Taller de Arquitecturas de Servidores de Aplicaciones, Meta protocolos de objetos, Patrón IoC, Reflexión

Jairo Eduardo Pulido

Profesor:
Luis Daniel Benavides Navarro

Arquitecturas Empresariales

Escuela Colombiana de Ingeniería Julio Garavito
11 de Febrero del 2021 Bogotá D.C.

TABLA DE CONTENIDO

1. INTRODUCCIÓN 2
2. OBJETIVOS 2
3. MARCO TEÓRICO 2
4. DISEÑO Y CONSTRUCCIÓN 4
4.1. PRUEBAS 5
5. ARQUITECTURA 9
6. CONCLUSIONES 10
7. REFERENCIAS 11

1. INTRODUCCIÓN

En el Taller de Arquitecturas de Servidores de Aplicaciones, Meta protocolos de objetos, Patrón IoC, Reflexión se explorarán los conceptos de esquemas de nombres y de clientes y servicios. Adicionalmente, el taller también explorará la arquitectura de las aplicaciones distribuidas sobre internet, para así ver el funcionamiento detallado de una aplicación web capaz de recibir múltiples solicitudes no concurrentes, creando un framework IoC para la construcción de aplicaciones web a partir de POJOS permitiendo publicar servicios web get y post para poder acceder a recursos estáticos como páginas web, javascripts, imágenes CSSs, entre otras cosas, desplegado usando un servidor web llamado Heroku para poder acceder a ella de manera totalmente remota. Para verificar el funcionamiento de cada uno de los requisitos, se realizó una simulación de una página llamada Fancy Wallpapers, en la cual el usuario tiene una interfaz de usuario en la que tiene varios recursos, tales como poder visualizar varios fondos de pantalla, ver la interfaz final de usuario con un fondo de pantalla y mensaje de bienvenida, y ver un agradecimiento usando js con un mensaje de agradecimiento, y asimismo poder visualizar un fondo de pantalla en esa misma página.

2. OBJETIVOS

- Elaborar un programa creando un framework que permita publicar los servicios web get y post.
- Implementar un servidor web que soporte múltiples solicitudes seguidas (no concurrentes). El servidor debe retornar todos los archivos solicitados, incluyendo páginas html e imágenes.
- Usando un servidor y Java, crear un framework IoC para la construcción de aplicaciones web a partir de POJOS permitiendo publicar servicios web get y post para poder acceder a recursos estáticos como páginas web, javascripts, imágenes CSSs.

3. MARCO TEÓRICO

- **JavaScript:** Lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., puedes apostar que probablemente JavaScript está involucrado. Es la tercera capa del pastel

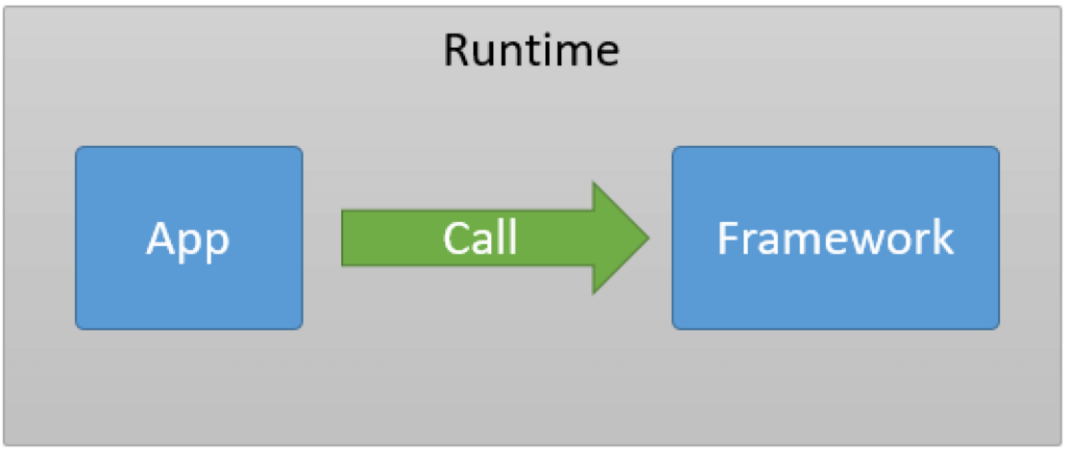
de las tecnologías web estándar, dos de las cuales (HTML y CSS) hemos cubierto con mucho más detalle en otras partes del Área de aprendizaje.

- **CSS:** Lenguaje que define la apariencia de un documento escrito en un lenguaje de marcado (por ejemplo, HTML).

  Así, a los elementos de la página web creados con HTML se les dará la apariencia que se desee utilizando CSS: colores, espacios entre elementos, tipos de letra, ... separando de esta forma la estructura de la presentación.

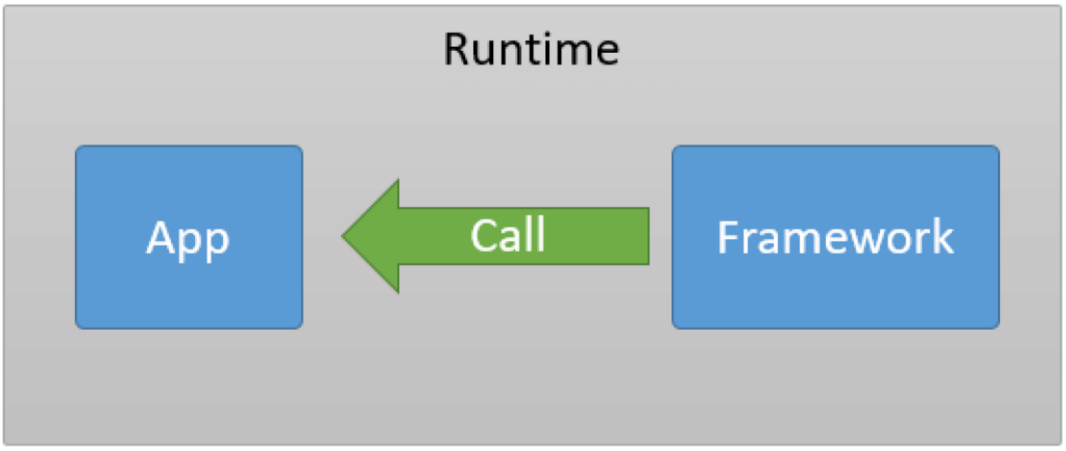
  Esta separación entre la estructura y la presentación es muy importante, ya que permite que sólo cambiando los CSS se modifique completamente el aspecto de una página web. Esto posibilita, entre otras cosas, que los usuarios puedan usar hojas de estilo personalizadas (como hojas de estilo de alto contraste o de accesibilidad).
- **HTML:** Lenguaje de marcado de hipertexto, y le permite al usuario crear y estructurar secciones, párrafos, encabezados, enlaces y elementos de cita en bloque (blockquotes) para páginas web y aplicaciones.

  HTML no es un lenguaje de programación, lo que significa que no tiene la capacidad de crear una funcionalidad dinámica. En cambio, hace posible organizar y formatear documentos, de manera similar a Microsoft Word.
- **Framework:** Conjuntos de componentes de software que se utilizan para crear la estructura de un software o una aplicación. Un framework puede verse como una caja de herramientas en la que el desarrollador busca los componentes necesarios. Este marco proporciona una estructura general para facilitar la labor de desarrollo. Los frameworks funcionan mediante el lenguaje de programación y desarrollan todo tipo de soporte: sitios web, juegos, aplicaciones móviles, etc. Sin embargo, también puede crear su propio marco de software.
- **IoC:** es una forma de trabajar que rompe con el formato tradicional, en donde el programar es el encargado de definir la secuencia de operaciones que se deben de realizar para llegar a un resultado. En este sentido el programador debe conocer todos los detalles del framework y las operaciones a realizar en él, por otra parte, el Framework no conoce absolutamente nada de nuestro programa. Es decir, solo expone operaciones para ser ejecutada. La siguiente imagen muestra la forma de trabajo normal.



Veamos que nuestra aplicación es la que llama en todo momento al Framework y este hace lo que la App le solicita. Por otra parte, al Framework no le interesa en absoluto la App.

Pero qué pasaría si invirtiéramos la forma de trabajo, y que en lugar de que la App llame funciones del Framework, sea el Framework el que ejecute operaciones sobre la App, ¿suena extraño no? Pues en realidad a esta técnica es a la que se le conoce como Inversion of Control. Veamos en la siguiente imagen como es que funciona IoC:



Esta última imagen se ve bastante parecida a la anterior, sin embargo, tiene una gran diferencia, y es que las llamadas se invirtieron, ahora es el Framework es el realiza operaciones sobre nuestra App.

- **POJO:** Son las iniciales de "Plain Old Java Object", que puede interpretarse como "Un objeto Java Plano Antiguo". Un POJO es una instancia de una clase que no extiende ni implementa nada en especial. Para los programadores Java sirve para enfatizar el uso de clases simples y que no dependen de un framework en especial. Este concepto surge en oposición al modelo planteado por los estándares EJB anteriores al 3.0, en los que los Enterprise JavaBeans (EJB) debían implementar interfaces especiales.

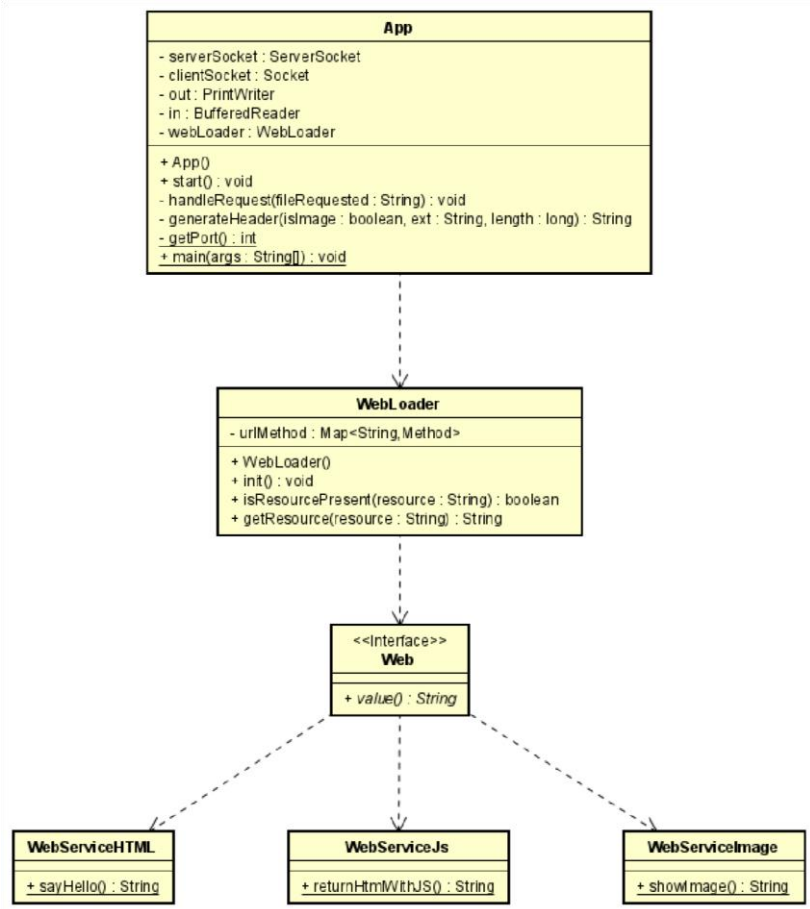
  Por ejemplo, un Controller de Spring tiene que extender de SimpleFormController, e implementar los métodos abstractos de ese tipo: por eso, no es un POJO. Un Servlet, tiene que extender de HttpServlet por lo que tampoco es un POJO. En cambio, si defines una clase Cliente con atributos y unas cuantas operaciones, tienes un simple y modesto POJO.

4. DISEÑO Y CONSTRUCCIÓN

Para la creación del proyecto, se dispuso de seis clases en total. La clase **App** que se encarga de llevar a cabo la ejecución del programa, en la cual se encarga directamente de ejecutar

el Socket tanto del cliente como el del servidor, para poder inicializar el programa, y generar un encabezado en la página web. Asimismo, también se encarga de ejecutar directamente la clase **WebLoader**, la cual se encarga de cargar todos los recursos presentes de la página web, que en total son tres. Para esto, utiliza las anotaciones de **Web**, que es una interfaz que mediante anotaciones se encarga de ejecutar la clase **WebServiceHTML**, que contiene la página en formato HTML del interfaz principal de usuario, la cual contiene un fondo de pantalla y un mensaje de bienvenida. También ejecuta la clase **WebServiceJs**, la cual se encarga de retornar un mensaje de despedida en **js**, acompañado de un fondo de pantalla, todo esto también en formato HTML. Por último, ejecuta la clase **WebServiceImage**, que se encarga de mostrar una imagen en formato **.jpg** que se encuentra dentro de la carpeta **resources** dentro del código fuente, la cual contiene la imagen que se muestra en formato HTML.

Diagrama de Clases:

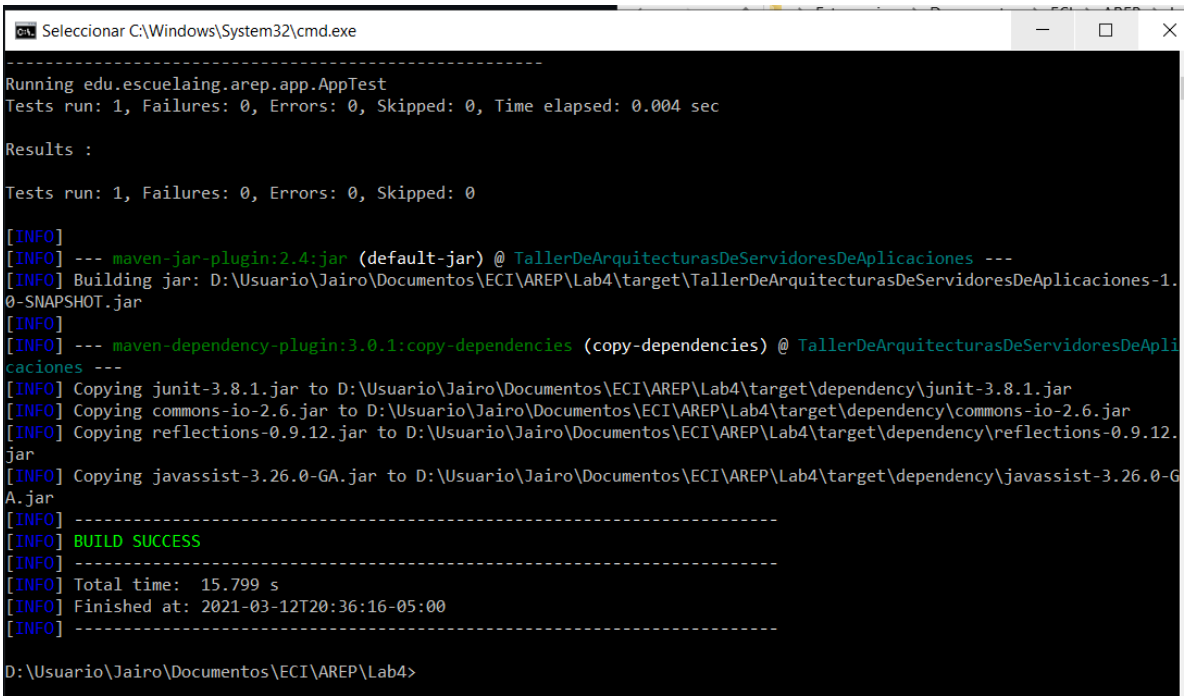


4.1. PRUEBAS

Pruebas en Maven

Para verificar que todo el código está funcionando con total normalidad, realizando los cálculos apropiadamente y obteniéndolos de una lista enlazada, se crearon varias pruebas

unitarias utilizando **Junit**, herramienta que nos provee el IDE que se utilizó para la realización de todo el código fuente, Eclipse. Para estas pruebas unitarias, también se utilizó Maven, en el cual ejecutando los comandos como **mvn test** también se probó que el código funcionara con total normalidad, demostrando así total funcionamiento del código fuente retornando los resultados esperados.

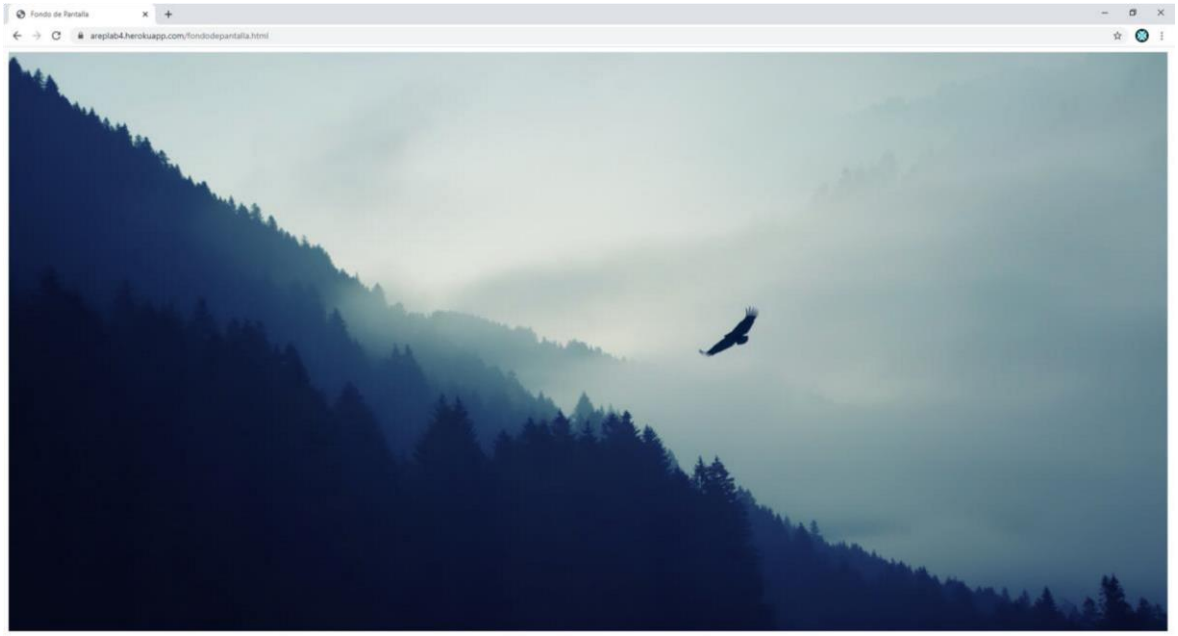


Pruebas en Heroku

Luego de realizar la respectiva configuración del código, con las dependencias añadidas en el archivo **pom.xml** y el **Procfile**, al acceder a la URL de la aplicación desplegada en Heroku (https://areplab4.herokuapp.com/home.html) vemos que la aplicación muestra la interfaz de usuario de la siguiente forma, demostrando que la aplicación dispone de una interfaz desplegada en Heroku, mostrando un mensaje de bienvenida al usuario que ingresa a la página web, donde el usuario puede encontrar una interfaz con un fondo de pantalla acorde con el mensaje de bienvenida.

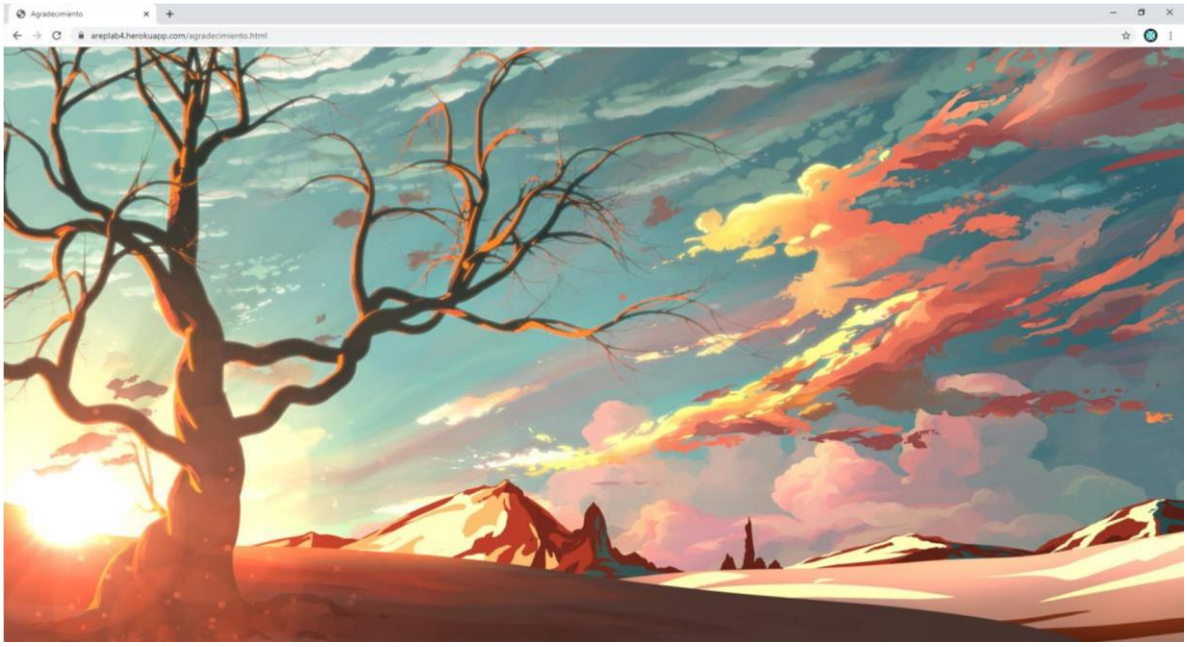
Para acceder a los diferentes recursos, como consultar la imagen implementada y desplegada en la página web, falta agregarle a la URL de Heroku, el recurso

/fondodepantalla.html, el cual despliega un fondo de pantalla que ha sido guardado en la carpeta **resources** del código fuente del programa.



Para acceder al siguiente recurso, el cual es comprobar que la ventana emergente escrito en **js** se despliegue al momento de ingresar a la página, acompañado de un fondo de pantalla acorde al mensaje desplegado, que es un agradecimiento al usuario por haber ingresado a la página web de Fancy Wallpapers. Para acceder a este recurso, se ingresa **/agradecimiento.html** al final de la URL de Heroku. A continuación, se muestra la página web que contiene dicha información.

Luego de presionar clic sobre el botón **OK** o **Aceptar**, se despliega el fondo de pantalla como agradecimiento al usuario por haber ingresado a la página web.



5. ARQUITECTURA

- Al realizar el respectivo despliegue en Heroku de la aplicación, y añadiendo la integración continua con la herramienta CircleCI, vemos que la arquitectura del programa es cliente-servidor, principalmente por lo que la aplicación se encuentra en un servidor de Heroku, la cual puede ser accedida desde cualquier navegador, enviando peticiones HTTP al servidor, el cual las recibe y las procesa para posteriormente por medio de un controlador ofrecido por un framework IoC para la construcción de la aplicacion web a partir de POJOS permitiendo publicar servicios web get y post para poder acceder a recursos estáticos como páginas web, javascripts, imágenes CSSs. La estructura de la arquitectura luego de realizar el despliegue y la integración continua puede ser descrita de la siguiente forma:
- La aplicación primero utiliza una estructura de datos que funciona como memoria, que es la lista enlazada, que se encarga de guardar los datos ingresados por el cliente, mediante el método **add** implementado en el código fuente, junto con el **read** que se encarga de leer los datos y realizar los cálculos.
- Luego el servidor maneja los request, también implementados en el código para manejar abstractamente la comunicación entre el cliente y el servidor, mediante el protocolo HTTP para el envío y el recibimiento de información.
- Después como el código está implementado en Java, procesa los datos utilizando este lenguaje, para así poder retornar el resultado correcto pedido por el cliente, y así mediante anotaciones implementadas dentro del código fuente, y mediante el mismo protocolo HTTP devolverle el resultado al cliente por medio de una interfaz creada en HTML (otro lenguaje), que son las imágenes, mensajes escritos en **js** y una interfaz de usuario conteniendo un fondo de pantalla también.
- Finalmente, la integración continua se maneja junto con Heroku, ya que dentro de la configuración de Heroku se optó por desplegar la aplicación si y solo si el código pasa las pruebas en CircleCI (herramienta utilizada para realizar la integración continua) para así llevar un mejor control de calidad del código, para que se compile apropiadamente para que el cliente o usuario no presente inconvenientes desplegando la aplicación.

6. CONCLUSIONES

- En el laboratorio se utilizó un framework IoC para la construcción de la aplicacion web a partir de POJOS permitiendo publicar servicios web get y post para poder acceder a recursos estáticos como páginas web, javascripts, imágenes CSSs, el cual permitió publicar servicios web "get" y permitió acceder a recursos estáticos como la página de Fancy Wallpapers, en la cual se podían acceder a imágenes como fondos de pantalla, interfaz de usuario y un mensaje de despedida implementado en **js** utilizando anotaciones.
- Asimismo, el despliegue en Heroku permitió poder ejecutar la aplicación de Fancy Wallpapers, probando el funcionamiento de la aplicación web en nube y compilando el código en cualquier ordenador, utilizando el protocolo HTTP para la comunicación cliente-servidor para asegurar una total ejecución del código de manera remota.
- Por otro lado, se implementó integración continua en todo el código fuente para llevar un control de calidad del código, para asegurar que el código esté funcionando totalmente sin ningún problema tanto de compilación como de los resultados retornados después de realizar las respectivas operaciones, y así poder desplegar la aplicación sin presentar ningún tipo de errores.

7. REFERENCIAS

- Vargas, Julián. “Aprende Sobre Desarrollo Web.” Aprende Sobre Desarrollo Web | MDN, 23 Junio 2015, developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript.

- Delgadillo, Daniel. "CSS, ¿Qué Es?" *Arume*, 1 Abr. 2019, www.arumeinformatica.es/blog/css/.
- Bernal, Gustavo. "¿Qué Es HTML? Explicación De Los Fundamentos." Tutoriales Hostinger, 10 Dic. 2020, www.hostinger.es/tutoriales/que-es-html/.
- Sánchez, Sebastián. "¿Qué Es Un Framework? - Wild Code School." Www.wildcodeschool.com, 13 Junio 2019, www.wildcodeschool.com/esES/blog/que-es-un-framework.
- Blancarte, Oscar. "El Concepto Inversion of Control." *Software Architecture*, 1 Dec. 2016, www.oscarblancarteblog.com/2016/12/01/concepto-inversion-ofcontrol/.
- Pesquera, Carlos. "¿Qué Es Un POJO, EJB y Un Bean?" *Transformación Digital*, 19 Mar. 2014, carlospesquera.com/que-es-un-pojo-ejb-y-un-bean/.