TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA WEB PARA LA VISUALIZACIÓN DE DATOS PROVENIENTES DE RED IOT APLICADO AL SECTOR PÚBLICO

**AUTOR: FRANCISCO NAVARRO MOLINA**

**TUTOR: JOSÉ RAFAEL LAJARA VIZCAÍNO**

**ENERO 2023**

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA WEB PARA LA VISUALIZACIÓN DE DATOS PROVENIENTES DE RED IOT APLICADO AL SECTOR PÚBLICO

**AUTOR: FRANCISCO NAVARRO MOLINA**

**TUTOR: JOSÉ RAFAEL LAJARA VIZCAÍNO**

**TRIBUNAL**

**PRESIDENTE/A:**

**VOCAL 1:**

**VOCAL 2:**

**FECHA DE DEFENSA:**

**CALIFICACIÓN:**

**Declaración de autoría:**

Yo, Francisco Navarro Molina, declaro la autoría del Trabajo Fin de Grado titulado “Diseño e implementación de una web para la visualización de datos provenientes de red IoT aplicado al sector turístico” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual. El material no original que figura en este trabajo ha sido atribuido a sus legítimos autores.

Valencia, 20 de diciembre de 2022

Fdo: Francisco Navarro Molina

**Resumen:**

Este trabajo final de grado se enmarca en un proyecto destinado al sector turístico. En este proyecto se ha creado una red de sensores de IoT en diferentes playas para la recogida de datos como temperatura, humedad, índice ultravioleta, etc. que es enviada a una base de datos, mostrada a usuarios y sirve de ayuda a las administraciones para mejorar las futuras estrategias turísticas.

En este trabajo se diseñará una parte de este proyecto, en concreto la aplicación web que muestra información a los usuarios proveniente de la base de datos y recopila información útil para las administraciones. La página web por una parte se conectará a la base de datos de forma asíncrona usando un REST API y formato de datos JSON, así como la tecnología AJAX. La web ha de ser *responsive* porque eso nos permite que sea universal, ya que las aplicaciones móviles sólo valen para una determinada plataforma. Además, durante la realización del trabajo también se utilizarán herramientas de trabajo y metodologías basadas en *git-flow*.

**Abstract:**

This final degree project is part of a project aimed at the tourist sector. In this project, a network of IoT sensors has been created on different beaches to collect data such as temperature, humidity, ultraviolet index, etc., which is sent to a database, displayed to users, and helps administrations to improve future tourist strategies.

In this work, a part of this project will be designed, specifically the web application that shows information to users from the database and collects useful information for administrations. The website will connect to the database asynchronously using a REST API and JSON data format, as well as AJAX technology. The website must be responsive because it allows us to make it universal, as mobile applications only work on a specific platform. In addition, during the work, tools and methodologies based on git-flow will also be used.

**Resum:**

Aquest treball final de grau s'emmarca dins d'un projecte destinat al sector turístic. En aquest projecte s'ha creat una xarxa de sensors d'IoT a diferents platges per a la recollida de dades com ara temperatura, humitat, índex ultravioleta, etc., que s'envia a una base de dades, es mostra als usuaris i serveix d'ajuda a les administracions per a millorar les futures estratègies turístiques.

En aquest treball es dissenyarà una part d'aquest projecte, concretament l'aplicació web que mostra informació als usuaris provinent de la base de dades i recopila informació útil per a les administracions. La pàgina web, per una banda, es connectarà a la base de dades de forma asíncrona utilitzant un REST API i format de dades JSON, així com la tecnologia AJAX. La web ha de ser responsive perquè això ens permet que sigui universal, ja que les aplicacions mòbils només valen per a una determinada plataforma. A més, durant la realització del treball també s'utilitzaran eines de treball i metodologies basades en git-flow.

ÍNDICE GENERAL

[1. Introducción 5](#_Toc123819676)

[1.1. Introducción 5](#_Toc123819677)

[1.2. Motivación. Proyecto Smart Beaches 5](#_Toc123819678)

[1.3. Objetivos 7](#_Toc123819679)

[1.3.1. Funcionales 7](#_Toc123819680)

[1.3.2. No funcionales 7](#_Toc123819681)

[1.4. Organización de la memoria 8](#_Toc123819682)

[2. Estado del arte 9](#_Toc123819683)

[2.1. Lenguajes de programación 9](#_Toc123819684)

[2.2. Servidores de web 9](#_Toc123819685)

[2.3. Servidores de bases de datos 10](#_Toc123819686)

[2.4. Tendencias y herramientas 11](#_Toc123819687)

[3. Especificaciones y planificación 13](#_Toc123819688)

[3.1. Especificaciones 13](#_Toc123819689)

[3.2. Planificación 13](#_Toc123819690)

[4. Metodología 15](#_Toc123819691)

[4.1. Herramientas 15](#_Toc123819692)

[4.2. Gitflow 18](#_Toc123819693)

[4.3. CI/CD 20](#_Toc123819694)

[5. Diseño 23](#_Toc123819695)

[5.1. Diseño gráfico 23](#_Toc123819696)

[5.2. Diseño de la interfaz de usuario (UI) 24](#_Toc123819697)

[5.3. Diseño de la experiencia de usuario (UX) 25](#_Toc123819698)

[5.4. Diseño adaptativo 25](#_Toc123819699)

[5.5. Diseño de la navegación 25](#_Toc123819700)

[6. Implementación y pruebas 29](#_Toc123819701)

[6.1. Implementación 29](#_Toc123819702)

[6.2. Pruebas funcionales 36](#_Toc123819703)

[7. Conclusiones 39](#_Toc123819704)

[8. Bibliografía 41](#_Toc123819705)

[9. Anexo I. Código fuente 43](#_Toc123819706)

[Ilustración 1. Sistema completo 7](#_Toc123819707)

[Ilustración 2. GitHub 15](#_Toc123819708)

[Ilustración 3. Área de trabajo VSCode 16](#_Toc123819709)

[Ilustración 4. Panel de control de XAMPP 18](#_Toc123819710)

[Ilustración 5. Terminal Git Bash 19](#_Toc123819711)

[Ilustración 6. Pull request 19](#_Toc123819712)

[Ilustración 7. Test unitarios. Action GitHub 21](#_Toc123819713)

[Ilustración 8. Rótulo de Smart Beach CV 23](#_Toc123819714)

[Ilustración 9. Selección de las comunidades 24](#_Toc123819715)

[Ilustración 10. Interfaz de usuario 24](#_Toc123819716)

[Ilustración 11. Header 26](#_Toc123819717)

[Ilustración 12. Sección Playas, página principal 26](#_Toc123819718)

[Ilustración 13. Sección TOP 4, página principal 26](#_Toc123819719)

[Ilustración 14. Formulario de comentarios 27](#_Toc123819720)

[Ilustración 15. Datos actuales playa 27](#_Toc123819721)

[Ilustración 16. Gráfica con histórico de datos 27](#_Toc123819722)

[Ilustración 17. Sección de reseñas 28](#_Toc123819723)

[Ilustración 18. Distribución código 29](#_Toc123819724)

[Ilustración 19. Petición fetch() 30](#_Toc123819725)

[Ilustración 20. URL para realizar la petición fetch() 30](https://universitatdevalencia-my.sharepoint.com/personal/namofran_alumni_uv_es/Documents/Escritorio/TFG-Francisco%20Navarro%20Moina/TFG/docs/Memoria%20proyecto/TFG%20Francisco%20Navarro%20Molina.docx#_Toc123819726)

[Ilustración 21. Estructura de clases (parser.js) 31](#_Toc123819727)

[Ilustración 22. Formato JSON 31](#_Toc123819728)

[Ilustración 23. Campos field4 32](#_Toc123819729)

[Ilustración 24. Parseado datos 32](#_Toc123819730)

[Ilustración 25. Beach database 33](#_Toc123819731)

[Ilustración 26. Configuración database 33](#_Toc123819732)

[Ilustración 27. Métodos database 34](#_Toc123819733)

[Ilustración 28. Elemento select de las playas 34](#_Toc123819734)

[Ilustración 29. Conexión base de datos, formulario de contacto 35](#_Toc123819735)

[Ilustración 30. Sanitización de entradas en formulario 35](#_Toc123819736)

[Ilustración 31. Test unitario de parseo 36](#_Toc123819737)

[Ilustración 32. Resultado test unitario parseo 36](#_Toc123819738)

[Ilustración 33. Proceso arreglo formato 37](#_Toc123819739)

[Ilustración 34. Ejecución ESLin 37](#_Toc123819740)

# Introducción

## Introducción

El trabajo fin de grado que se presenta a continuación tiene como objetivo el diseño e implementación de una web para la visualización de datos provenientes de una red de Internet de las cosas (IoT, por sus siglas en inglés) aplicada al sector público (Pastor, 2019).

La red IoT es una red de dispositivos conectados que recopilan y comparten datos a través de Internet. Estos dispositivos pueden ser de diversa índole, desde sensores de temperatura hasta cámaras de vigilancia. En el sector público, la implementación de redes IoT puede tener diversas aplicaciones, desde el seguimiento del consumo de energía en edificios públicos, hasta la monitorización del tráfico en ciudades.

Los datos adquiridos por los nodos de una red IoT normalmente se envían a un nodo central que puede estar conectado a una base de datos. Esos datos se pueden tratar, analizar, visualizar, etc. Una forma habitual de visualizar los datos es a través de una web.

Se ha escogido el formato web debido a que es una herramienta universal, a diferencia de una aplicación móvil que es más cerrada, no estándar y fragmentada en diferentes plataformas.

El objetivo de la web que se va a diseñar en este proyecto es proporcionar una herramienta de visualización de datos fácil de usar y accesible para cualquier usuario. Esta herramienta permitirá a los usuarios acceder a los datos recopilados por la red IoT de manera sencilla y visual, lo que les permitirá obtener información valiosa para la toma de decisiones y el análisis de datos.

Para llevar a cabo el diseño e implementación de la web se utilizarán diversas técnicas y tecnologías, tales como lenguajes de programación, *frameworks* y librerías de visualización de datos.

Además, en este proyecto no nos centraremos solamente en la parte técnica, sino también en el desarrollo e implementación de la aplicación. En este proyecto se seguirá la metodología de trabajo basada en G*itflow,* una forma organizada y estructurada de gestionar el flujo de trabajo en un proyecto de software utilizando Git como herramienta de control de versiones, revisiones de código, tests, etc. Todo esto con el objetivo de mejorar la seguridad y calidad del producto final.

En resumen, el trabajo fin de grado que se presenta busca proporcionar una herramienta útil y accesible para la visualización de datos provenientes de redes IoT aplicadas al sector público, y contribuir así a mejorar la toma de decisiones y el análisis de datos en este ámbito. Y se implementará con metodologías profesionales.

## Motivación. Proyecto Smart Beaches

El proyecto Smart Beaches es una iniciativa que tiene como objetivo la implementación de tecnología de la información y la comunicación (TIC) en playas y zonas costeras con el fin de mejorar la experiencia de los visitantes y promover la sostenibilidad ambiental.

Entre las posibles aplicaciones de la tecnología en el proyecto Smart Beaches se pueden mencionar:

* Monitoreo del estado de las playas y zonas costeras: a través de sensores y dispositivos de monitoreo es posible recopilar datos sobre el estado de la arena, la temperatura del agua, la calidad del aire, etc. Esta información puede ser útil para tomar decisiones sobre el mantenimiento y conservación de las playas y zonas costeras, así como para alertar a los visitantes de cualquier riesgo o peligro.
* Gestión del tráfico y el estacionamiento: la implementación de sistemas de gestión del tráfico y el estacionamiento puede ayudar a mejorar la circulación en las zonas costeras y a evitar el aparcamiento ilegal.
* Servicios de información y orientación: a través de aplicaciones móviles o de paneles informativos es posible proporcionar a los visitantes información útil sobre la zona, como la ubicación de servicios públicos, actividades recreativas, puntos de interés turístico, etc.
* Fomento de la sostenibilidad ambiental: el proyecto Smart Beaches también puede incluir iniciativas para fomentar la sostenibilidad ambiental, como la implementación de sistemas de recogida y gestión de residuos, campañas de concientización sobre la conservación del medio ambiente, etc.

En resumen, el proyecto Smart Beaches es una iniciativa que tiene como objetivo mejorar la experiencia de los visitantes en las playas y zonas costeras a través de la implementación de tecnología de la información y la comunicación, y promover al mismo tiempo la sostenibilidad ambiental en estas áreas (Instituto Valenciano de Tecnologías Turísticas, 2018).

Para todo ello, se han desarrollado una serie de nodos remotos con sensores y comunicaciones inalámbricas en colaboración con la conserjería de turismo para colocarse en varias playas de Valencia. Los nodos se alimentan por energía solar, recogen información de temperatura, humedad, índice ultravioleta, etc. y la envían a un nodo central como se muestra en la Ilustración 1. Esta información se presenta a los usuarios en una web/aplicación para que se pueda consultar, que es donde se enmarca este TFG. En la se puede ver un diagrama con todos los elementos de este proyecto.

Diagrama

Descripción generada automáticamente

Ilustración . Sistema completo

Además, también se recopilan datos sobre el uso de la aplicación para que la conserjería de turismo u otras administraciones tenga información sobre las preferencias de los usuarios.

## Objetivos

Los objetivos que se pretenden conseguir con este proyecto son los siguientes:

### Funcionales

* Capacidad de realizar la conexión con los nodos IoT, recibir los datos y procesarlos utilizando la tecnología AJAX.
* Ser capaces de mostrar los últimos datos, así como un histórico de estos.
* Recopilar información de uso de la aplicación por parte de los usuarios.

### No funcionales

* *Responsiveness*: capacidad para visualizar la web en diferentes tipos de dispositivos, especialmente desde dispositivos móviles.
* Utilización de librerías conocidas: con ello se aumentará la eficacia y seguridad de la solución al no reinventar la rueda y apoyarnos en tecnologías probadas como JQuery, Chart.js, Fetch, etc.
* Accesibilidad: es deseable que las herramientas financiadas por la Administración tengan unos mínimos de criterios de accesibilidad para que puedan ser usadas por la mayor parte de la población.
* Seguridad.
* Requisitos legales (política de privacidad, cookies, política de gestión de datos).
* Calidad de código (ver siguiente capítulo).
* Metodología de trabajo Gitflow (ver siguiente capítulo).

## Organización de la memoria

Esta memoria de Trabajo de Fin de Grado se dividirá en diferentes apartados:

* Estado del arte: la recopilación de información sobre el estado actual de la tecnología, técnicas y tendencias en el campo del diseño web.
* Especificaciones y planificación: este incluirá información detallada sobre el alcance del proyecto, los requisitos del sitio web, la arquitectura del sitio y el plan de desarrollo.
* Metodología: describe el enfoque y los pasos seguidos para llevar a cabo el proyecto.
* Diseño: describe el proceso de diseño y los elementos visuales utilizados en el sitio web.
* Implementación: describe cómo se ha llevado a cabo la implementación del sitio web y cómo se ha puesto en marcha.
* Validación y pruebas: describe cómo se han llevado a cabo pruebas y validación del sitio web para asegurar su correcto funcionamiento y usabilidad. De esta manera podemos asegurar que las especificaciones anteriormente listadas se cumplen.
* Resultados: describe los resultados y logros alcanzados con el proyecto.
* Conclusiones: se resume todo lo que se ha hecho durante el proceso de diseño, así como las lecciones aprendidas y la posible continuación del proyecto.

# Estado del arte

El estado del arte de un proyecto de diseño web es la recopilación de información sobre el estado actual de la tecnología, técnicas y tendencias en el campo de diseño web.

## Lenguajes de programación

En el desarrollo de páginas web y aplicaciones web existen una gran variedad de lenguajes de programación que se pueden utilizar. En este apartado se revisan los lenguajes de programación más utilizados en la actualidad y se mencionan sus principales características y ventajas.

Uno de los lenguajes más utilizados en el desarrollo web es HTML (*HyperText Markup Language*). Se trata de un lenguaje de marcado que se utiliza para estructurar y dar formato al contenido de una página web. HTML es un lenguaje sencillo de aprender y es compatible con la mayoría de los navegadores web (W3C, 2009 - 2017).

Otro lenguaje fundamental en el desarrollo web es CSS (*Cascading Style Sheets*). CSS se utiliza para definir el estilo y la apariencia de una página web, permitiendo separar el contenido y la presentación. Gracias a CSS es posible crear diseños atractivos y adaptables a diferentes dispositivos y tamaños de pantalla.

JavaScript es otro lenguaje de programación muy utilizado en el desarrollo web. Se trata de un lenguaje de programación del lado del cliente que se utiliza para añadir interactividad y funcionalidades dinámicas a una página web. JavaScript es un lenguaje muy versátil y se puede utilizar tanto en el *front-end* como en el *back-end* de una aplicación web.

PHP (*Hypertext Preprocessor*) es un lenguaje de programación del lado del servidor que se utiliza para generar contenido dinámico en una página web. Es un lenguaje de propósito general que se puede utilizar en conjunción con bases de datos y es muy popular en el desarrollo de sistemas de gestión de contenidos (CMS) (The PHP Group, 2001 - 2022).

Ruby es un lenguaje de programación orientado a objetos y de propósito general que se utiliza para el desarrollo de aplicaciones web y otras aplicaciones. Es un lenguaje sencillo y expresivo que se ha ganado una gran popularidad en los últimos años.

Por último, Python es otro lenguaje de programación de propósito general que se utiliza en una amplia variedad de aplicaciones, incluyendo el desarrollo de aplicaciones web. Es un lenguaje interpretado y de alto nivel que se caracteriza por su sintaxis clara y legible. Algunas de las ventajas de Python en el desarrollo web son su facilidad de aprendizaje, su gran comunidad de desarrolladores y la amplia variedad de *frameworks* y librerías disponibles.

En este proyecto, de los tres lenguajes de servidor comentados, nos decantaremos por PHP. Este lenguaje se integra muy bien en los servidores web usados y proporciona todas las herramientas que se necesitarán. Además, su uso en este tipo de aplicaciones está más que probado desde hace décadas.

## Servidores de web

En el desarrollo web se utilizan principalmente dos tipos de servidores: servidores de desarrollo y servidores de producción.

Los servidores de desarrollo se utilizan para probar y depurar código durante el desarrollo de un sitio web o aplicación. Estos servidores suelen ser más pequeños y menos potentes que los servidores de producción, y se utilizan principalmente para propósitos internos.

Un servidor (máquina) de producción es más potente normalmente porque tiene que tratar con más tráfico. Nosotros en este proyecto sólo usamos un laptop como servidor (máquina) de desarrollo.

Pero luego está el servidor como software. Esto es simplemente el programa que se ejecuta sobre la máquina. Tanto en el servidor de desarrollo como en el de producción iría un apache.

En cuanto a los servidores de producción más utilizados, hay varias opciones populares, como:

* Servidores web Apache: Apache es uno de los servidores web más utilizados en todo el mundo y es especialmente popular en el desarrollo web.
* Servidores web Nginx: Nginx es otro servidor web popular que se utiliza ampliamente en el desarrollo web.
* Servidores web IIS: IIS es el servidor web de Microsoft y se utiliza principalmente en sistemas operativos Windows.
* Servidores web Lighttpd: Lighttpd es un servidor web ligero y rápido que se utiliza en algunos proyectos web.
* Servidores web Node.js: Node.js es un servidor web basado en JavaScript que se utiliza en algunos proyectos de desarrollo web.

En general, la elección del servidor web depende de las necesidades del proyecto y de las preferencias personales del desarrollador. En este proyecto se utilizará principalmente el servidor web Apache a través de la herramienta XAMPP en el equipo de desarrollo. Este software replica muy bien el entorno de un servidor de producción, lo que hará que la portabilidad entre ambos sea más sencilla.

## Servidores de bases de datos

En el desarrollo de aplicaciones web se utilizan principalmente dos tipos de servidores de bases de datos: servidores de bases de datos relacionales y servidores de bases de datos no relacionales.

Los servidores de bases de datos relacionales almacenan datos en tablas con filas y columnas y permiten la relación entre ellos a través de claves foráneas. Algunos de los servidores de bases de datos relacionales más utilizados son:

* MySQL: MySQL es uno de los servidores de bases de datos relacionales más populares y se utiliza ampliamente en el desarrollo web.
* Microsoft SQL Server: SQL Server es un servidor de bases de datos relacional de Microsoft y se utiliza principalmente en sistemas operativos Windows.
* Oracle Database: Oracle es un servidor de bases de datos relacional de empresa y se utiliza ampliamente en entornos empresariales.

Los servidores de bases de datos no relacionales, por otro lado, almacenan datos en documentos en lugar de tablas con filas y columnas. Algunos de los servidores de bases de datos no relacionales más utilizados son:

* MongoDB: MongoDB es uno de los servidores de bases de datos no relacionales más populares y se utiliza ampliamente en el desarrollo web.
* Cassandra: Cassandra es un servidor de bases de datos no relacional distribuido y se utiliza para manejar grandes cantidades de datos.
* Redis: Redis es un servidor de bases de datos no relacional en memoria y se utiliza para almacenar y recuperar datos rápidamente.

Al igual que con los servidores web, la elección del servidor de bases de datos depende de las necesidades del proyecto y de las preferencias personales del desarrollador. En este proyecto utilizaremos como servidor de base de datos relacional MySQL, a través de la herramienta de XAMPP.

## Tendencias y herramientas

En cuanto a las tendencias y herramientas emergentes en el ámbito del desarrollo web, cabe mencionar el uso cada vez más extendido de *frameworks* como React, Angular o JQuery que facilitan el desarrollo de aplicaciones web de una manera más rápida y sencilla. En este proyecto se utilizará JQuery por ser el más extendido y no necesitar las particularidades de los otros.

También están surgiendo nuevos lenguajes de programación como TypeScript, que combina las características de lenguajes como JavaScript y Java, y Go, que se caracteriza por su rendimiento y simplicidad. El soporte de TypeScript no está muy extendido a día de hoy, por tanto, se usará JavaScript por ser prácticamente universal.

En cuanto a los aspectos de una web, las tendencias actuales incluyen el diseño *responsive* para adaptarse a diferentes dispositivos, la accesibilidad para garantizar la usabilidad para todos los usuarios, el desarrollo basado en componentes para una mayor eficiencia, el desarrollo basado en el rendimiento para mejorar la experiencia del usuario, el desarrollo basado en la privacidad para proteger la información del usuario y el desarrollo basado en la nube para mayor flexibilidad y escalabilidad.

Por otra parte, en cuanto a la aplicación en su conjunto, se ha hecho un análisis de herramientas españolas que pueden mostrar datos en vivo de una playa a través de dispositivos IoT. Algunas opciones populares incluyen:

* Playas de España: Playas de España es una aplicación móvil desarrollada por el Ministerio de Transporte, Movilidad y Agenda Urbana de España que proporciona información sobre las playas españolas, incluyendo la calidad del agua y la presencia de servicios como duchas y aparcamientos (Ministerio para la Transición Ecológica y el Reto Demográfico, 2020).
* Surf-Forecast: Surf-Forecast es una plataforma de monitoreo de olas y condiciones de surf que proporciona datos en vivo de las playas de todo el mundo, incluyendo algunas playas españolas. Incluye información sobre la altura y la dirección de las olas, la temperatura del agua y el pronóstico del tiempo (Surf Forecast, 2022).

En general, estas herramientas se basan en dispositivos IoT instalados en la playa, como sensores meteorológicos y de oleaje, que recopilan y transmiten datos en tiempo real a través de la red.

# Especificaciones y planificación

## Especificaciones

El proyecto consiste en el diseño y desarrollo de un sitio web de visualización de datos de Smart Beaches. El objetivo del proyecto es crear una plataforma online que permita a los usuarios acceder a información en tiempo real sobre el estado de las playas, incluyendo la temperatura, índice ultravioleta (UV), humedad relativa y la presión atmosférica.

El público objetivo de la herramienta son principalmente los turistas y residentes de la Comunidad Valenciana interesados en conocer el estado de las playas y planificar sus visitas. También está enfocado a servir de herramienta de recopilación de datos para autoridades y empresas que deseen obtener información sobre el uso y la demanda de las playas.

El diseño del sitio web se basa en una maqueta visual y atractiva, con gráficos e imágenes interactivos que permiten a los usuarios visualizar la información de manera clara y sencilla donde la información esté al menor número de clics posible. Se ha tenido en cuenta la usabilidad y la accesibilidad en todas las etapas del proceso de diseño.

El contenido del sitio web se organizará en diferentes secciones, incluyendo una página de inicio con imágenes para cada una de las provincias, una sección de playas y datos y una sección de comentarios.

El sitio web debe tener un diseño fresco y atractivo, con una paleta de colores inspirada en el mar y la arena. Se deben usar tipografías legibles y se deben evitar imágenes o iconos que distraigan o confundan al usuario. Este debe ser compatible con diferentes dispositivos móviles. Debe tener una estructura responsive para adaptarse a diferentes tamaños de pantalla. Debe también cumplir con las normativas de accesibilidad y privacidad.

En cuanto a las especificaciones técnicas, se ha elegido utilizar los más estándares con el fin de que a la hora de encontrar un *hosting* sea sencillo que este soporte todas las herramientas utilizadas en la programación y el diseño del sitio web. el lenguaje de programación HTML, JavaScript, CSS, MySQL y PHP para el desarrollo del sitio web. Se utilizará una base de datos ThingSpeak para almacenar los datos recibidos de los nodos IoT de las Smart Beaches. Así como mostrar los últimos valores y el histórico de los datos recibidos.

## Planificación

A continuación, se ilustrará la planificación seguida:

1. Reunión de inicio del proyecto: En esta reunión se definirán los objetivos del proyecto, se establecerán los plazos y se asignarán los roles y responsabilidades.
2. Diseño de la arquitectura de la información: En este paso se definirá la estructura de la página web y se crearán los diagramas de flujo de la información.
3. Diseño de la interfaz de usuario: Se crearán los *wireframes* y maquetas de la interfaz de usuario y se realizarán pruebas de usabilidad.
4. Pruebas de conexión con los nodos IoT: Se realizarán las pertinentes pruebas con ThingSpeak para comprobar la correcta comunicación con la API Rest.
5. Desarrollo del sitio web: Se codificará el frontend del sitio web utilizando HTML, CSS y JavaScript. También se integrará en el backend con cualquier otro sistema necesario, como bases de datos a través de PHP.
6. Pruebas y depuración: Se realizarán pruebas para asegurar la compatibilidad y se corregirán cualquier error o problema encontrado.
7. Mantenimiento y actualización: Se realizarán tareas de mantenimiento periódicas y se añadirán nuevas funcionalidades o se actualizará el contenido del sitio web según sea necesario.

# Metodología

## Herramientas

A continuación, se explicarán las diferentes herramientas utilizadas a lo largo de este proyecto, así como la metodología del trabajo seguida.

* **GitHub:**

GitHub (GitHub Inc., 2022) es una plataforma de alojamiento de código fuente y colaboración en línea para proyectos de software. Es utilizado por desarrolladores, diseñadores y científicos de datos para almacenar y administrar sus proyectos de software, compartirlos con otros miembros del equipo y colaborar en el desarrollo de aplicaciones (ver Ilustración 2).

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración . GitHub

GitHub utiliza el sistema de control de versiones Git para administrar los cambios en el código fuente y proporciona una serie de herramientas y características para facilitar la colaboración y el trabajo en equipo. Por ejemplo, permite a los miembros del equipo crear y discutir problemas, realizar revisiones del código y crear ramas de trabajo para probar nuevas funcionalidades.

* **Docker:**

Docker (Docker Inc., 2022) es una plataforma de contenedores que permite a los desarrolladores empaquetar y distribuir aplicaciones de manera rápida y sencilla. Un contenedor es una forma de aislar una aplicación y todas sus dependencias, como librerías y archivos de configuración, en un paquete que puede ser ejecutado en cualquier entorno de forma consistente.

Al utilizar contenedores, es posible asegurar que una aplicación funcionará de la misma manera en diferentes entornos, como en el entorno de desarrollo de un equipo, en un servidor de producción o en la nube. Esto hace que sea más fácil desarrollar, probar y desplegar aplicaciones, especialmente en entornos distribuidos o en la nube.

Docker proporciona una herramienta de línea de comandos y una API que permiten a los desarrolladores crear, enviar y ejecutar contenedores de forma sencilla. También ofrece una plataforma en línea, Docker Hub, que permite a los desarrolladores compartir y descargar contenedores creados por otros usuarios.

En este proyecto será utilizado por GitHub para ejecutar acciones sobre el código, como ejecución de tests y análisis estático, como se verá más adelante.

* **IDE:**

IDE es el acrónimo de Integrated Development Environment, que se traduce al español como "Entorno de Desarrollo Integrado". Un IDE es una herramienta de software que se utiliza para desarrollar aplicaciones y programas informáticos. Un IDE suele incluir un editor de código, un depurador, un compilador y otras herramientas útiles para el desarrollo de software (Code Academy, 2020).

Visual Studio Code es uno de los IDEs más populares y utilizados en la actualidad. Este es el IDE que utilizaremos en este proyecto (ver Ilustración 3). Algunas de sus características más importantes son (VS Code, Miscrosoft, 2022):

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración . Área de trabajo VSCode

1. Editor de código: Visual Studio Code incluye un editor de código potente y personalizable con soporte para varios lenguajes de programación.
2. Depuración: Visual Studio Code incluye un depurador integrado que permite a los desarrolladores encontrar y solucionar errores en el código.
3. Integración con Git: Visual Studio Code está integrado con Git, lo que permite a los desarrolladores realizar operaciones de control de versiones de manera fácil y rápida.
4. Extensiones: Visual Studio Code tiene una amplia gama de extensiones disponibles que permiten a los desarrolladores personalizar y ampliar las funcionalidades del IDE.
5. Portabilidad: Visual Studio Code es una aplicación de escritorio que se ejecuta en varias plataformas, como Windows, Mac y Linux.
6. Diseño moderno: Visual Studio Code tiene una interfaz de usuario moderna y atractiva que facilita el trabajo de los desarrolladores.

* ***Browser Inspector*:**

El inspector de navegador es una herramienta que se utiliza para examinar y depurar el código HTML, CSS y JavaScript de una página web. Está integrado en la mayoría de los navegadores modernos, como Google Chrome, Mozilla Firefox, Safari y Microsoft Edge, y se accede a él mediante una opción del menú del navegador o mediante el uso de atajos de teclado.

Con el inspector de navegador, es posible ver el código fuente de una página web, modificar el CSS y el HTML en tiempo real y ver cómo afectan esos cambios al diseño y al comportamiento de la página. También puedes utilizar el inspector para depurar problemas de JavaScript, examinar los datos de las solicitudes de red y ver los errores que ocurren en la página. Es una herramienta muy útil para desarrolladores web y diseñadores que necesitan examinar y optimizar el código de una página web.

* **ESLint:**

ESLint es una herramienta de *linting* de JavaScript, es decir, una herramienta que analiza el código fuente de un programa y busca posibles errores o problemas de estilo. Es una forma de asegurar que el código cumpla con un conjunto de reglas predefinidas y se mantenga un estilo consistente a lo largo del proyecto. ESLint se utiliza comúnmente en proyectos de JavaScript para asegurar la calidad del código y evitar errores comunes (ESLint, 2022).

ESLint se basa en reglas configurables, que pueden ser estrictas o suaves dependiendo de las necesidades del proyecto. Las reglas pueden ser personalizadas y se pueden utilizar módulos de terceros que añaden reglas adicionales. Además, ESLint es compatible con plugins que añaden soporte para lenguajes adicionales, como TypeScript y JSX.

Se puede integrar con diferentes entornos de desarrollo y se puede configurar para que se ejecute automáticamente cada vez que se guarda un archivo o se realiza un *commit* en un control de versiones. En este proyecto esto se realizará en GitHub a través de Docker.

* **Node.js y Jest:**

Jest es un marco de pruebas de JavaScript diseñado para ser fácil de usar y rápido en la ejecución de pruebas. Es muy popular entre los desarrolladores de JavaScript y se utiliza ampliamente en proyectos de React y Node.js.

Jest se ejecuta en Node.js, un entorno de tiempo de ejecución de JavaScript que se ejecuta en el servidor en lugar de en un navegador. Jest se integra con Node.js y permite a los desarrolladores escribir pruebas para sus aplicaciones de Node.js de manera sencilla.

Con Jest los desarrolladores pueden escribir pruebas unitarias (un tipo de pruebas de software que se utilizan para asegurar que cada unidad o componente de una aplicación funciona correctamente por sí misma), pruebas de integración y pruebas de aceptación de manera sencilla. Además, Jest incluye una serie de características útiles, como el *snapshot testing* y la automatización de *mocks*, que facilitan la escritura y la ejecución de pruebas. Jest es una herramienta muy útil para garantizar la calidad y la estabilidad de una aplicación de Node.js.

En este proyecto se utilizará esta herramienta para realizar test unitarios en Jest ejecutados sobre Node.js y este sobre Docker, lanzado a su vez desde github.

* **XAMPP:**

XAMPP es una distribución de Apache que incluye PHP y MySQL. Se utiliza comúnmente para desarrollar aplicaciones web localmente en una computadora, ya que proporciona un entorno de servidor local completo y fácil de usar.

XAMPP se puede instalar en sistemas operativos Windows, MacOS y Linux y se utiliza para crear un entorno de desarrollo de aplicaciones web localmente. Una vez instalado, XAMPP proporciona un servidor Apache con PHP y MySQL instalados y configurados tal y como se muestra en la Ilustración 4, lo que permite a los desarrolladores crear y probar aplicaciones web sin tener que configurar un servidor remoto.

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración . Panel de control de XAMPP

XAMPP es muy popular entre los desarrolladores web debido a su facilidad de uso y a la amplia gama de herramientas que ofrece. Además de Apache, PHP y MySQL, XAMPP también incluye otros componentes como PHPMyAdmin, y una herramienta de gestión de bases de datos MySQL.

## Gitflow

Git es un sistema de control de versiones distribuido que se utiliza para llevar un registro de los cambios en archivos y directorios. Git permite a los desarrolladores trabajar en proyectos de manera colaborativa y mantener un historial de todas las modificaciones realizadas (Git, 2022).

Gitflow es una metodología de trabajo basada en Git que se utiliza para organizar el flujo de trabajo de un proyecto.

A continuación, se explicarán diferentes conceptos que han sido claves para la realización del proyecto, así como para la metodología de trabajo.

* Repositorio: es un lugar donde se almacenan los archivos de un proyecto y se lleva un registro de todos los cambios realizados en ellos. Cada repositorio de Gitflow consta de dos ramas principales: la rama principal (también conocida como "master") y la rama de desarrollo (también conocida como *"develop*", o cualquier otro nombre). Además, Gitflow utiliza otras ramas para la creación y el mantenimiento de versiones y características.
* Ramas: en Git, las ramas son copias de un repositorio que se utilizan para trabajar en cambios de manera aislada. En Gitflow se utilizan ramas para controlar el flujo de trabajo y el proceso de desarrollo. Por ejemplo, se pueden crear ramas de características para trabajar en nuevas funcionalidades del proyecto y ramas de liberación para preparar versiones para su liberación al público. Estas ramas se fusionan eventualmente con la rama de desarrollo o la rama principal cuando están listas para ser incluidas en el proyecto.
* Git Bash: es una interfaz de línea de comandos para Git que se ejecuta en la ventana de comandos de Windows o en la terminal de Unix. Git Bash proporciona acceso a todos los comandos de Git y permite a los desarrolladores interactuar con el repositorio de Gitflow de manera sencilla (ver Ilustración 5).

Texto

Descripción generada automáticamente

Ilustración . Terminal Git Bash

* *Pull request*: en Git, un "*pull request*" es una solicitud para fusionar una rama con otra. Por lo general, se utiliza para solicitar a otro desarrollador que revise y apruebe los cambios realizados en una rama antes de fusionarlos con otra rama. Por ejemplo, si se ha trabajado en una rama de característica y se quiere fusionar esa rama con la rama de desarrollo, se puede crear un *pull request* para solicitar a otros desarrolladores que revisen y aprueben tus cambios antes de fusionarlos. Los *pull requests* son una parte importante del proceso de revisión de código y garantizan que el código añadido al proyecto sea de alta calidad.

En este proyecto se han utilizado *pull request* para solicitar revisiones críticas del código con el fin de mejorar la calidad del resultado final (ver Ilustración 6).

Captura de pantalla con la imagen de una pantalla

Descripción generada automáticamente

Ilustración . Pull request

## CI/CD

El concepto de *continuous integration* (CI) y *continuous development* (CD) se refiere a un enfoque de desarrollo de software que busca integrar y probar de manera continua los cambios en el código fuente de una aplicación.

En CI, se utilizan herramientas automatizadas para integrar y probar el código cada vez que se realiza un cambio en el repositorio de código fuente. Esto permite detectar problemas de integración y errores de manera temprana, lo que facilita la identificación y corrección de problemas.

En CD, se lleva a cabo una entrega continua de cambios al código de producción, lo que permite que los cambios se implementen de manera más rápida y sin interrupciones para el usuario final. Esto se logra mediante la automatización de tareas de integración, pruebas e implementación, lo que permite una entrega más rápida y sostenible de nuevas funcionalidades.

En resumen, CI y CD son enfoques de desarrollo de software que buscan mejorar la calidad y la eficiencia del proceso de desarrollo mediante la integración y prueba continua del código y la entrega continua de cambios al código de producción.

Github permite implementar tareas de CI/CD mediante lo que llama *actions*, que son automatizaciones de tareas por medio de scripts. En este proyecto se utilizarán para lanzar tests unitarios y análisis estático de código sobre cada *commit*.

Los tests unitarios son pruebas que se realizan para verificar el correcto funcionamiento de una unidad individual de código, como una función o un módulo. Estos tests suelen ser muy específicos y se utilizan para asegurar que cada unidad de código esté funcionando correctamente de manera aislada, sin tener en cuenta su integración con el resto del sistema.

Los tests unitarios se utilizan comúnmente durante el desarrollo de software para asegurar que el código cumple con los requisitos especificados y para detectar errores temprano en el proceso de desarrollo. Además, los tests unitarios también pueden ser útiles para detectar cambios en el comportamiento de una unidad de código debido a cambios en el código fuente, lo que puede ser útil para evitar errores y problemas en el futuro.

En general, la realización de tests unitarios es una práctica importante en el desarrollo de software debido a que puede ayudar a mejorar la calidad del código y a reducir el riesgo de errores en el sistema.

En este proyecto, como se ha comentado, se han implementado mediante *actions* de github que, para cada commit, levanta un Docker con node.js, se descarga el repositorio y lanza jest sobre los test unitarios (ver Ilustración 7).

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración . Test unitarios. Action GitHub

El análisis estático de código es una técnica que se utiliza para examinar el código fuente de una aplicación sin ejecutarlo. Se puede utilizar para detectar errores, problemas de diseño y vulnerabilidades de seguridad en el código, así como para aplicar reglas de estilo y buenas prácticas de programación.

Una herramienta de análisis estático de código se conoce comúnmente como un lint (en inglés). Un lint es un programa que analiza el código fuente de una aplicación y busca problemas o patrones que puedan indicar errores o problemas de diseño. Por ejemplo, un lint puede detectar variables no utilizadas, llamadas a funciones que no existen o problemas de sintaxis en el código.

El análisis estático de código puede ser muy útil en el desarrollo de software ya que permite detectar problemas de manera temprana y corregirlos antes de que el código se ejecute. Además, también puede ayudar a mejorar la calidad del código y a asegurar que se cumplan las reglas de estilo y buenas prácticas de programación.

Al igual que los test unitarios, este análisis se ha configurado en las acciones de GitHub para que se ejecute cada vez que se lance un *commit* en el servidor.

# Diseño

## Diseño gráfico

El diseño gráfico es un elemento fundamental en la creación de un sitio web, ya que es el encargado de darle una identidad visual y de transmitir de manera efectiva la información y el mensaje que se quiere transmitir.

La paleta de colores es uno de los elementos más importantes del diseño gráfico, ya que cada color tiene un significado y puede transmitir diferentes emociones y sensaciones. Es importante seleccionar una paleta de colores que sea coherente con la imagen y el mensaje que se quiere transmitir.

Para este proyecto se han escogido dos colores, el amarillo y el gris (ver Ilustración 8). El amarillo es una opción atractiva para el diseño gráfico de una empresa, ya que transmite sensaciones de accesibilidad y cercanía, y es el color más alegre de todos. Su uso puede aportar un toque extra de energía, calidez y juventud a la marca. Algunas empresas reconocidas como Ferrari, Ikea y Shell han elegido el amarillo para comunicar felicidad, optimismo y despreocupación a través de su imagen. El gris es un color que se obtiene al mezclar el negro y el blanco. Es un tono neutral que puede resultar equilibrado y sofisticado, y se puede considerar tanto masculino como femenino dependiendo de su claridad u oscuridad. Algunas de las características asociadas al gris son la madurez, la solemnidad, la formalidad y la conservación. Es una opción atractiva para transmitir una imagen seria y madura en el diseño gráfico.

Imagen que contiene Logotipo

Descripción generada automáticamente

Ilustración . Rótulo de Smart Beach CV

La tipografía es otro elemento clave del diseño gráfico. La elección de la fuente y el tamaño de letra pueden afectar significativamente la legibilidad y la estética del sitio web. Es importante seleccionar una tipografía que sea legible y coherente con la imagen del sitio.

Las dos tipografías utilizadas en este sitio web han sigo Bebas Neue y Montserrat. Bebas Neue y Montserrat son unas fuentes tipográficas sans-serif que se han vuelto muy populares en los últimos años, especialmente en el diseño gráfico y la publicidad. Estas fuentes se caracterizan por tener una forma clara y legible, con un estilo minimalista y moderno.

Al utilizarlas, se puede transmitir una imagen de simplicidad, claridad y modernidad. También se pueden utilizar para dar un toque de frescura y originalidad a la marca (Tsunekawa, 2005).

El uso de imágenes y otros elementos visuales también es importante en el diseño gráfico. Las imágenes pueden ayudar a transmitir el mensaje de manera más efectiva y a hacer que el sitio web sea más atractivo visualmente (ver Ilustración 9). Para este proyecto se han utilizado imágenes libres de derechos.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración . Selección de las comunidades

## Diseño de la interfaz de usuario (UI)

El diseño de interfaz de usuario (UI, por sus siglas en inglés) es la parte de la experiencia del usuario en un producto o servicio que se relaciona con la interacción visual y táctil del usuario con el mismo. Se encarga de crear una interfaz fácil de usar y atractiva para el usuario, y debe tener en cuenta el contexto en el que se utilizará el producto o servicio.

Para diseñar una interfaz de usuario efectiva es importante considerar los siguientes aspectos:

* Usabilidad: la interfaz debe ser fácil de usar y entender para el usuario.
* Diseño visual: el diseño visual debe ser atractivo y coherente con el propósito del producto o servicio.
* Consistencia: la interfaz debe ser consistente en su uso de elementos visuales y de interacción, lo que ayuda a que el usuario se sienta cómodo y entienda cómo funciona el producto o servicio.
* Flexibilidad: la interfaz debe ser flexible y adaptarse a las necesidades y preferencias del usuario.

El sitio web se ha diseñado con el fin de que no se necesite un “manual de usuario” para navegar por él. Se han dividido las playas por comunidades con el fin de que sea más sencillo para el usuario (ver Ilustración 10).

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Ilustración . Interfaz de usuario

En resumen, el diseño de interfaz de usuario es una parte importante de cualquier producto o servicio, ya que afecta directamente a la experiencia del usuario y su satisfacción con el mismo .

## Diseño de la experiencia de usuario (UX)

El diseño de la experiencia del usuario (UX, por sus siglas en inglés) se refiere a la forma en que una persona siente e interactúa con un producto o servicio. Se encarga de crear una experiencia de usuario positiva y coherente a lo largo de todo el ciclo de vida del producto o servicio, desde la investigación y el diseño hasta la implementación y el mantenimiento.

Para diseñar una experiencia de usuario efectiva es importante considerar los siguientes aspectos:

* Investigación: es fundamental entender las necesidades, deseos y expectativas de los usuarios para poder crear una experiencia que satisfaga sus requisitos.
* Diseño: se trata de crear una estructura y una interacción que sean fáciles de usar y atractivas para el usuario.
* Evaluación: es importante evaluar la experiencia del usuario durante y después de su uso para poder mejorarla y adaptarla a las necesidades de los usuarios.

Con la estructura elegida se ha intentado que la experiencia del usuario sea agradable y que se llegue a la información sobre la playa deseada en el menor número de clics posible. Con las futuras actualizaciones también se intentará mejorar esta experiencia.

## Diseño adaptativo

El diseño adaptativo es un enfoque de diseño web que se centra en crear páginas que se adapten de forma automática al tamaño y las capacidades de los dispositivos de los usuarios, como computadoras de escritorio, tabletas y teléfonos móviles. Esto permite que las páginas se vean bien y sean fáciles de usar en todos los dispositivos, independientemente de su tamaño y resolución.

Uno de los principales elementos del diseño adaptativo es el uso de la regla @media de CSS (hojas de estilo en cascada). Esta regla permite definir estilos específicos para diferentes tipos de dispositivos y tamaños de pantalla, lo que permite crear una experiencia de usuario óptima para cada dispositivo. Por ejemplo, se pueden utilizar diferentes estilos para dispositivos móviles y de escritorio, o para pantallas de diferentes tamaños.

En resumen, el diseño adaptativo es un enfoque de diseño web que se centra en crear páginas que se adapten automáticamente al tamaño y las capacidades de los dispositivos de los usuarios, y que utiliza la regla @media de CSS para definir estilos específicos para diferentes dispositivos y tamaños de pantalla. Además, esto nos permite prescindir de diseñar una aplicación con un solo navegador y con una única página.

## Diseño de la navegación

El diseño de la navegación se refiere a cómo se presenta y se organiza la información en un sitio web o aplicación para que sea fácil de encontrar y usar para el usuario.



Ilustración . Header

Se encarga de crear una estructura lógica y coherente que guíe al usuario a través del sitio y le permita encontrar la información que busca de manera eficiente.

El encabezado del sitio web contiene cuatro apartados (*Home*, Playas, Top 4, Contacto) (ver Ilustración 11):

* El apartado de *Home* conduce a la página principal. El apartado de playas conduce a la sección en la que permite al usuario elegir comunidad (Castellón, Valencia, Alicante) (ver Ilustración 12).

Una captura de pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Ilustración . Sección Playas, página principal

* El apartado TOP 4 de la página principal muestra las playas más visitadas de todas las comunidades (ver Ilustración 13).

Interfaz de usuario gráfica, Aplicación, Word, PowerPoint

Descripción generada automáticamente

Ilustración . Sección TOP 4, página principal

* El apartado Contacto es un formulario que permite al usuario dejar un comentario sobre la playa que ha visitado y este se almacenará en una sección de reseñas de cada comunidad (ver Ilustración 14).

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Ilustración . Formulario de comentarios

Al seleccionar una comunidad hay pequeños cambios en la estructura. Como ejemplo seleccionaremos la provincia de Alicante.

* El apartado de playas al elegir una comunidad muestra un desplegable con todas las playas de dicha comunidad. Una vez elegida la playa deseada se muestra los datos de esta (Temperatura, humedad relativa, índice ultravioleta y presión atmosférica) (ver Ilustración 15).

Texto

Descripción generada automáticamente con confianza baja

Ilustración . Datos actuales playa

Al lado de estos datos se encuentra un botón que despliega una gráfica con el histórico de los datos (ver Ilustración 16). Se puede seleccionar que datos mostrar, así como el periodo que mostrar.

Interfaz de usuario gráfica, Gráfico, Gráfico de líneas

Descripción generada automáticamente

Ilustración . Gráfica con histórico de datos

* El apartado del TOP 4 al elegir la comunidad se filtra por comunidades, por lo que solo se muestran las playas más visitadas de la comunidad seleccionada.
* Al haber seleccionado la comunidad ahora aparece un apartado de reseñas, en el que se muestran pequeñas tarjetas con solamente los comentarios de la playa seleccionada, el nombre del usuario que lo ha escrito y la fecha (ver Ilustración 17).

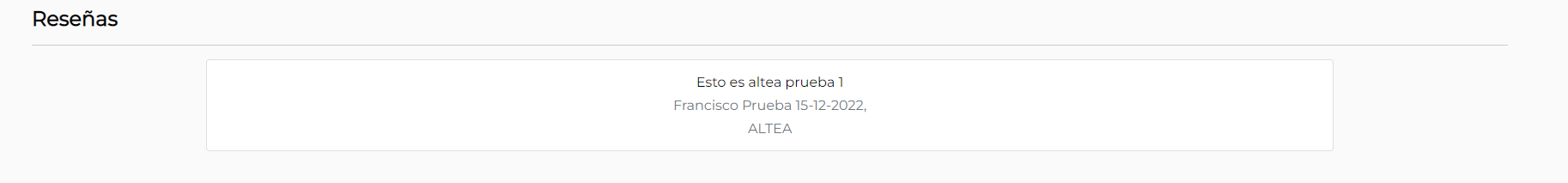


Ilustración . Sección de reseñas

* El apartado de Contacto continúa siendo el mismo que el de la página principal.

# Implementación y pruebas

## Implementación

Para la implementación de nuestro proyecto de diseño web hemos decidido utilizar HTML, CSS, JavaScript y PHP como lenguajes, tal y como se vio en el apartado de estado del arte. Estos nos permitirán estructurar y dar formato al contenido del sitio web, así como añadir funcionalidades interactivas y conectarnos a las bases de datos (ver Ilustración 1).

Los dos archivos principales PHP son main.php y graph.php. El primero es la *landing page*, donde se encuentra la selección de la comunidad, véase Anexo I archivo main.php.

En el segundo, un poco más extenso, se encuentra el HTML de la selección de las playas, el código para recibir los datos y mostrarlos, ya sea como iconos o en gráficas y el código que guarda la actividad de usuario, véase Anexo I archivo graph.php.

Diagrama

Descripción generada automáticamente

Ilustración . Distribución código

Hay algunos archivos como header.php, footer.php y comments.php que se han separado de las páginas principales con el fin de mejorar la mantenibilidad (ver Ilustración 18).

En cuanto al diseño del código, a continuación, entraremos en detalle y comentaremos algunos de los aspectos más importantes:

* AJAX: Ajax es un término genérico que se refiere a la técnica de programación web que permite a una página web actualizar parte de su contenido de forma asíncrona. En este proyecto utilizaremos fetch(), una forma de realizar Ajax utilizando JavaScript, pero no es la única forma de hacerlo. También se pueden utilizar otras herramientas y tecnologías para implementar Ajax, como XMLHttpRequest o librerías de terceros como jQuery.

Fetch es más compatible con promesas, lo que significa que se pueden utilizar métodos como .then() y .catch() para manejar la respuesta de la solicitud. XMLHttpRequest, por otro lado, utiliza un sistema de manejo de eventos para manejar la respuesta de la solicitud (ver Ilustración 19).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Petición fetch()

Esto se encuentra en el archivo thingspeak.js donde tenemos la conexión con la plataforma ThingSpeak una plataforma de Internet de las cosas (IoT, por sus siglas en inglés) que permite a los usuarios recopilar y almacenar datos de dispositivos IoT en la nube y luego visualizar y analizar esos datos mediante gráficos y herramientas de análisis (MathWorks, 2022).

* Estructura de clases: es una característica de la programación orientada a objetos (POO) que permite crear plantillas para objetos que comparten atributos y comportamientos. En la estructura de clases, se define una clase como una plantilla para crear objetos, y cada objeto creado a partir de esa clase se conoce como una instancia de la clase.

En este proyecto esto se ha utilizado para crear instancias y con estas llamar a métodos de las clases para que el objeto realice ciertas acciones, en el archivo thingspeak.js creamos diferentes url para realizar la petición fetch(). Estas acciones concretamente realizaran diferentes tipos de llamadas a ThingSpeak, con las que podemos conseguir filtrar los datos de los nodos (último dato, datos en una franja horaria, escoger algún dato en concreto, etc…) (ver Ilustración 20).

Texto

Descripción generada automáticamente

Ilustración . URL para realizar la petición fetch()

* Parser: en el archivo parser.js tenemos otro ejemplo de la estructura de clases (ver Ilustración 21), aquí se realiza el filtrado de los datos recibimos por la petición fetch(). El formato de los datos los recibimos en formato JSON (ver Ilustración 22).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Estructura de clases (parser.js)

Texto

Descripción generada automáticamente

Ilustración . Formato JSON

El campo que nos interesa es el field4 una secuencia de números (ver Ilustración 23).

Texto

Descripción generada automáticamente

Ilustración . Campos field4

En el archivo parser.js se realiza el parseo automático por JavaScript a un objeto (ver Ilustración 24).

Texto

Descripción generada automáticamente

Ilustración . Parseado datos

* Gráfico de datos: para la visualización de los datos en la gráfica hemos escogido una biblioteca llamada highcharts (Highcharts, 2022), con ella podemos mostrar un array de datos en una gráfica interactiva. Esto se ha realizado en el archivo graph.js (véase Anexo I. Código fuente).
* Etiquetas *Select* : para integrar los datos en estas listas de selección hemos utilizado las bases de datos de MySQL (ver Ilustración 25). Primero se ha tenido que configurar la conexión con esta base de datos, tal y como se muestra en la Ilustración 26.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Ilustración . Beach database

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Configuración database

Una vez creada la conexión se ha utilizado la estructura de clases para crear diferentes métodos con los cuales comunicarnos con estas bases de datos y utilizar dicha información (ver Ilustración 27).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Métodos database

Con esto se ha conseguido que los elementos select se visualicen como en la Ilustración 28.

Texto

Descripción generada automáticamente con confianza media

Ilustración . Elemento select de las playas

* Formulario de contacto: para realizar el formulario de contacto también se ha utilizado la conexión con la base de datos (ver Ilustración 29).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Conexión base de datos, formulario de contacto

* Seguridad: hay varios tipos de ataques que habría que tener en cuenta. La inyección SQL es un tipo de ataque que se aprovecha de vulnerabilidades en la seguridad de una base de datos para enviar comandos maliciosos a través de una consulta SQL. El objetivo del atacante puede ser modificar o acceder a información confidencial almacenada en la base de datos.

El ataque de inyección XSS es un tipo de ataque que se aprovecha de vulnerabilidades en la seguridad de un sitio web para inyectar código malicioso en el sitio, que luego es ejecutado por los usuarios que acceden al sitio. El objetivo del atacante puede ser robar información confidencial de los usuarios o redirigir a los usuarios a sitios maliciosos (Avast, 2021).

Es importante tomar medidas para protegerse contra estos tipos de ataques, como validar y sanear adecuadamente las entradas de datos y utilizar contraseñas seguras. También es importante mantener actualizado el software y utilizar herramientas de seguridad para detectar y prevenir estos tipos de ataques.

Uno de los métodos de seguridad que hemos utilizado ha sido la sanitización de entradas, el proceso de limpiar y filtrar los datos proporcionados por el usuario para asegurar que son seguros de usar o mostrar. Esto implica verificar los datos en busca de caracteres o código potencialmente peligroso y eliminarlos o escaparlos. Sanitizar entradas es importante para prevenir ataques maliciosos y es comúnmente utilizado en el desarrollo web y en otras aplicaciones donde se utilizan datos proporcionados por el usuario (ver Ilustración 30).

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Ilustración . Sanitización de entradas en formulario

## Pruebas funcionales

Las pruebas funcionales de este proyecto han sido importantes para asegurar que el código del sitio web es de alta calidad y está libre de errores.

Para llevar a cabo las pruebas de código, debemos definir un conjunto de reglas y estándares de calidad que deben seguirse al escribir el código del sitio web. Estos deben incluir información sobre la estructura y formato del código, la legibilidad y el uso de buenas prácticas de programación.

Es importante también utilizar herramientas de análisis de código para verificar el cumplimiento de estas reglas y estándares. Estas herramientas pueden incluir linters, que se encargan de verificar el formato y estructura del código, y analizadores de código estático, que se encargan de verificar la calidad y seguridad del código (ver 4.1).

Empezaremos con los comentados anteriormente test unitarios. Para este proyecto solamente se han podido implementar en una parte del código, el parseo de la secuencia de los datos (ver Ilustración 31). Se ha creado una secuencia de datos de muestra, así como los valores que deberían salir y mediante la depuración del código en el terminal integrado de VSCode se comprobará si la programación del parseo se ha hecho correctamente (ver Ilustración 32).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Test unitario de parseo

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Resultado test unitario parseo

Como se ha comprobado en la Ilustración 31 el resultado de este test unitario ha sido satisfactorio lo que indica que la programación del parseo ha sido correcta.

A continuación, comentaremos las pruebas realizadas para verificar la correcta utilización de los estándares y el formato de programación.

Para ello se ha utilizado la herramienta ESLint para la cual se ha utilizado la revisión *standard.* Esta tiene en cuenta diferentes reglas para el formato:

* Sin tabulaciones → 2 espacios
* Comas innecesarias
* Líneas en blanco
* Variables que no se utilizan

Si el análisis del código detecta alguna de estas reglas, este sale *failed* en caso contrario sale *succeeded* (ver Ilustración 33).

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración . Proceso arreglo formato

Este análisis se ejecuta cada vez que se realiza un *commit* (ver 4.1 e Ilustración 34).

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración . Ejecución ESLin

# Conclusiones

En este proyecto, nuestro objetivo principal era desarrollar una plataforma web que permitiera visualizar y analizar de manera sencilla y efectiva los datos recogidos por los sensores instalados en las Smart Beach. Además, queríamos conseguir que la plataforma fuera responsive, es decir, que se adaptara correctamente a distintos tamaños de pantalla y dispositivos. También queríamos utilizar las herramientas más actuales que nos permitieran aumentar la eficacia y la seguridad del código. Adicionalmente queríamos ser capaces de realizar la conexión con los nodos IoT, recibir los datos y procesarlos utilizando la tecnología AJAX. Todo esto utilizando la metodología Gitflow como metodología de trabajo y GitHub como plataforma de control de versiones.

Esta metodología nos ha permitido dividir el trabajo en ramas separadas y manejar los cambios de manera sencilla. Además, la utilización de tags y versiones nos ha ayudado a controlar y documentar los distintos estados del proyecto.

En general, consideramos que la metodología Gitflow ha sido muy útil para el proyecto. Nos ha permitido trabajar de manera más eficiente y organizada y ha contribuido a que el proceso de desarrollo fuera más fluido.

En cuanto a los resultados obtenidos, consideramos que la plataforma web desarrollada cumple con los objetivos que nos habíamos propuesto. La plataforma es responsive y se adapta correctamente a distintos tamaños de pantalla y dispositivos. Además, hemos utilizado la tecnología AJAX para la correcta comunicación con los nodos IoT. Los gráficos dinámicos permiten una fácil interpretación de los datos y pueden ser muy útiles para la toma de decisiones. La utilización de distintos lenguajes de programación nos a permitido conocer mejor las exigencias del mercado actual, así como las últimas tecnologías en el ámbito de gestión de proyectos.

Sin embargo, también hemos identificado algunas debilidades en el proyecto. Por ejemplo, en algunos casos, la carga de datos puede ser lenta y requerir algún tiempo. En el futuro, podríamos trabajar en optimizar la carga de datos para mejorar la eficiencia de la plataforma.

En general, consideramos que este proyecto ha sido muy enriquecedor y hemos adquirido importantes conocimientos y habilidades que esperamos poder aplicar en proyectos similares en el futuro. Estamos seguros de que la plataforma desarrollada será una herramienta muy útil para la visualización y análisis de los datos de las Smart Beach y que contribuirá a la toma de decisiones en el futuro.

# Bibliografía

Avast. (2021). *¿Qué es la inyección de SQL?* Obtenido de Avast Academy: https://www.avast.com/es-es/c-sql-injection

Code Academy. (2020). *What Is an IDE?*

Docker Inc. (2022). *Docker*. Obtenido de docker.com

ESLint. (2022). *ESLint*. Obtenido de https://eslint.org/docs/latest/user-guide/getting-started

Git. (2022). *Git*. Obtenido de https://git-scm.com/

GitHub Inc. (2022). *GitHub*. Obtenido de github.com

Google Fonts. (s.f.). *Bebas Neue*. Obtenido de https://fonts.google.com/specimen/Bebas+Neue?query=bebas+neue

Highcharts. (2022). *Highcharts*. Obtenido de https://www.highcharts.com/

Instituto Valenciano de Tecnologías Turísticas. (2018). *La transformación digital. El modelo de playas inteligentes de la Comunitat Valenciana.* Comunitat Valenciana: Turisme Comunitat Valenciana. Obtenido de Generalitat Valenciana: https://www.turismecv.com/wp-content/uploads/2018/11/Ebook-playas-inteligentes-OK.pdf

MathWorks. (2022). *ThingSpeak*. Obtenido de https://thingspeak.com/

Ministerio para la Transición Ecológica y el Reto Demográfico. (2020). *Guía de playas*. Obtenido de https://datos.gob.es/es/aplicaciones/guia-de-playas

Pastor, L. M. (2019). Un proyecto basado en el internet de las cosas (IoT) aplicado a las playas de Benidorm, Gandía y Benicàssim. Equipamiento y servicios municipales. . *Dialnet*, 70-74.

Surf Forecast. (2022). *Surf Forecast*. Obtenido de https://www.surf-forecast.com/

The PHP Group. (2001 - 2022). *PHP*. Obtenido de php.net

Tsunekawa, R. (2005). *Adobe Fonts*. Obtenido de https://fonts.adobe.com/fonts/bebas-neue-pro?locale=es#about-section

VS Code, Miscrosoft. (2022). *Visual Studio Code*. Obtenido de https://code.visualstudio.com/

W3C. (2009 - 2017). *W3C*. Obtenido de w3c.org

# Anexo I. Código fuente

* Archivo main.php:

<?php include 'inc/header.php'?>

  <link rel="stylesheet" href="css/style\_main.css">

</head>

<body>

  <!-- Navbar -->

  <div class="navBar">

    <!-- navMenu -->

    <div class="navMenu">

      <a href="#title">home</a>

      <a href="#playas">playas</a>

      <a href="#top\_4">top 4</a>

      <a href="#contacto">contacto</a>

      <div class="dot"></div>

    </div>

    <!-- Header -->

    <header id="title">

      <a href="main.php" id="header-title">SMART BEACH COMUNITAT VALENCIANA</a>

    </header>

  </div>

  <!-- Page content -->

    <!-- Playas Section-->

  <div class="content">

    <div class="section">

      <h3 class="section-1" id="playas">Playas</h3>

      <p>

        Puedes navegar por las diferentes playas de Valencia, las playas de Alicante y las playas de Castellón.

        Si lo prefieres, selecciona tus preferencias en el menú de la izquierda o

        busca por nombre de playa en el formulario que encontrarás justo por encima de dicho menú.

        También puedes ver cada una de las playas en el mapa de playas de la Comunidad Valenciana. Además,

        podrás enviarnos las mejores fotos de tus playas de la Comunidad Valenciana preferidas.

        Para ello sólo tienes que navegar hasta la playa y subir tu foto.

      </p>

    </div>

    <div class="row">

      <div class="column">

        <div class="container">

          <a href="graph.php?location=castellon"><img src="img/playa\_castellon.jpg" alt="castellon" style="width:100%"></a>

          <a href="graph.php?location=castellon" class="centered">CASTELL&oacuteN</a>

        </div>

      </div>

      <div class="column">

        <div class="container">

          <a href="graph.php?location=valencia"><img src="img/playa\_valencia.jpg" alt="valencia" style="width:100%"></a>

          <a href="graph.php?location=valencia" class="centered">VALENCIA</a>

        </div>

      </div>

      <div class="column">

        <div class="container">

          <a href="graph.php?location=alicante"><img src="img/playa\_alicante.jpg" alt="alicante" style="width:100%"></a>

          <a href="graph.php?location=alicante" class="centered">ALICANTE</a>

        </div>

      </div>

    </div>

    <div class="section">

      <h3 class="section-2" id="top\_4">TOP 4</h3>

      <p>

        Puedes navegar por las diferentes playas de Valencia, las playas de Alicante y las playas de Castellón.

        Si lo prefieres, selecciona tus preferencias en el menú de la izquierda o

        busca por nombre de playa en el formulario que encontrarás justo por encima de dicho menú.

        También puedes ver cada una de las playas en el mapa de playas de la Comunidad Valenciana Además,

        podrás enviarnos las mejores fotos de tus playas de la Comunidad Valenciana preferidas.

        Para ello sólo tienes que navegar hasta la playa y subir tu foto.

      </p>

      <div>

          <?php

          $result = $db->get\_top\_beaches(4, 'ALL');

          ?>

        <div class="top-beach">

          <div class="row1">

            <div class="column1">

              <div class="container1">

                <img src="img/carousel\_beach\_1.jpg" alt="imagen top playa 1">

                <div class="title-top4"><?php echo $result[0]['name'] ?></div>

              </div>

            </div>

            <div class="column1">

              <div class="container1">

                <img src="img/carousel\_beach\_2.jpg" alt="imagen top playa 2">

                <div class="title-top4"><?php echo $result[1]['name'] ?></div>

              </div>

            </div>

            <div class="column1">

              <div class="container1">

                <img src="img/carousel\_beach\_3.jpg" alt="imagen top playa 3">

                <div class="title-top4"><?php echo $result[2]['name'] ?></div>

              </div>

            </div>

            <div class="column1">

              <div class="container1">

                <img src="img/carousel\_beach\_4.jpg" alt="imagen top playa 4">

                <div class="title-top4"><?php echo $result[3]['name'] ?></div>

              </div>

            </div>

          </div>

        </div>

      </div>

    </div>

    <div id="past\_comments">

      <?php include 'all\_past\_comments.php'; ?>

    </div>

    <?php include 'comments.php'; ?>

    <!-- Scroll top button -->

    <div class="top-button">

      <a href="#title"><img src="img/top\_button.png" alt="top button" class="scroll-up" height="100" width="100"></a>

    </div>

  </div>

  <?php include 'inc/footer.php'; ?>

</body>

</html>

* Archivo graph.php

<?php include 'inc/header.php' ?>

<link rel="stylesheet" href="css/sytle\_graphs.css">

</head>

<body>

  <!-- navBar -->

  <div class="navBar">

    <!-- navMenu -->

    <div class="navMenu">

      <a href="main.php">home</a>

      <a href="graph.php?location=<?php print\_r(h($\_GET["location"]));?>#playas">playas</a>

      <a href="graph.php?location=<?php print\_r(h($\_GET["location"]));?>#top\_4">top 4</a>

      <a href="graph.php?location=<?php print\_r(h($\_GET["location"]));?>#contacto">contacto</a>

      <div class="dot"></div>

    </div>

    <!-- Header -->

    <header id="title">

      <a href="main.php" id="header-title">SMART BEACH COMUNITAT VALENCIANA</a>

    </header>

  </div>

  <!-- Page Content  -->

  <div class="container">

    <img src="" alt="" class="header-img" id="header\_img" height="900" width="1875">

    <h2 class="centered" id="location"></h2>

  </div>

  <div class="content">

    <div class="section">

      <h3 class="section-1" id="playas">Playas</h3>

      <p>

        Puedes navegar por las diferentes playas de Valencia, las playas de Alicante y las playas de Castellón.

        Si lo prefieres, selecciona tus preferencias en el menú de la izquierda o

        busca por nombre de playa en el formulario que encontrarás justo por encima de dicho menú.

        También puedes ver cada una de las playas en el mapa de playas de la Comunidad Valenciana Además,

        podrás enviarnos las mejores fotos de tus playas de la Comunidad Valenciana preferidas.

        Para ello sólo tienes que navegar hasta la playa y subir tu foto.

      </p>

      <?php

        $location = $\_GET["location"];

        $result = $db->get\_all\_beaches("$location");

      ?>

      <div class="custom-select" id="beach-container">

        <label for="beach">Seleccione una playa:</label>

        <select name="beach" id="beach\_select" class="beach-select" onchange="update\_beach(this.value)">

        <option value="standar" class="beach-option" selected><?php if(isset($\_GET['beach'])){echo $\_GET['beach'];}else{echo ' ';}?></option>

        <?php

          for($i=0;$i<count($result);$i++){

            echo '<option class="beach-option" value='.$result[$i]['name'].'>'.$result[$i]['name'].'</option>';

          };

        ?>

        </select>

      </div>

      <div id="beach-title" class="beach-title"></div>

    </div>

    <div class="data-section-1">

      <div class="actual-data">

        <ul class="data-row">

          <li class="data-col-temp"><div id="actual\_temp"></div></li>

          <li class="data-col-icon" id="data\_col\_icon\_temp"><img src="img/icon\_temperature.png" alt="icon temperature" height="50px" id="data-type" value="0" onclick="type\_data\_icon(this)"></li>

          <li class="data-col-hum"><div id="actual\_hum"></div></li>

          <li class="data-col-icon" id="data\_col\_icon\_hum"><img src="img/icon\_drop.png" alt="icon drop" height="50px" id="data-type" value="1" onclick="type\_data\_icon(this)"></li>

          <li class="data-col-uv"><div id="actual\_uv"></div></li>

          <li class="data-col-icon" id="data\_col\_icon\_uv"><img src="img/icon\_uv\_index.png" alt="icon uv index" height="60px" id="data-type" value="2" onclick="type\_data\_icon(this)"></li>

          <li class="data-col-pres"><div id="actual\_pres"></div></li>

          <li class="data-col-icon" id="data\_col\_icon\_pres"><img src="img/icon\_atm\_pres.png" alt="icon atmospheric pressure" height="70px" id="data-type" value="3" onclick="type\_data\_icon(this)"></li>

          <li class="data-col"><img src="img/plus\_button.png" alt="plus button" class="plus-button"  id="flip" height="30px"></li>

        </ul>

      </div>

      <div id="flip-container">

        <div class="graph-container" id="graph-container"></div>

        <div class="hours-container" id="hours-container">

          <li class="hours-col-history"><img src="img/icon\_history.png" alt="history icon" height="50px"></li>

          <ul class="hours-row">

            <li class="hours-col" id="hours-col1"><img src="img/icon\_one\_hour.png" alt="one\_hour" height="20px" id="hours" value="1" onclick="hours\_select\_icon(this)"></li>

            <li class="hours-col" id="hours-col3"><img src="img/icon\_three\_hour.png" alt="three\_hours" height="20px" id="hours" value="3" onclick="hours\_select\_icon(this)"></li>

            <li class="hours-col" id="hours-col5"><img src="img/icon\_five\_hour.png" alt="five hours" height="20px" id="hours" value="5" onclick="hours\_select\_icon(this)"></li>

            <li class="hours-col" id="hours-col12"><img src="img/icon\_twelve\_hour.png" alt="twelve hours" height="20px" id="hours" value="12" onclick="hours\_select\_icon(this)"></li>

            <li class="hours-col" id="hours-col24"><img src="img/icon\_one\_day.png" alt="one day" height="20px" id="hours" value="24" onclick="hours\_select\_icon(this)"></li>

          </ul>

        </div>

      </div>

    </div>

    <div class="section">

      <h3 class="section-2" id="top\_4">TOP 4</h3>

      <p>

        Puedes navegar por las diferentes playas de Valencia, las playas de Alicante y las playas de Castellón.

        Si lo prefieres, selecciona tus preferencias en el menú de la izquierda o

        busca por nombre de playa en el formulario que encontrarás justo por encima de dicho menú.

        También puedes ver cada una de las playas en el mapa de playas de la Comunidad Valenciana Además,

        podrás enviarnos las mejores fotos de tus playas de la Comunidad Valenciana preferidas.

        Para ello sólo tienes que navegar hasta la playa y subir tu foto.

      </p>

      <div>

          <?php

          $location = $\_GET["location"];

          $result = $db->get\_top\_beaches(4, $location);

          ?>

        <div class="top-beach">

          <div class="row">

            <div class="column">

              <div class="container1">

                <img src="img/carousel\_beach\_1.jpg" alt="imagen top playa 1">

                <div class="title-top4"><?php echo $result[0]['name'] ?></div>

              </div>

            </div>

            <div class="column">

              <div class="container1">

                <img src="img/carousel\_beach\_2.jpg" alt="imagen top playa 2">

                <div class="title-top4"><?php echo $result[1]['name'] ?></div>

              </div>

            </div>

            <div class="column">

              <div class="container1">

                <img src="img/carousel\_beach\_3.jpg" alt="imagen top playa 3">

                <div class="title-top4"><?php echo $result[2]['name'] ?></div>

              </div>

            </div>

            <div class="column">

              <div class="container1">

                <img src="img/carousel\_beach\_4.jpg" alt="imagen top playa 4">

                <div class="title-top4"><?php echo $result[3]['name'] ?></div>

              </div>

            </div>

          </div>

        </div>

      </div>

    </div>

    <?php include 'past\_comments.php'; ?>

    <?php include 'comments.php'; ?>

  </div>

  <div class="top-button">

    <a href="#title"><img src="img/top\_button.png" alt="top button" class="scroll-up" height="80"></a>

  </div>

  <?php include 'inc/footer.php'; ?>

</body>

<script>

<?php

  function h($s) {

    echo htmlspecialchars($s, ENT\_QUOTES, 'UTF-8');

  }

?>

let active\_temp = document.getElementById("data\_col\_icon\_temp");

let active\_hum = document.getElementById("data\_col\_icon\_hum");

let active\_uv = document.getElementById("data\_col\_icon\_uv");

let active\_pres = document.getElementById("data\_col\_icon\_pres");

$(document).ready(function(){

  $("#flip").click(function(){

    $("#flip-container").slideToggle("slow");

    active\_temp.classList.add("active-btn");

    active\_hum.classList.toggle("active-btn");

    active\_uv.classList.toggle("active-btn");

    active\_pres.classList.toggle("active-btn");

    if($("#flip").attr("src") == "img/plus\_button.png"){

      $("#flip").attr("src", "img/minus\_button.png")

    }

    else($("#flip").attr("src", "img/plus\_button.png"))

  })

})

function getParameterByName(name) {

    name = name.replace(/[\[]/, "\\[").replace(/[\]]/, "\\]");

    var regex = new RegExp("[\\?&]" + name + "=([^&#]\*)"),

    results = regex.exec(location.search);

    return results === null ? "" : decodeURIComponent(results[1].replace(/\+/g, " "));

}

var province = getParameterByName('location');

document.getElementById("location").innerHTML = province.toUpperCase();

var beach\_title = getParameterByName('beach');

document.getElementById("beach-title").innerHTML = beach\_title;

var src = "";

var alt = "";

switch (getParameterByName('location')){

  case "castellon":

    src = "img/playa\_castellon.jpg";

    alt = "playa castellon";

  break;

  case "valencia":

    src = "img/playa\_valencia.jpg";

    alt = "playa valencia";

  break;

  case "alicante":

    src = "img/playa\_alicante.jpg";

    alt = "playa alicante";

  break;

}

var img = document.getElementById('header\_img');

img.src = src;

img.alt = alt;

parser = new Parser\_v1();

cloud = new Thingspeak("1831954");

function update\_beach(beach){

  // var e = document.getElementById("beach\_select");

  // var channel\_id = e.value;

  // parser = new Parser\_v1();

  // cloud = new Thingspeak(channel\_id);

  // console.log(channel\_id);

  <?php

    if(isset($\_GET['beach'])){

      $beach=$\_GET['beach'];

    }else{

      $beach= '';

    }

    $db->set\_new\_visit($beach);

    $location = $\_GET["location"];

    $db->get\_top\_beaches(4, $location);

  ?>

  location.href='graph.php?location=<?php print\_r(h($\_GET["location"]));?>&beach=' + beach +'#playas';

}

// ----------------------------------------------------------------------------------------------------------------------

cloud.get\_latest\_field(4).then(callback\_ok\_last).catch(callback\_err);

setInterval(latest\_wrapper,120000);

function latest\_wrapper(){

  cloud.get\_latest\_field(4).then(callback\_ok\_last).catch(callback\_err);

}

cloud.get\_field\_by\_time(4,60).then(callback\_ok).catch(callback\_err);

let selected;

let upper\_limit = 60;

function type\_data\_icon(element){

  selected = element.getAttribute("value")

  cloud.get\_field\_by\_time(4,upper\_limit).then(callback\_ok).catch(callback\_err);

}

var active\_1 = document.getElementById("hours-col1");

active\_1.classList.add("active-btn");

function hours\_select\_icon(element){

  var hours = element.getAttribute("value");

  var active\_1 = document.getElementById("hours-col1");

  var active\_3 = document.getElementById("hours-col3");

  var active\_5 = document.getElementById("hours-col5");

  var active\_12 = document.getElementById("hours-col12");

  var active\_24 = document.getElementById("hours-col24");

  switch(hours){

    case "1":

      upper\_limit = 60;

      active\_1.classList.add("active-btn");

      active\_3.classList.remove("active-btn");

      active\_5.classList.remove("active-btn");

      active\_12.classList.remove("active-btn");

      active\_24.classList.remove("active-btn");

    break;

    case "3":

      upper\_limit = 180;

      active\_3.classList.add("active-btn");

      active\_1.classList.remove("active-btn");

      active\_5.classList.remove("active-btn");

      active\_12.classList.remove("active-btn");

      active\_24.classList.remove("active-btn");

    break;

    case "5":

      upper\_limit = 300;

      active\_5.classList.add("active-btn");

      active\_1.classList.remove("active-btn");

      active\_3.classList.remove("active-btn");

      active\_12.classList.remove("active-btn");

      active\_24.classList.remove("active-btn");

    break;

    case "12":

      upper\_limit = 720;

      active\_12.classList.add("active-btn");

      active\_1.classList.remove("active-btn");

      active\_3.classList.remove("active-btn");

      active\_5.classList.remove("active-btn");

      active\_24.classList.remove("active-btn");

    break;

    case "24":

      upper\_limit = 1440;

      active\_24.classList.add("active-btn");

      active\_1.classList.remove("active-btn");

      active\_3.classList.remove("active-btn");

      active\_5.classList.remove("active-btn");

      active\_12.classList.remove("active-btn");

    break;

  }

  cloud.get\_field\_by\_time(4,upper\_limit).then(callback\_ok).catch(callback\_err);

}

function callback\_ok(data){

  const frames = parser.parser(data.feeds);

  const temp\_graph = frames.map(x=>x.temperature);

  const hum\_graph = frames.map(x=>x.humidity);

  const pres\_graph = frames.map(x=>x.pressure);

  const uv\_graph = frames.map(x=>x.uv\_index);

  const batt\_graph = frames.map(x=>x.battery);

  const parsed\_date = frames.map(x=>x.time);

  var graph\_label;

  switch(selected){

    case "0":

      graph\_label = "Temperatura (ºC)";

      canvas(temp\_graph, parsed\_date, graph\_label);

      active\_temp.classList.add("active-btn");

      active\_hum.classList.remove("active-btn");

      active\_uv.classList.remove("active-btn");

      active\_pres.classList.remove("active-btn");

    break;

    case "1":

      graph\_label = "Humedad relativa (%)";

      canvas(hum\_graph, parsed\_date, graph\_label);

      active\_hum.classList.add("active-btn");

      active\_temp.classList.remove("active-btn");

      active\_uv.classList.remove("active-btn");

      active\_pres.classList.remove("active-btn");

    break;

    case "2":

      graph\_label = "Índice UV";

      canvas(uv\_graph, parsed\_date, graph\_label);

      active\_uv.classList.add("active-btn");

      active\_hum.classList.remove("active-btn");

      active\_temp.classList.remove("active-btn");

      active\_pres.classList.remove("active-btn");

    break;

    case "3":

      graph\_label = "Presión atmosférica (mB)"

      canvas(pres\_graph, parsed\_date, graph\_label);

      active\_pres.classList.add("active-btn");

      active\_hum.classList.remove("active-btn");

      active\_temp.classList.remove("active-btn");

      active\_uv.classList.remove("active-btn");

    break;

    default:

      graph\_label = "Temperatura (ºC)";

      canvas(temp\_graph, parsed\_date, graph\_label);

      active\_temp.classList.add("active-btn");

      active\_hum.classList.add("active-btn");

      active\_uv.classList.add("active-btn");

      active\_pres.classList.add("active-btn");

    break;

  }

}

function callback\_ok\_last(data){

  const frames = parser\_last(data);

  const temp\_last = frames[0];

  const hum\_last = frames[1];

  const uv\_last = frames[2];

  const pres\_last = frames[3];

  console.log("esto es actual\_temp: ", temp\_last)

  console.log("esto es actual\_hum: ", hum\_last)

  console.log("esto es actual\_uv: ", uv\_last)

  console.log("esto es actual\_pres: ", pres\_last)

  document.getElementById("actual\_temp").innerHTML = temp\_last;

  document.getElementById("actual\_hum").innerHTML = hum\_last;

  document.getElementById("actual\_uv").innerHTML = uv\_last;

  document.getElementById("actual\_pres").innerHTML = pres\_last;

}

function callback\_err(data){

  console.log("error");

  console.log(data)

}

</script>

</html>

* Archivo thingspeak.php

class Thingspeak{

  url\_base = "https://api.thingspeak.com/channels/";

  feeds = "feeds";

  fields = "fields";

  last\_endpoint = "none";

  last\_format = "none";

  channel\_id;

  field\_name;

  format;

  constructor(channel\_id, format = "json") {

    this.channel\_id = channel\_id;

    this.format = format;

  }

  /\*!

  \* Method: build\_url

  \* Return a string representing an URL by concatenating different fields:

  \*  - endpoint: web to ask for, for instance, feeds

  \*  - params: url parameters, for instance, "start" and "end". It must follow the url name convention (example: "results=120&average=60")

  \*/

  //"params != "" ? "?" + params : """ ------> if params is different of "" then ?params if not ""//

  #build\_url(endpoint, params = ""){

    const p = params != "" ? "?" + params : "";

    const url = this.url\_base + this.channel\_id + "/" + endpoint + "." + this.format + p;

    this.last\_endpoint = endpoint;

    this.last\_format = this.format;

    console.log(url);

    return url;

  }

  /\*!

  \* Method: connect

  \* Perform the FETCH request.

  \* Return data

  \*/

  #connect(url){

    switch(this.last\_format){

      case 'json':

        return fetch(url).then(blob=>blob.json());

    }

  }

  //OK

  get\_field(field\_name){

    //example: https://api.thingspeak.com/channels/<channel\_id>/fields/<field\_id>.<format>

    const url = this.#build\_url(this.fields + "/" + field\_name);

    const promise = this.#connect(url, callback\_ok, callback\_err);

    return promise;

  }

  //OK

  get\_latest\_frame(){

    //example: https://api.thingspeak.com/channels/<channel\_id>/feeds/last.<format>

    const url = this.#build\_url(this.feeds + "/last");

    const promise = this.#connect(url, callback\_ok, callback\_err);

    return promise;

  }

  //OK

  get\_latest\_field(field\_name){

    //example: https://api.thingspeak.com/channels/<channel\_id>/fields/<field\_id>/last.<format>

    const url = this.#build\_url(this.fields + "/" + field\_name + "/last");

    const promise = this.#connect(url, callback\_ok\_last, callback\_err)

    return promise;

  }

  //DATE FORMAT START "AAAA-MM-DD" END "AAAA-MM-DD"

  //OK

  get\_frame\_by\_date(start, end){

    //example: https://api.thingspeak.com/channels/<channel\_id>/feeds.<format>?start=...&end=...

    const parameters = "start=" + start + "&end=" + end;

    const url = this.#build\_url(this.feeds, parameters);

    const promise = this.#connect(url, callback\_ok, callback\_err);

    return promise;

  }

  //OK

  get\_field\_by\_date(field\_name, start, end){

    //example: https://api.thingspeak.com/channels/<channel\_id>/fields/<field\_id>.<format>?start...&end...

    const parameters = "start=" + start + "&end=" + end;

    const url = this.#build\_url(this.fields + "/" + field\_name, parameters);

    const promise = this.#connect(url, callback\_ok, callback\_err);

    return promise;

  }

  get\_field\_by\_time(field\_name,minutes){

    //example: https://api.thingspeak.com/channels/<channel\_id>/fields/<field\_id>.<format>

    const parameters = "minutes=" + minutes

    const url = this.#build\_url(this.fields + "/" + field\_name, parameters);

    const promise = this.#connect(url, callback\_ok, callback\_err);

    return promise;

  }

}

* Archivo parser.js

/\*!

\* Data classes

\*/

class IFrame{

  temperature = NaN;

  humidity = NaN;

  pressure = NaN;

  uv\_index = NaN;

  battery = NaN;

  time = NaN;

  constructor() {}

}

class Frame\_v1 extends IFrame{

  temperature = NaN;

  humidity = NaN;

  pressure = NaN;

  uv\_index = NaN;

  battery = NaN

  time = NaN;

  // node\_id = NaN;

  constructor() {super();}

}

/\*!

 \* Parser classes.

 \* For new parsers, extend from IParser

 \*/

class IParser{

  parser(str) {

    throw new Error("Abstract Method has no implementation");

  }

}

class Parser\_v1 extends IParser{

  parser(str){

    var frames = [];

    for(var i=0; i<str.length; i++){

      var DatosJSON = str[i].field4;

      var DateJSON = str[i].created\_at;

      this.frame = new Frame\_v1()

      this.frame.temperature = parseInt(DatosJSON.substring(0, 4))/100;

      this.frame.humidity = parseInt(DatosJSON.substring(4, 6));

      this.frame.pressure = parseInt(DatosJSON.substring(6, 10)); //en milibares o hectopascales 1atm=1013.25mb

      this.frame.uv\_index = parseInt(DatosJSON.substring(10,12));

      this.frame.battery = parseInt(DatosJSON.substring(22, 24));

      this.frame.time = ParserTime(DateJSON);

      frames.push(this.frame);

    }

    //console.log("esto es frames array filter temperature: ", frames);

    return frames;

  }

}

function ParserTime(DateJSON){

  const raw\_date = new Date(DateJSON);

  const parsed\_date = raw\_date.toLocaleString('es-ES', {timeStyle: 'short', hour12: false, timeZone: 'Europe/Madrid' });

  return parsed\_date;

}

function parser\_last(str){

  var DatosJSON = str.field4;

  var DateJSON = str.created\_at;

  var frames = [];

  const temperature = parseInt(DatosJSON.substring(0, 4))/100;

  const humidity = parseInt(DatosJSON.substring(4, 6));

  const pressure = parseInt(DatosJSON.substring(6, 10)); //en milibares o hectopascales 1atm=1013.25mb

  const uv\_index = parseInt(DatosJSON.substring(10,12));

  const battery = parseInt(DatosJSON.substring(22, 24));

  const time = ParserTime(DateJSON);

  frames.push(temperature);

  frames.push(humidity);

  frames.push(uv\_index);

  frames.push(pressure);

  return frames;

}

//node.js export. Needed for unit testing

module.exports = {IFrame, Frame\_v1, IParser, Parser\_v1};

* Archivo graph.js

/\*!

\* Graph classes

\* Plot the given data into a graph canvas

\*/

function canvas(data, parsed\_date, graph\_label){

  Highcharts.chart('graph-container', {

    chart: {

        type: 'line'

    },

    title: {

        text: ''

    },

    xAxis: {

        categories: parsed\_date

    },

    plotOptions: {

        line: {

            dataLabels: {

                enabled: false

            },

            enableMouseTracking: true

        },

        series: {

            events: {

                hide: function(a) {

                    this.yAxis.axisTitle.hide();

                },

                show: function() {

                this.yAxis.axisTitle.show();

              }

            }

        },

    },

    yAxis: [{

        title:{

            text: 'Valores'

        }

    }],

    series: [{

        yAxis: 0,

        name: graph\_label,

        data: data

    }]

  });

}