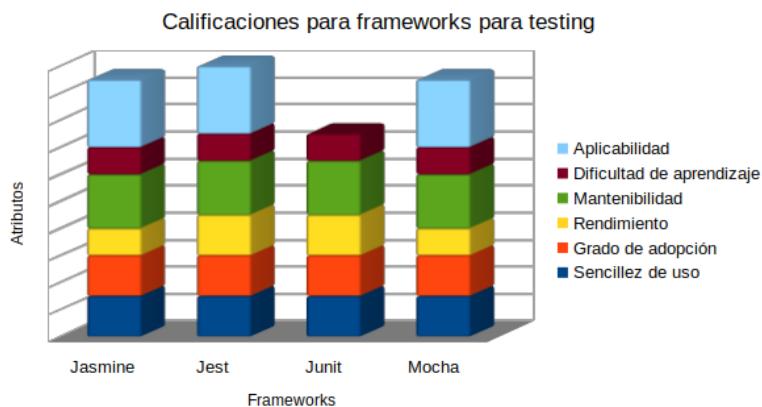
En cuanto a la aplicabilidad, Jest, Mocha y Jasmine pueden ser usados para testear el código escrito en JavaScript, mientras que JUnit está destinado a testear el código escrito en Java. Esto implica que en el lado del cliente podían ser usados cualquiera de los tres primeros, mientras que en el backend se debería elegir entre ellos o JUnit dependiendo del lenguaje de programación seleccionado previamente.

En cuanto al grado de adopción, Greif y Benitte (2020) indican que 68 % de los encuestados usaron Jest, 53 % Mocha y 42 % Jasmine durante 2020. Por otra parte, JUnit es uno de los frameworks de testing para Java más populares (Kanai, 2022).

Al asignar pesos como calificaciones a los atributos a evaluar, y luego hacer lo mismo a los atributos de cada tecnología, se llegó a la distribución que se muestra en la fig. 12.

**Figura 12**

*Gráfico de calificaciones de frameworks para hacer testing del código*



Como resultado del proceso de selección detallado en el Anexo I, se determinó que Jest sería la opción elegida para el desarrollo del sistema.

## 4.2. Análisis de requisitos

El sistema consta de dos requisitos principales: en primer lugar, aportar información sobre qué puestos se encuentran libres, actualizándose de manera automática y permitiendo a quienes lo controlen modificar dicha información; y en segundo lugar, brindar un método de recreación a través de una trivia, la cual también deberá ser editable por quienes controlen el sistema.

Por lo tanto, resulta necesario contar con la forma de efectuar altas, bajas, modificaciones y consultas para los registros de ambas partes del sistema, las cuales funcionarán gracias a la interacción entre el cliente web, el servidor y la base de datos, que almacenará información tanto de los puestos como de la trivia.

A continuación se indicarán con mayor detalle los requisitos de la sección de trivia y de la correspondiente para los puestos libres y sus editores.

#### **4.2.1. Puestos**

Quienes estén esperando para ir a un puesto necesitan una forma de saber si hay puestos libres y cuáles son. Para ello en la pantalla principal se requiere mostrar los nombres de los puestos libres y dar esa información también a través de una salida de audio.

Esa información debe actualizarse automáticamente y reflejar los datos ingresados por quienes controlen el sistema, que en principio deberían ser los empleados que ocupen los puestos, además de su jefe o el encargado del sistema. Esos datos deben poder ingresarse con una interfaz de usuario con opciones para ocupar o liberar puestos de una manera sencilla y rápida, minimizando la curva de aprendizaje.

Además, en caso de querer agregar o quitar puestos, o simplemente contar con un editor más elaborado, una interfaz alternativa y más avanzada es necesaria. De esta forma al usar por primera vez el sistema se podrán agregar los puestos y editar su información desde el editor de puestos avanzado, y una vez configurado los empleados podrán usar el editor simple para indicar cuando sus puestos se liberan u ocupen.

Los datos deberán guardarse en una base de datos para que los mismos queden almacenados de forma persistente. En caso de apagarse la computadora en la cual corra la base de datos, estos no se perderán, contrario a lo que ocurriría en caso de almacenarse en la memoria RAM.

#### **4.2.2. Trivia**

La trivia constará de preguntas con varias respuestas posibles, algunas correctas y otras incorrectas, y una explicación de por qué son correctas las respuestas así indicadas. Contar con animaciones para las transiciones entre preguntas y respuestas resulta apropiado para dar una mejor experiencia al usuario. Las trivias se podrán ver desde la pantalla principal junto con los puestos.

Cada trivia podrá ser ingresada por los encargados del sistema. Deberá proveerse una interfaz de usuario que cuente con opciones para eliminar trivias y editarlas, modificando campos como la pregunta, las respuestas, cuáles son correctas y la explicación.

### **4.3. Especificaciones del sistema**

Para la base de datos se optó por usar MongoDB. Esto implicó que la conexión entre el servidor y la base de datos se realice a través de Mongoose. Este servidor se implementó con tecnologías basadas en JavaScript, como NodeJS® y Express.js. El testing se realizó usando Jest. Para posibilitar el despliegue usando contenedores se usó Docker junto con Kubernetes.

Para el frontend se eligió crear un cliente web que fuera accesible desde un navegador de internet. Al igual que en el backend se usaron tecnologías basadas en JavaScript, en este caso React junto con Fluent UI, Tailwind y Vite.

### **4.4. Análisis del sistema**

La información necesaria para ambas partes del sistema se representó a través de documentos en la base de datos no relacional hecha con MongoDB. Estos documentos podían contar con dos estructuras, dependiendo si se trataba de trivias o de puestos.

Para los puestos, se usaron el nombre (cadena de caracteres) y su disponibilidad (booleano) como campos.

Para las trivias, se usaron la pregunta (cadena de caracteres), la explicación (cadena de caracteres) y las respuestas (vector de cadena de caracteres) como campos.

Por otra parte, un router fue usado para que al ingresar pedidos HTTP a determinadas URLs, el controlador apropiado lo procese. Un controlador para los pedidos relacionados con las trivias y otro para los puestos fueron usados. A su vez cada uno constó de una función asíncrona que permitía, según su caso, gestionar pedidos de altas, bajas, modificaciones o consultas.

Mediante Jest se realizaron tests para verificar el buen funcionamiento del sistema, los mismos abarcan los pedidos tanto para trivias como para puestos de altas, bajas, modificaciones y consultas.

En cuanto al frontend, su principal componente incluye tanto una sección para ver las trivias de manera secuencial como otra para indicar los puestos que se encuentren libres. Para actualizar los puestos de forma automática se recurrió a la técnica de polling a través de Tanstack Query, el cual en intervalos recurrentes hace que el cliente pida datos nuevos al servidor para poder contar con información relevante sin necesidad de recargar la página manualmente.

El editor simple para puestos permite elegir a través de una lista desplegable el puesto a modificar y con un botón pasar su estado de libre a ocupado.

El editor avanzado para puestos permite agregar, quitar y editar puestos. Los datos que deja modificar son los nombres y la disponibilidad de cada puesto.

El editor de trivias permite agregar, quitar y editar trivias. En cada trivia pueden modificarse la pregunta, la explicación, las respuestas y la validez de cada una.

## 5. Resultados

El sistema implementado consta de los componentes descriptos anteriormente, por lo que en esta sección podrán verse el aspecto que tomaron, detalles sobre su funcionalidad y características finales.

En la pantalla principal (fig. 13), ideada para usarse en una pantalla que sea vista por quienes estén esperando para pasar a un puesto libre, muestra aquellos puestos que estén libres y la sección de la trivia.

**Figura 13**

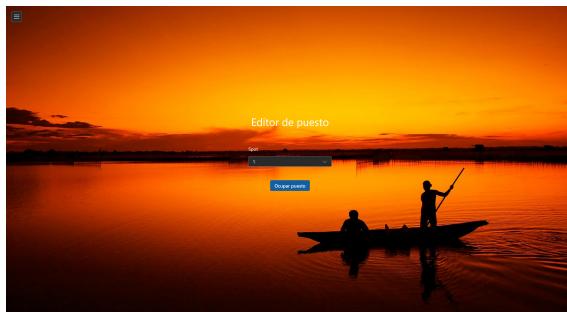
*Interfaz de la pantalla principal del sistema*



En el editor de puesto (fig. 14), pensado para ser usado por cada empleado encargado de un puesto, permite que se elija un puesto y se indique su liberación u ocupación, lo cual se ve reflejado en la pantalla principal que se actualiza de manera automática.

**Figura 14**

*Interfaz del editor de puesto*



En el editor de puestos avanzado (fig. 15), necesario para un administrador del sistema, permite no solo liberar y ocupar puestos sino también cambiar sus nombres, agregar e incluso remover puestos.

**Figura 15**

*Interfaz del editor de puestos avanzado*



En el editor de trivias (fig. 16), necesario para un administrador del sistema, se pueden agregar, eliminar y editar trivias. Entre los campos que pueden editarse se encuentran las preguntas, respuestas y explicaciones de las trivias.

**Figura 16**

*Interfaz del editor de trivias*



Se incluyó un fondo de pantalla para las interfaz de usuario para que la misma tuviera un mejor aspecto visual, el mismo en primera instancia se intentó que estuviera relacionado con la medicina o incluso el mismo Hospital de Clínicas, sin embargo al dar un aspecto poco conveniente se optó por un fondo que, aunque pudiera parecer fuera de lugar (lo mismo podría decirse de la trivia, que es en realidad una funcionalidad fundamental del sistema), brinda un aspecto más ameno a una interfaz que, de otra forma, podría verse desagradable.

En el repositorio ubicado en el siguiente enlace: <https://github.com/charrassebastian/sirhc> se encuentra tanto el código del sistema como también archivos de configuración para realizar el despliegue con las tecnologías Docker y Kubernetes.

Al usar contenedores se pueden aislar las capas del sistema (cliente, servidor y base de datos). Para contar con una versión disponible como parte de un portafolio de proyectos, en un servidor presente en una red local se realizó un despliegue que al momento de redactar este informe puede accederse a través del siguiente enlace: <https://sirhc.sebastiancharras.tech>.

Los tests implementados para la API permiten contar con un mayor grado de confianza en el sistema, asegurando el buen funcionamiento de esta parte esencial del backend, necesaria para que pueda usarse sin problemas. El testing no incluyó, por ejemplo, a los componentes del cliente, sino que se centró exclusivamente en la API, particularmente en el servidor.

La base de datos usa MongoDB, para lo que puede aprovecharse la imagen oficial de dicho software para crear un contenedor y correrlo de manera sencilla. Las versiones más recientes requieren soporte de instrucciones AVX en el procesador, por lo que en computadoras más antiguas y de gamas más bajas es posible encontrar incompatibilidades debido a este requisito. Sin embargo, usando una versión antigua es posible resolver este problema, como por ejemplo la versión 4.

## 6. Conclusiones

La experiencia de desarrollo fue enriquecedora, contribuyendo a adquirir y reforzar conocimientos en diversas áreas del desarrollo, tanto sobre tecnologías como de metodologías.

Una de las cosas más importantes que se obtuvo con este proyecto fue aprender sobre la metodología Test Driven Development. Esta pone su foco en crear tests unitarios para cada código escrito, incluso escribiendo los tests antes que las implementaciones. De esta forma se asegura la calidad, previniendo la entrega de productos defectuosos.

Ahora bien, aunque sea muchas veces vista de una manera muy positiva debido a las razones ya mencionadas, esta metodología de desarrollo puede llevar a que se tome más tiempo en crear un programa que realizando un uso menos frecuente de tests. Durante el desarrollo de este sistema, utilizar esta metodología en frontend demostró ser particularmente complejo en comparación con su uso en el backend, lo que llevó a que se decidiera enfocar la escritura de tests en asegurar el buen funcionamiento de la API, dejando de lado el testeo de los componentes del cliente web.

Por otra parte, las tecnologías usadas contribuyeron a la creación del sistema sin requerir cantidades de tiempo de aprendizaje demasiado grandes. Su sencillez de uso y amplia documentación, así como la extensa cantidad de bibliotecas disponibles hicieron que la implementación del sistema se llevara a cabo de una manera rápida.

Aunque la cantidad de tecnologías y de metodologías en las que se tuvo que ahondar para realizar el desarrollo fue mayor que en otros proyectos realizados, las elecciones tomadas hicieron que esto no llevara a un atraso en los tiempos de desarrollo ni a una merma en las características implementadas. Los nuevos conocimientos adquiridos y la experiencia obtenida son algunas de las cosas más útiles que se consiguieron en este proyecto.

## Referencias

- A., J. (2022). *What Is Bootstrap?* <https://www.hostinger.com/tutorials/what-is-bootstrap/>
- Abba, I. (2022). *How to Use Tailwind CSS to Rapidly Develop Snazzy Websites.* <https://kinsta.com/blog/tailwind-css/>
- Adams, D. (2022). *Learn TypeScript – The Ultimate Beginners Guide.* <https://www.freecodecamp.org/news/learn-typescript-beginners-guide/>
- adservio. (2022). *What is Spring Boot?* <https://www.adservio.fr/post/what-is-spring-boot>
- afp. (2014). "Preguntados": el juego argentino que conquista teléfonos en América Latina. <https://www.abc.com.py/ciencia/preguntados-el-juego-argentino-que-conquista-telefonos-en-america-latina-1243640.html>
- Agile Alliance. (s.f.). *Pair Programming.* <https://www.agilealliance.org/glossary/pairing>
- Akki, A. (2020). *Why is Java slower than C++ programs?* <https://www.tutorialspoint.com/Why-is-Java-slower-than-Cplusplus-programs>
- AltexSoft. (2022). *The Good and the Bad of Angular Development.* <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/>
- Azam, S. (2021). *What is Express JS.* <https://linuxhint.com/what-is-express-js/>
- Bootstrap team y sus contribuidores. (2023). *Build fast, responsive sites with Bootstrap.* <https://getbootstrap.com/>
- Calle, I. (s.f.). *Petición · Las nuevas pantallas de los hospitales discriminan a las personas ciegas como yo.* <https://www.change.org/p/las-nuevas-pantallas-de-los-hospitales-discriminan-a-las-personas-ciegas-como-yo-eruizescudero>
- Cluster, K. (2019). *Agile Project Management Learn How To Manage a Project With Agile Methods, Scrum, Kanban and Extreme Programming.*
- Code Intelligence. (2023). *Best Practices for JUnit Testing.* <https://www.code-intelligence.com/junit-testing>
- Daityari, S. (2023). *Angular vs React vs Vue: Which Framework to Choose.* <https://www.codeinwp.com/blog/angular-vs-vue-vs-react/>
- Darling, G. (2022). *How Hard Is It To Learn TypeScript.* <https://thecodebytes.com/how-hard-is-it-to-learn-typescript/>
- Davies, O. (2023). *Long-term maintainability with utility classes and Tailwind CSS.* <https://www.oliverdavies.uk/archive/2023/01/19/long-term-maintainability-with-utility-classes-and-tailwind-css>
- DBQuest, Inc. (2012). *Top Reasons to Use MySQL.* <https://www.databasequest.com/index.php/product-service/mysql-dbquest>
- Digitalya. (2021). *Vue vs React: What to choose in 2021?* <https://dev.to/digitalyaops/vue-vs-react-what-to-choose-in-2021-3bc>
- Eboka, P. (2022). *Java vs JavaScript.* <https://www.lighthouselabs.ca/en/blog/java-vs-javascript>
- Etermax. (2023). *Preguntados.* [https://play.google.com/store/apps/details?id=com.etermax.preguntados-lite&hl=es\\_AR&gl=US](https://play.google.com/store/apps/details?id=com.etermax.preguntados-lite&hl=es_AR&gl=US)
- Fitzgerald, A. (2022). *Tailwind CSS: What It Is, Why Use It & Examples.* <https://blog.hubspot.com/website/what-is-tailwind-css>
- Flexiple Tech India Private Limited. (2023). *Intro to Java Spring.* <https://flexiple.com/spring/deep-dive/>

- Gaba, I. (2023). *What Is JUnit: An Overview of the Best Java Testing Framework*. <https://www.simplilearn.com/tutorials/java-tutorial/what-is-junit>
- Galik, O. (2023). *When and Why You Should Use React*. <https://www.uptech.team/blog/why-use-react>
- Google. (2023a). *Deliver web apps with confidence*. <https://angular.io/>
- Google. (2023b). *Introduction to the Angular Docs*. <https://angular.io/docs>
- Greif, S., & Benitte, R. (2020). *State of JS 2020: Testing*. <https://2020.stateofjs.com/en-US/technologies/testing/>
- Hao's learning log. (2018). *TypeScript: the key to a maintainable JavaScript codebase*. <https://blog.hao.dev/typescript-the-key-to-a-maintainable-javascript-codebase>
- IBM. (2023). *Advantages of Java*. <https://www.ibm.com/docs/en/aix/7.1?topic=monitoring-advantages-java>
- Ieremenko, D. (2023). *Why Mocha is our ‘5-star’ Javascript unit testing framework*. <https://www.mindk.com/blog/javascript-testing-framework/>
- Instituto Europeo de Posgrado. (s.f.). *Metodología Waterfall: Modelo de gestión de proyectos en cascada*. <https://www.iep.edu.es/metodologia-waterfall/>
- Jasmine. (2023a). *Frequently Asked Questions*. <https://jasmine.github.io/pages/faq.html>
- Jasmine. (2023b). *Jasmine Documentation*. <https://jasmine.github.io/>
- Jasmine. (2023c). *Your first suite*. [https://jasmine.github.io/tutorials/your\\_first\\_suite](https://jasmine.github.io/tutorials/your_first_suite)
- JavaTPoint. (2023). *Features of Java*. <https://www.javatpoint.com/features-of-java>
- Kanai, S. (2022). *JUnit: A Complete Guide*. <https://www.headspin.io/blog/junit-a-complete-guide>
- Kinsta Inc. (2022). *¿Qué es Express.js? Todo lo que Debes Saber*. <https://kinsta.com/es/base-de-conocimiento/que-es-express/>
- Kumar, P. (2023). *Node.js for Beginners: How to Get Started*. <https://www.simplilearn.com/nodejs-for-beginners-article>
- La Capital. (2015). *Instalan Wi-Fi y pantallas LED en nueve colectivos*. <https://www.lacapital.com.ar/la-ciudad/instalan-wi-fi-y-pantallas-led-nueve-colectivos-n474477.html>
- Lyon, J. (2023). *PostgreSQL - Market Share, Competitor Insights in Relational Databases*. <https://6sense.com/tech/relational-databases/postgresql-market-share>
- Maciel, E. (2023). *Will Vue Become the Most Popular Front-end Framework in 2023?* <https://www.scalablepath.com/front-end/vue-js-popularity-framework>
- Maliborski, M. (2015). *Why You Should Stop Misusing Bootstrap*. <https://10clouds.com/blog/web/stop-misusing-bootstrap/>
- Manthena, K. (2023). *How MongoDB Improves Database Optimization at Optimal Costs*. <https://blog.vsoftconsulting.com/blog/how-mongodb-improves-database-optimization-at-optimal-costs>
- Manz. (2023). *¿Qué es Vue?* <https://lenguajejs.com/vuejs/introduccion/que-es-vue/>
- MDN Web Docs. (2022). *JavaScript*. <https://developer.mozilla.org/es/docs/Web/JavaScript>
- Meta Open Source. (2023). *React*. <https://es.react.dev/>
- Meta Platforms, Inc. y afiliados. (2023). *Jest · Delightful JavaScript Testing*. <https://jestjs.io/>
- MINUTOUNO. (2014). *Furor por "Preguntados", un nuevo y adictivo juego para iOS y Android*. <https://www.minutouno.com/tecnologia/videojuegos/furor-preguntados-un-nuevo-y-adictivo-juego-ios-y-android-n319173>
- MongoDB, I. (2023). *What Is MongoDB?* <https://www.mongodb.com/what-is-mongodb>
- MongoDB, Inc. (2023). *Build the next big thing*. <https://www.mongodb.com/>

- Njoku, M. U. (2022). *How Hard Is It to Learn JavaScript?* <https://careerkarma.com/blog/is-javascript-hard-to-learn/>
- Open JS Foundation y contribuidores de Node.js®. (2023). *Acerca de Node.js®*. <https://nodejs.org/es/about>
- OpenJS Foundation and Mocha contributors. (2023). *Mocha - the fun, simple, flexible JavaScript test framework*. <https://mochajs.org/>
- Oracle. (2022). *What is Java technology and why do I need it?* [https://www.java.com/en/download/help/whatis\\_java.html](https://www.java.com/en/download/help/whatis_java.html)
- Oracle. (2023). *What is MySQL?* <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>
- Palmer, R. (2022). *Java Is in Decline, and There's Data To Prove It*. <https://devm.io/java/java-decline-kotlin>
- Reed, B. (2015). *NodeJS: Building a Maintainable Codebase*. <https://hub.packtpub.com/nodejs-building-maintainable-codebase/>
- Ries, J. (2022). *Agile Project Management: 3 Books in 1: The Complete Guide to Agile Project Management, Methodology & Software Development*.
- Risso, I. (2022). *Domina el modelo en cascada y potencia al máximo tus proyectos de software*. <https://www.crehana.com/blog/transformacion-digital/modelo-en-cascada/>
- Scriven, M. (1991). *Evaluation Thesaurus*. Sage Publications.
- Shubel, M. (2022). *What is TypeScript?* <https://thenewstack.io/what-is-typescript/>
- Singh, S. K. (2021). *Test Driven Development Using JavaScript and Jest*.
- Smallcombe, M. (2023). *MongoDB vs. MySQL: Detailed Comparison of Performance and Speed*. <https://www.integrate.io/blog/mongodb-vs-mysql/>
- Smith, M. (2021). *Is Bootstrap hard to learn? 6 crucial things to consider*. <https://developerpitstop.com/is-bootstrap-hard-to-learn/>
- Stefanowicz, P. (s.f.). *Vue.js vs. React: A Comparison of Top JavaScript Frameworks*. <https://www.stxnext.com/blog/vue-vs-react-comparison/>
- Tailwind Labs Inc. (2023a). *Optimizing for Production*. <https://tailwindcss.com/docs/optimizing-for-production>
- Tailwind Labs Inc. (2023b). *Rapidly build modern websites without ever leaving your HTML*. <https://tailwindcss.com/>
- The PostgreSQL Global Development Group. (2023a). *About*. <https://www.postgresql.org/about/>
- The PostgreSQL Global Development Group. (2023b). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. <https://www.postgresql.org/>
- Theodosiou, P. (2023). *Comparing Vue and React in 2023: Pros and Cons*. <https://dev.to/ptheodosiou/comparing-vue-and-react-in-2023-pros-and-cons-10nl>
- Thinkful, Inc. (2023). *How Hard is it to Learn SQL?* <https://www.thinkful.com/blog/how-hard-is-it-to-learn-sql/>
- Turing. (2023). *Spring vs Spring Boot: An in-depth Comparison (Updated 2023)*. <https://www.turing.com/kb/spring-vs-spring-boots-best-web-apps>
- Tutorials Point. (2023). *JUnit - Overview*. [https://www.tutorialspoint.com/junit/junit\\_overview.htm](https://www.tutorialspoint.com/junit/junit_overview.htm)
- tutorialspoint. (2023). *Spring Framework - Overview*. [https://www.tutorialspoint.com/spring/spring\\_overview.htm](https://www.tutorialspoint.com/spring/spring_overview.htm)

- Vailshery, L. S. (2022). *Most used web frameworks among developers worldwide, as of 2022*. <https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/>
- van Schie, T. (s.f.). *JavaScript: the main culprit of slow websites*. <https://www.avivasolutions.nl/en/techblog/javascript-the-main-culprit-of-slow-websites>
- W3Schools. (2023a). *AngularJS Tutorial*. <https://www.w3schools.com/angular/>
- W3Schools. (2023b). *JavaScript Tutorial*. <https://www.w3schools.com/js/>
- W3Schools. (2023). *TypeScript Introduction*. [https://www.w3schools.com/typescript/typescript\\_intro.php](https://www.w3schools.com/typescript/typescript_intro.php)
- Washington, V. (2022). *A Guide to Getting Started with PostgreSQL*. <https://www.topcoder.com/thrive/articles/a-guide-to-getting-started-with-postgresql>
- Wells, D. (1999). *When should Extreme Programming be Used?* <http://www.extremeprogramming.org/when.html>
- Wojnar, A. (2014). *JavaScript is slow*. <https://kariera.future-processing.pl/blog/javascript-is-slow/>
- You, E. (2023). *The Progressive JavaScript Framework*. <https://vuejs.org/>
- Zhang, S. (2022). *Why I don't use Tailwind CSS in Production*. <https://blog.shimin.io/why-i-dont-use-tailwind-in-production/>

## Bibliografía

- Authors, T. K. (2023a). *Install and Set Up kubectl on Linux*. <https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>
- Authors, T. K. (2023b). *minikube start*. <https://minikube.sigs.k8s.io/docs/start/>
- Ayako Sayama. (s.f.). *Best Practices for Writing Clean and Maintainable TypeScript Code*. <https://blog.bitsrc.io/best-practices-for-writing-clean-and-maintainable-typescript-code-1cc6a7f029dc>
- Corsi, D. P. (2008). *Sistema Inteligente para la selección de un Administrador de Contenidos a través de la Web* [Tesis de maestría, Universidad Tecnológica Nacional].
- Harbuzava, K. (2021). *Vue JS vs React: What is Trending? [Detailed Guide]*. <https://flatlogic.com/blog/vue-vs-react-what-is-easier-what-is-trending-detailed-guide-with-all-2021/>
- Inc., D. (2023a). *Install Docker Engine on Ubuntu*. <https://docs.docker.com/engine/install/ubuntu/>
- Inc., D. (2023b). *Linux post-installation steps for Docker Engine*. <https://docs.docker.com/engine/install/linux-postinstall/>
- Sergio Gordillo. (2021). *Angular*. <https://www.algoritmosalvaje.com/angular/>

## Anexos

### 6.1. Selección de metodologías y tecnologías

#### 6.1.1. Selección de metodologías

Para realizar la selección se usaron los mismos símbolos como pesos indicados en el ejemplo. Los pesos de referencia de los atributos para este caso son los siguientes:

- Mejoramiento de la calidad - \*
- Tiempo necesario para el desarrollo - #
- Aplicabilidad - E
- Capacidad de adaptación a cambios de requerimientos - +

A continuación se indican los símbolos asignados a cada metodología como calificación por atributo.

#### Kanban

- Mejoramiento de la calidad - \*
- Tiempo necesario para el desarrollo - #
- Aplicabilidad - E
- Capacidad de adaptación a cambios de requerimientos - +

#### Test Driven Development

- Mejoramiento de la calidad - \*
- Tiempo necesario para el desarrollo - #
- Aplicabilidad - E
- Capacidad de adaptación a cambios de requerimientos - +

#### Extreme Programming

- Mejoramiento de la calidad - \*
- Tiempo necesario para el desarrollo - #
- Aplicabilidad - 0
- Capacidad de adaptación a cambios de requerimientos - +

#### Desarrollo en cascada

- Mejoramiento de la calidad - +
- Tiempo necesario para el desarrollo - #
- Aplicabilidad - E
- Capacidad de adaptación a cambios de requerimientos - |

Al realizar este paso puede comprobarse que Extreme Programming no cumplió el requisito de aplicabilidad, por lo que no podría ser usada. Además, las metodologías restantes obtuvieron

resultados idénticos para el campo de tiempo necesario para el desarrollo, por lo que dicho atributo debería ser eliminado. Esto da lugar a los siguientes resultados:

#### Kanban

- Mejoramiento de la calidad - \*
- Aplicabilidad - E
- Capacidad de adaptación a cambios de requerimientos - +

#### Test Driven Development

- Mejoramiento de la calidad - \*
- Aplicabilidad - E
- Capacidad de adaptación a cambios de requerimientos - +

#### Desarrollo en cascada

- Mejoramiento de la calidad - +
- Aplicabilidad - E
- Capacidad de adaptación a cambios de requerimientos - |

Al elaborar el ranking, los resultados fueron los siguientes:

- Kanban - 1E, 1\* y 1+
- Test Driven Development - 1E, 1\* y 1
- Desarrollo en cascada - 1E, 1+ y 1|

Por la escasa capacidad de adaptación a cambios de requerimientos del desarrollo en cascada, quedó en evidencia la necesidad de excluirla a la hora de desarrollar el sistema. Kanban y Test Driven development resultaron ser según el criterio aplicado las metodologías más apropiadas, por lo que se eligió usar ambas en el desarrollo del sistema.

#### 6.1.2. Selección de lenguajes frontend

Al momento de seleccionar entre ambos lenguajes se optó por otorgar los siguientes pesos a los atributos a considerar:

- Sencillez de uso - \*
- Grado de adopción - +
- Rendimiento - +
- Mantenibilidad - #
- Dificultad de aprendizaje - |
- Aplicabilidad - E

Al calificar mediante símbolos los atributos en cuestión de JavaScript y TypeScript, los resultados fueron los siguientes:

### JavaScript

- Sencillez de uso - \*
- Grado de adopción - +
- Rendimiento - +
- Mantenibilidad - |
- Dificultad de aprendizaje - |
- Aplicabilidad - E

### TypeScript

- Sencillez de uso - +
- Grado de adopción - +
- Rendimiento - +
- Mantenibilidad - #
- Dificultad de aprendizaje - |
- Aplicabilidad - E

Pueden observarse resultados idénticos entre ambas tecnologías para los campos de grado de adopción, rendimiento, dificultad de aprendizaje y aplicabilidad, por lo que se deben eliminar dichos atributos, dando lugar a los siguientes resultados:

### JavaScript

- Sencillez de uso - \*
- Mantenibilidad - |

### TypeScript

- Sencillez de uso - +
- Mantenibilidad - #

El ranking quedó de la siguiente forma:

- JavaScript - 1\* y 1|
- TypeScript - 1# y 1+

La ventaja de TypeScript en cuanto a mantenibilidad sobre JavaScript permitió compensar su menor sencillez de uso, por lo que entregó un resultado más equilibrado. La importancia otorgada a la sencillez de uso era mayor que la dada a la mantenibilidad, por lo que si bien ambas opciones eran buenas candidatas para el desarrollo en cuestión, se optó por usar JavaScript.

#### 6.1.3. Selección de framework para CSS

Al momento de optar por una de las dos opciones, a los atributos a considerar se les asignaron los siguientes símbolos como pesos:

- Sencillez de uso - #
- Grado de adopción - +
- Rendimiento - +
- Mantenibilidad - \*
- Dificultad de aprendizaje - #
- Aplicabilidad - E

Los frameworks en cuestión recibieron estas calificaciones para cada atributo:

#### Bootstrap

- Sencillez de uso - #
- Grado de adopción - +
- Rendimiento - +
- Mantenibilidad - +
- Dificultad de aprendizaje - #
- Aplicabilidad - E

#### Tailwind

- Sencillez de uso - #
- Grado de adopción - +
- Rendimiento - +
- Mantenibilidad - #
- Dificultad de aprendizaje - +
- Aplicabilidad - E

Los atributos de sencillez de uso, grado de adopción, rendimiento y aplicabilidad recibieron pesos idénticos para ambas tecnologías, por lo que debieron eliminarse, dando lugar a los siguientes resultados:

#### Bootstrap

- Mantenibilidad - +
- Dificultad de aprendizaje - #

#### Tailwind

- Mantenibilidad - #
- Dificultad de aprendizaje - +

Lo que lleva a que se forme el siguiente ranking:

- Tailwind - 1# y 1+
- Bootstrap - 1# y 1+

Aunque los resultados lucen idénticos, Tailwind recibió una calificación superior en el campo de mantenibilidad, mientras que Bootstrap la recibió para el de dificultad de aprendizaje, que era menos valorado. Esto llevó a que se optara por usar tailwind ya que era más importante contar con un código mantenable que con un framework de CSS más sencillo de aprender.

#### 6.1.4. Selección de frameworks o bibliotecas frontend

Para elegir entre las opciones detalladas anteriormente, se otorgaron los siguientes pesos a los atributos a tener en cuenta:

- Sencillez de uso - \*
- Grado de adopción - #
- Rendimiento - +
- Mantenibilidad - #
- Dificultad de aprendizaje - +
- Aplicabilidad - E

Luego se asignó pesos por cada atributo a las tecnologías en cuestión:

Angular

- Sencillez de uso - +
- Grado de adopción - #
- Rendimiento - +
- Mantenibilidad - #
- Dificultad de aprendizaje - +
- Aplicabilidad - 0

React

- Sencillez de uso - \*
- Grado de adopción - #
- Rendimiento - +
- Mantenibilidad - #
- Dificultad de aprendizaje - +
- Aplicabilidad - E

Vue.js

- Sencillez de uso - #
- Grado de adopción - #
- Rendimiento - +
- Mantenibilidad - #

- Dificultad de aprendizaje - +
- Aplicabilidad - E

Al usar TypeScript en lugar de JavaScript, Angular tuvo que ser eliminado de entre las opciones ya que anteriormente fue elegido JavaScript como lenguaje a utilizar durante el desarrollo del proyecto. Además, los atributos grado de adopción, rendimiento, mantenibilidad y dificultad de aprendizaje tuvieron resultados idénticos por lo que también debieron eliminarse, quedando los siguientes resultados.

#### React

- Sencillez de uso - \*
- Aplicabilidad - E

#### Vue.js

- Sencillez de uso - #
- Aplicabilidad - E

Esto llevó a que se obtuviera al próximo ranking:

- React - 1\* y 1E
- Vue.js - 1# y 1E

Aunque se considera más fácil de usar a Vue.js para aplicaciones pequeñas, la experiencia previa con JavaScript y React llevó a que se considere más fácil de usar a React para el desarrollo del sistema.

#### **6.1.5. Lenguajes y frameworks backend**

Para elegir entre los dos grupos de tecnologías, se asignó los siguientes pesos a los atributos que se tomarían en consideración:

- Sencillez de uso - \*
- Grado de adopción - +
- Rendimiento - |
- Mantenibilidad - #
- Dificultad de aprendizaje - +
- Aplicabilidad - E

La calificación dada a cada atributo de los grupos de tecnologías detallados anteriormente fue la siguiente:

#### Java y Spring

- Sencillez de uso - +
- Grado de adopción - +
- Rendimiento - |

- Mantenibilidad - #
- Dificultad de aprendizaje - +
- Aplicabilidad - E

Node.js® y Express.js

- Sencillez de uso - \*
- Grado de adopción - +
- Rendimiento - |
- Mantenibilidad - +
- Dificultad de aprendizaje - +
- Aplicabilidad - E

Los atributos de grado de adopción, rendimiento, dificultad de aprendizaje y aplicabilidad recibieron pesos idénticos, por lo que debieron eliminarse, dando lugar a los próximos resultados:

Java y Spring

- Sencillez de uso - +
- Mantenibilidad - #

Node.js® y Express.js

- Sencillez de uso - \*
- Mantenibilidad - +

Lo que llevó a que se formara este ranking:

- Node.js y Express.js - 1\* y 1+
- Java y Spring - 1# y 1+

Node.js y Express.js tuvieron un resultado mejor, por lo que fueron elegidos para llevar a cabo el desarrollo del lado del servidor del sistema.

#### **6.1.6. Selección de Sistema Gestor de Base de Datos**

Para realizar la selección el software gestor de base de datos, se asignaron los siguientes pesos a los atributos que se tomarían en cuenta:

- Sencillez de uso - \*
- Grado de adopción - +
- Rendimiento - |
- Mantenibilidad - |
- Dificultad de aprendizaje - #
- Aplicabilidad - E

Luego se asignaron pesos por cada atributo a las tecnologías detalladas con anterioridad:

#### MongoDB

- Sencillez de uso - \*
- Grado de adopción - +
- Rendimiento - |
- Mantenibilidad - |
- Dificultad de aprendizaje - #
- Aplicabilidad - E

#### MySQL

- Sencillez de uso - #
- Grado de adopción - +
- Rendimiento - |
- Mantenibilidad - |
- Dificultad de aprendizaje - +
- Aplicabilidad - E

#### PostgreSQL

- Sencillez de uso - #
- Grado de adopción - +
- Rendimiento - |
- Mantenibilidad - |
- Dificultad de aprendizaje - +
- Aplicabilidad - E

Se obtuvieron calificaciones idénticas para los atributos grado de adopción, rendimiento, mantenibilidad y aplicabilidad, por lo que estos debieron eliminarse dando lugar a los próximos resultados:

#### MongoDB

- Sencillez de uso - \*
- Dificultad de aprendizaje - #

#### MySQL

- Sencillez de uso - #
- Dificultad de aprendizaje - +

#### PostgreSQL

- Sencillez de uso - #

- Dificultad de aprendizaje - +

A continuación se formó el siguiente ranking:

- MongoDB - 1\* y 1#
- MySQL - 1# y 1+
- PostgreSQL - 1# y 1+

MongoDB obtuvo la mayor calificación por lo que se eligió como software gestor de base de datos para el proyecto.

#### **6.1.7. Selección de framework para testing**

Para elegir entre los frameworks detallados anteriormente, se asignaron los siguientes pesos a los atributos en cuestión:

- Sencillez de uso - #
- Grado de adopción - #
- Rendimiento - +
- Mantenibilidad - \*
- Dificultad de aprendizaje - +
- Aplicabilidad - E

Luego se asignaron pesos por cada atributo a las tecnologías candidatas:

Jasmine

- Sencillez de uso - #
- Grado de adopción - #
- Rendimiento - +
- Mantenibilidad - \*
- Dificultad de aprendizaje - +
- Aplicabilidad - E

Jest

- Sencillez de uso - #
- Grado de adopción - #
- Rendimiento - #
- Mantenibilidad - \*
- Dificultad de aprendizaje - +
- Aplicabilidad - E

JUnit

- Sencillez de uso - #
- Grado de adopción - #
- Rendimiento - #
- Mantenibilidad - \*
- Dificultad de aprendizaje - +
- Aplicabilidad - 0

#### Mocha

- Sencillez de uso - #
- Grado de adopción - #
- Rendimiento - +
- Mantenibilidad - \*
- Dificultad de aprendizaje - +
- Aplicabilidad - E

Al no ser aplicable por estar dirigido al testing de código escrito en Java, JUnit debió eliminarse de entre las opciones posibles, ya que el lenguaje que se seleccionó para el desarrollo del lado del servidor fue JavaScript al optarse por usar Node.js®. Los campos de sencillez de uso, grado de adopción, mantenibilidad y dificultad de aprendizaje dieron resultados idénticos por lo que también debieron eliminarse, dando lugar a los próximos resultados:

#### Jasmine

- Rendimiento - +
- Aplicabilidad - E

#### Jest

- Rendimiento - #
- Aplicabilidad - E

#### Mocha

- Rendimiento - +
- Aplicabilidad - E

A continuación se puede ver el ranking para estas tecnologías:

- Jest - 1E y 1#
- Jasmine - 1E y 1+
- Mocha - 1E y 1+

Por su rendimiento fue elegido Jest como framework para testing, que obtuvo la calificación más alta de entre las opciones consideradas.

## 6.2. Manual de uso

El Sistema Informativo para la Recepción del Hospital de Clínicas consta de 4 secciones principales. Aquí se verá la funcionalidad que tiene cada sección y cómo utilizarlas. Para pasar de una sección a otra, presionar el botón en la esquina superior izquierda y seleccionar la pantalla que se desee abrir, luego se puede cerrar el menú presionando de nuevo el mismo botón usado para abrirlo.

En primer lugar se encuentra la pantalla principal (fig. 17), en la cual pueden observarse los puestos libres y las trivias.

**Figura 17**

*Interfaz de la pantalla principal del sistema*



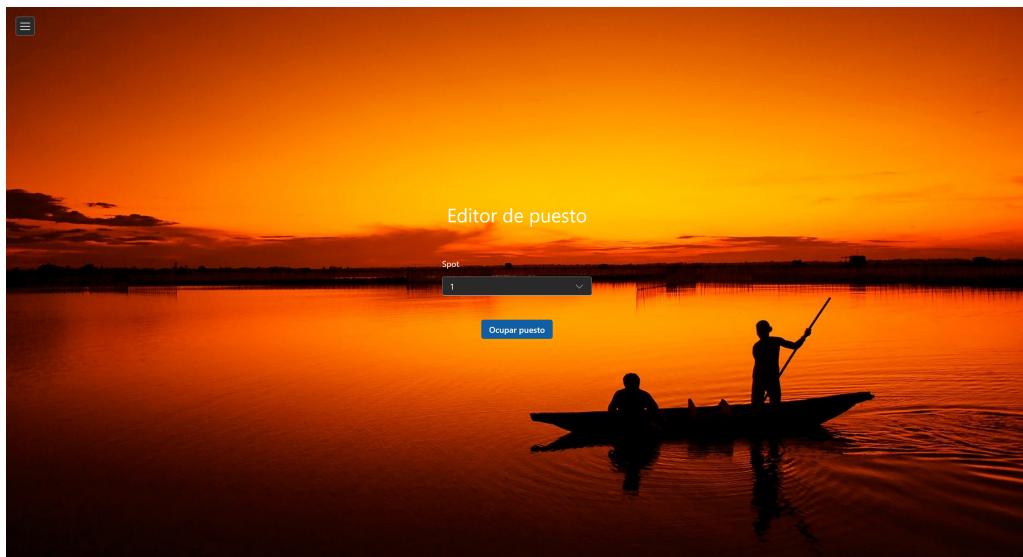
Al liberarse u ocuparse puestos, la lista de puestos libres se actualiza sin necesidad de recargar manualmente la página. En caso de querer tener una salida sonora, al abrir esta página habilitar el audio haciendo click en el botón con forma de altavoz en la parte inferior de la sección de puestos libres. El tipo de voz usada para esta salida sonora puede variar de un navegador de internet a otro.

La sección de trivias va mostrando las distintas trivias que hay guardadas en el sistema, alternando entre mostrar solo la pregunta y opciones de respuestas, y mostrar esto junto con la explicación, indicando las respuestas correctas.

La pantalla del editor de puesto (fig. 18) permite que alguien que está atendiendo en un puesto indique cuando su puesto se ocupe o se libere. Conforme se vayan ocupando y liberando puestos, la pantalla principal actualizará la lista de puestos libres y también informará sobre la nueva lista audiblemente en caso de estar habilitada la salida sonora. Tiene un selector de puesto y un botón para alternar su estado entre ocupado y liberado.

**Figura 18**

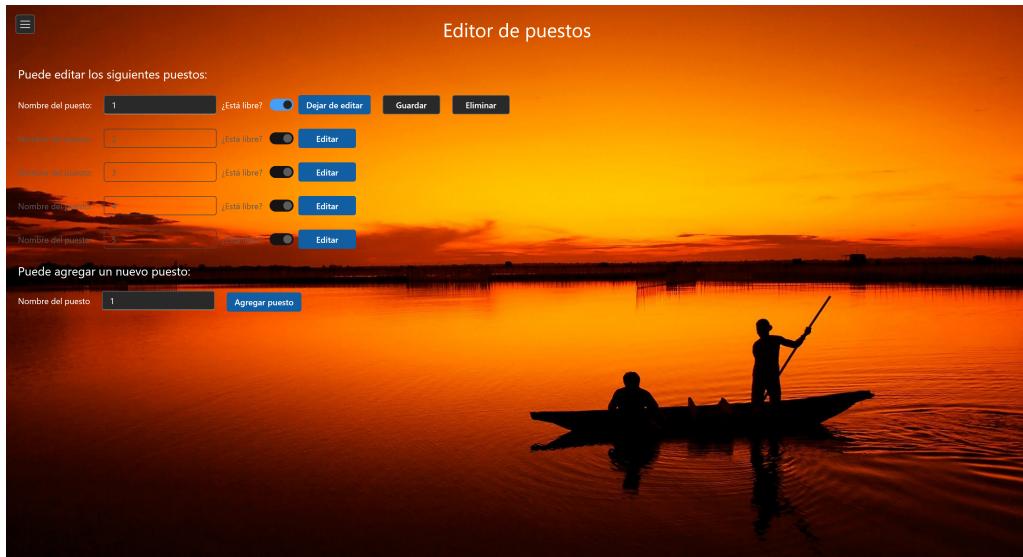
*Interfaz del editor de puesto*



La pantalla del editor de puestos avanzado (fig. 19) permite modificar el nombre y la disponibilidad de cada puesto, agregar más puestos y quitar los ya existentes. Antes de editar un puesto es necesario presionar en el botón Editar de dicho puesto.

**Figura 19**

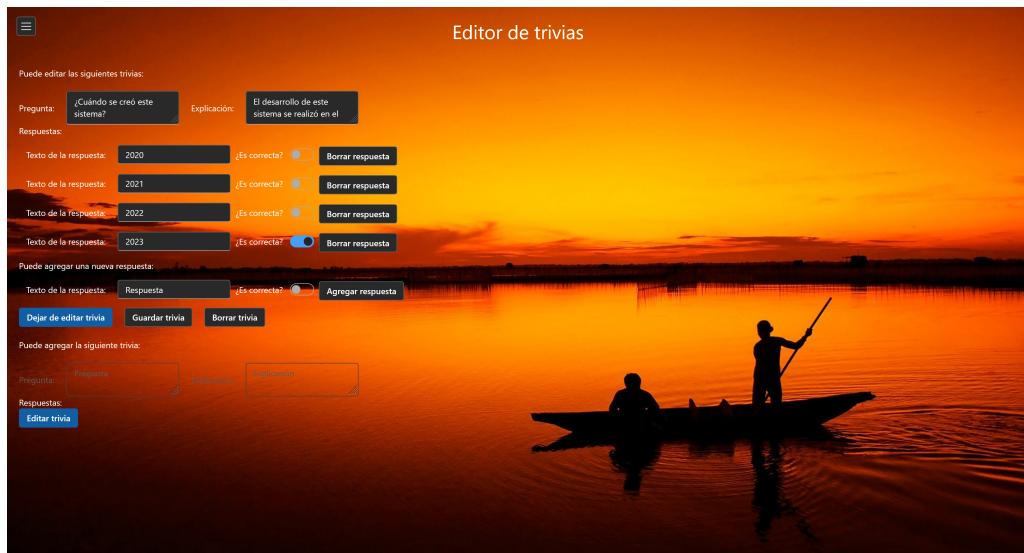
*Interfaz del editor de puestos avanzado*



La pantalla del editor de trivias (fig. 20) permite agregar y remover trivias, y también editar cada trivia. Al presionar el botón Editar, se habilitan las opciones para modificar la pregunta, explicación y respuestas (tanto su texto y si son correctas como su cantidad, pudiendo agregar o remover opciones).

**Figura 20**

*Interfaz del editor de trivias*



## 6.3. Manual de instalación

### 6.3.1. Requisitos

A continuación se mostrará cómo puede configurarse el sistema para que este funcione y cuáles son sus requisitos.

Este sistema está pensado para correr en una computadora principal que actuará como servidor y en otras (una por puesto) que se encuentren conectadas en la misma red local de internet.

El servidor deberá correr, preferentemente, un sistema basado en Linux, en este caso se darán las instrucciones para Ubuntu 22.04.

La base de datos requiere que el procesador tenga soporte para las instrucciones AVX, por lo que procesadores muy antiguos o de gamas más bajas pueden dar problemas. Esto puede resolverse usando un procesador más moderno o una versión más antigua de MongoDB (4 o inferior).

Para realizar el despliegue del sistema se hará uso de tecnologías basadas en contenedores: Docker y Kubernetes.

Será necesario contar con Docker instalado y configurado para poder ser ejecutado sin necesitar permisos de superusuario. También será necesario tener instalado minikube y kubectl para poder orquestar los contenedores con kubernetes.

A continuación se dejan algunos enlaces útiles para instalar y configurar el software necesario para ejecutar el sistema.

- <https://docs.docker.com/engine/install/ubuntu/>
- <https://docs.docker.com/engine/install/linux-postinstall/>
- <https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>
- <https://minikube.sigs.k8s.io/docs/start/>

### 6.3.2. Iniciar minikube

Para poder correr las capas del sistema, ejecutar en primer lugar el siguiente comando:

```
minikube start
```

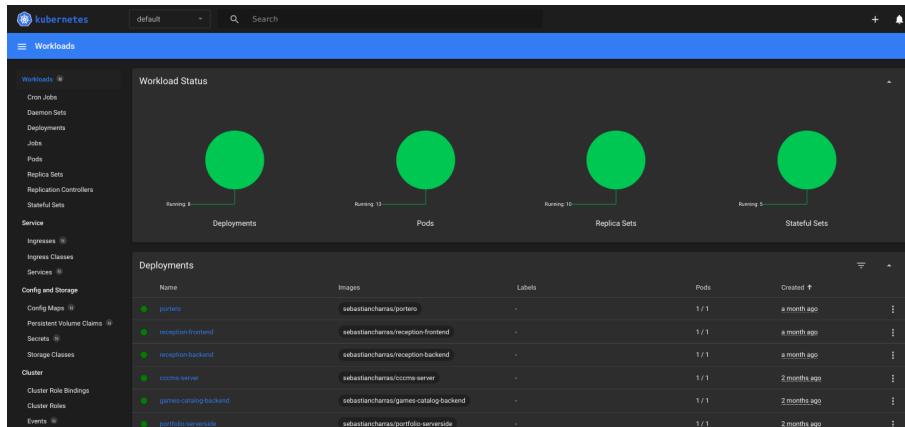
Para poder ver el estado de las capas del sistema ejecutar el próximo comando:

```
minikube dashboard
```

Este comando abrirá en un navegador de internet una interfaz como la que puede observarse en la figura 21.

**Figura 21**

*Interfaz del dashboard de Kubernetes*



### 6.3.3. Aplicar configuraciones de kubernetes

Es necesario aplicar las configuraciones del backend, frontend y la base de datos, para ello hay que primero clonar el repositorio del sistema:

```
git clone https://github.com/charrassebastian/sirhc
```

Los archivos de configuraciones tendrán un contenido similar al que puede observarse en la figura 22.

**Figura 22**

*Contenido de un archivo de configuración de kubernetes*

A screenshot of a terminal window titled 'database : vim — Konsole'. The window shows a code editor with a file containing a YAML configuration for a PersistentVolumeClaim. The code is as follows:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sirhc-pvc
  labels:
    app: sirhc-database
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: gp2
```

Ahora ir al directorio que contiene el código (sirhc), abrir el subdirectorio database. Luego ejecutar los siguientes comandos para aplicar cada archivo de configuración:

```
kubectl apply -f persistentvolume.yaml
```

```
kubectl apply -f persistentvolumeclaim.yaml
```

```
kubectl apply -f service.yaml  
kubectl apply -f statefulset.yaml  
kubectl apply -f storageclass.yaml
```

Con estas configuraciones para la base de datos aplicadas, es hora de configurar el backend. Acceder dentro del directorio sirhc al subdirectorio backend, y luego ejecutar:

```
kubectl apply -f deployment.yaml  
kubectl apply -f service.yaml
```

Ahora que las configuraciones para el frontend ya se han aplicado, hay que configurar el frontend. Abrir dentro del directorio sirhc el subdirectorio frontend y ejecutar:

```
kubectl apply -f sirhc-frontend-deployment.yaml  
kubectl apply -f sirhc-frontend-service.yaml
```

#### 6.3.4. Hacer accesible el sistema

Luego de unos minutos los contenedores ya deberían estar corriendo una vez que hayan sido descargados los datos necesarios. Cuando en el dashboard de minikube aparezcan todos los deployments con el estado running, entonces es hora de permitir las conexiones a los contenedores:

```
kubectl port-forward svc/sirhc-backend 8082:80 &  
kubectl port-forward svc/sirhc-frontend 8080:80 &
```

Para saber cuál es la dirección IP del servidor, desde este ejecutar:

```
ip a
```

Esta dirección IP deberá usarse desde los navegadores de internet que vayan a acceder al sistema. Es importante que las computadoras que se conecten se encuentren en la misma red local, ya que de lo contrario no podrán hacer uso del sistema.

Ahora se puede ingresar al navegador de internet la url con el formato:

```
http://localhost:8080
```

Reemplazando localhost por la dirección IP del servidor, se podrá ver la pantalla principal del sistema.

De aquí en adelante, cada vez que se inicie nuevamente el servidor habrá que ejecutar estos comandos para que funcione el sistema:

```
minikube start  
kubectl port-forward svc/sirhc-backend 8082:80 &  
kubectl port-forward svc/sirhc-frontend 8080:80 &
```

Y luego obtener la dirección IP para conectarse al servidor:

```
ip a
```