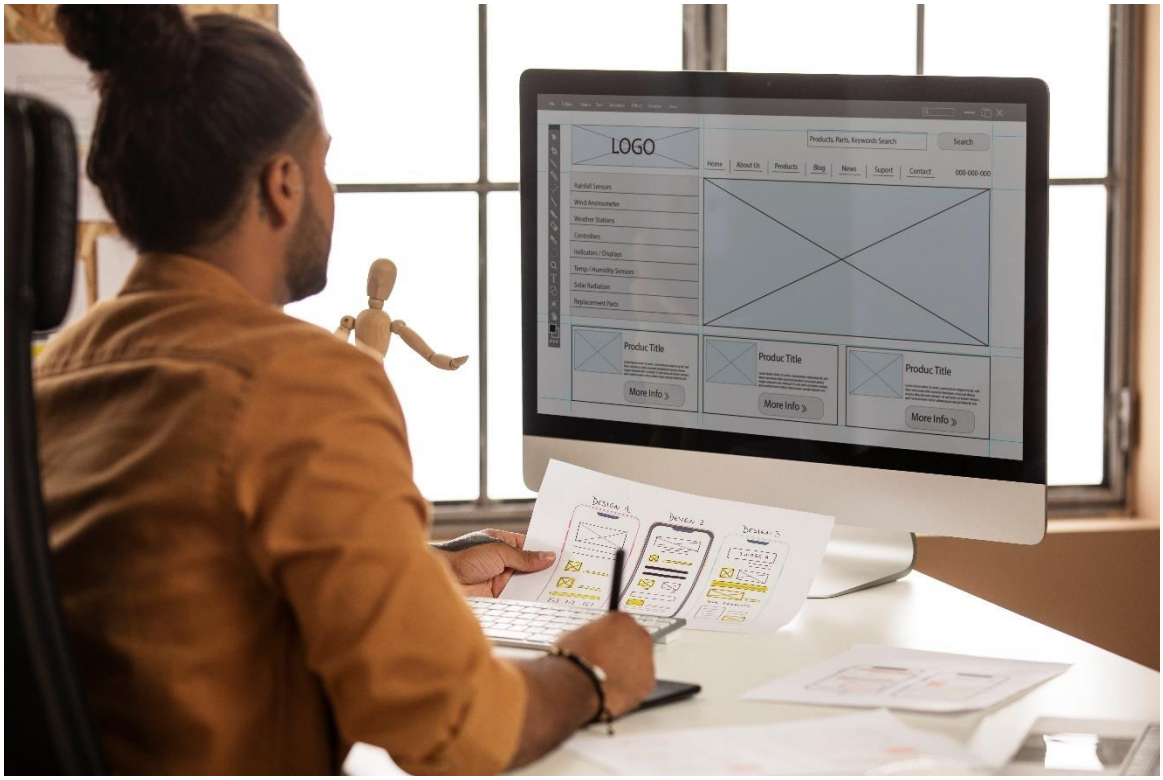
Desarrollo y despliegue de aplicaciones web



El desarrollo de aplicaciones web ha evolucionado considerablemente en los últimos años. Antes, crear una aplicación web requería conocimientos profundos de varios lenguajes de programación y mucho tiempo de desarrollo. Hoy en día, existen modelos y herramientas que facilitan mucho el proceso, permitiendo crear aplicaciones más robustas, escalables y fáciles de mantener. En esta sección, se va a explicar cómo funciona el desarrollo y el despliegue de aplicaciones web, con un enfoque en el modelo vista-controlador (MVC), así como una comparativa de las herramientas más usadas en el desarrollo web.

1. Modelos básicos de desarrollo de aplicaciones web. El modelo vista-controlador (MVC)

Al desarrollar una aplicación web, es importante tener un modelo de desarrollo bien estructurado que facilite la organización del código, la separación de responsabilidades y el mantenimiento a largo plazo. Uno de los modelos más usados y efectivos para esto es el modelo vista-controlador (MVC). Este modelo separa la lógica de negocio, la presentación y la gestión de las interacciones del usuario, lo que hace que el código sea más fácil de gestionar y escalar.

¿Cómo funciona el modelo MVC?

El modelo vista-controlador (MVC) divide una aplicación en tres componentes principales:

- Modelo (Model): Es la parte de la aplicación que maneja la lógica de los datos. Aquí es donde se gestiona cómo se almacenan, procesan y recuperan los datos. En un contexto de aplicación web, el modelo se encarga de interactuar con la base de datos, gestionar la información que necesita ser mostrada o actualizada, y asegurarse de que los datos fluyan correctamente entre la aplicación y el usuario. Por ejemplo, en una aplicación de tienda online, el modelo podría manejar la lógica relacionada con los productos, como calcular el precio total de un carrito de compras o actualizar el inventario cuando se realiza una compra.
- Vista (View): La vista es lo que el usuario ve y con lo que interactúa. En términos más sencillos, es la interfaz de usuario (UI). Las vistas se encargan de mostrar los datos que proporciona el modelo, pero no tienen acceso a la lógica de negocio ni realizan cálculos. Se limitan a mostrar la información en un formato adecuado. Por ejemplo, la página del carrito de compras en una tienda online es una vista. Muestra los productos que el usuario ha seleccionado, los precios y el total a pagar.
- Controlador (Controller): El controlador es el intermediario entre el modelo y la vista. Recibe las entradas del usuario (clics, formularios, etc.), las procesa y decide qué hacer con esos datos. En base a las acciones del usuario, el controlador puede actualizar el modelo, cambiar la vista o ambos. Por ejemplo. si el usuario añade un producto al carrito, el controlador recibirá esa acción, actualizará el modelo (para reflejar el nuevo producto en el carrito) y luego enviará la información a la vista para que se actualice y muestre los cambios al usuario.

Anotación

Este modelo de tres partes permite que el código esté bien organizado. Por ejemplo, si en el futuro quieres cambiar la forma en que se presenta la información al usuario, solo tendrás que modificar la vista, sin afectar el modelo ni el controlador.

¿Por qué es útil el modelo MVC?

El modelo MVC es popular porque facilita el mantenimiento y escalabilidad de las aplicaciones web. Al dividir la aplicación en partes, es más fácil hacer cambios y mejoras sin afectar otras áreas del código. También es más fácil trabajar en equipo: un desarrollador puede encargarse de la lógica de negocio (modelo), mientras que otro se enfoca en la interfaz (vista) y un tercero en la lógica de las interacciones del usuario (controlador).

2. Herramientas de desarrollo web de uso común.

El desarrollo web moderno se apoya en diversas herramientas que facilitan la creación y despliegue de aplicaciones web. Estas herramientas permiten a los desarrolladores escribir, probar y depurar código de manera más eficiente, además de gestionar otros aspectos como el control de versiones, la colaboración en equipo y la automatización de tareas.

2.1. Características.

Vamos a revisar las principales herramientas de desarrollo y sus características, para entender cómo ayudan en el proceso de creación de aplicaciones web:

- Editores de código: Los editores de código son la herramienta principal que usan los desarrolladores para escribir el código fuente de una aplicación web. Los más populares incluyen Visual Studio Code, Sublime Text y Atom. Estos editores ofrecen características como la autocompletación, resaltado de sintaxis, integración con sistemas de control de versiones como Git y una gran cantidad de extensiones que ayudan a personalizar el entorno de desarrollo según las necesidades del proyecto. Si estás trabajando en una aplicación web en JavaScript, Visual Studio Code puede sugerir automáticamente las funciones de JavaScript que necesitas mientras escribes, además de marcar los errores de sintaxis en tiempo real.
- Frameworks web: Un framework es una estructura o conjunto de herramientas que facilita el desarrollo de aplicaciones al proporcionar un conjunto de reglas y convenciones. En lugar de escribir cada parte de una aplicación desde cero, los desarrolladores pueden usar un framework que ya tiene muchas funciones integradas. Algunos frameworks populares son:
    - Django (para Python): Se centra en la rapidez y la simplicidad. Es ideal para aplicaciones que necesitan desarrollarse rápidamente, sin comprometer la calidad.
    - Ruby on Rails (para Ruby): Un framework que promueve la simplicidad y la convención sobre la configuración. Es muy utilizado en startups debido a su rapidez de desarrollo.
    - Express.js (para Node.js): Utilizado principalmente en aplicaciones que usan JavaScript tanto en el frontend como en el backend.

Anotación

Si quieres desarrollar un blog desde cero, usar un framework como Django o Ruby on Rails te permitirá hacerlo mucho más rápido, ya que estas herramientas ya tienen módulos integrados para la autenticación de usuarios, la gestión de bases de datos y el enrutamiento de las páginas.

- Sistemas de control de versiones: Estos sistemas permiten llevar un registro de todos los cambios que se hacen en el código a lo largo del tiempo, facilitando la colaboración entre equipos y la recuperación de versiones anteriores. Git es el sistema de control de versiones más utilizado, y herramientas como GitHub o GitLab permiten almacenar el código en línea y trabajar de forma colaborativa.

- Automatización de tareas: Herramientas como Gulp, Grunt o Webpack permiten automatizar tareas comunes del desarrollo web, como la compilación de archivos CSS, la minificación de código JavaScript o la optimización de imágenes. Esto hace que el desarrollo sea más rápido y menos propenso a errores. Si estás trabajando en una aplicación web que usa muchos archivos JavaScript, Webpack puede combinarlos en un solo archivo más pequeño, lo que mejora el rendimiento de la aplicación cuando se carga en el navegador.

2.2. Comparativa.

A continuación, se presenta una comparativa de algunas de las herramientas más utilizadas en el desarrollo web, teniendo en cuenta factores como facilidad de uso, popularidad y flexibilidad:

Herramienta	Tipo	Características principales	Popularidad	Facilidad de uso
Visual Studio Code	Editor de código	Autocompletación, extensiones, integración con Git, multiplataforma	Muy alta	Alta
Sublime Text	Editor de código	Ligero, rápido, personalización avanzada con plugins	Alta	Media
Django	Framework (Python)	Basado en el modelo MVC, muchas características integradas (autenticación, ORM, administración)	Alta	Media-alta
Ruby on Rails	Framework (Ruby)	Convención sobre configuración, velocidad de desarrollo rápida	Media-alta	Media
Express.js	Framework (Node.js)	Flexibilidad, rápido y liviano, perfecto para APIs y aplicaciones de una sola página	Alta	Media
Git	Control de versiones	Control de versiones distribuido, muy usado en proyectos colaborativos	Muy alta	Media
Webpack	Automatización	Agrupación de módulos, optimización de archivos, ideal para proyectos grandes	Alta	Media-baja

Saber más...

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es gratuito, de código abierto y compatible con Windows, macOS y Linux. Ofrece una amplia gama de extensiones que mejoran su funcionalidad.

A continuación, se presentan los pasos para instalar este editor y algunas de sus extensiones:

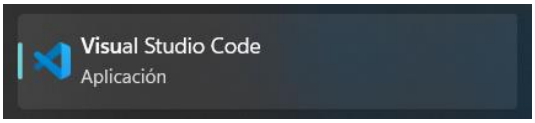
1. Ve al sitio oficial de Visual Studio Code en tu navegador web.

2. Haz clic en el botón de descarga para tu sistema operativo (Windows, MacOS, Linux):

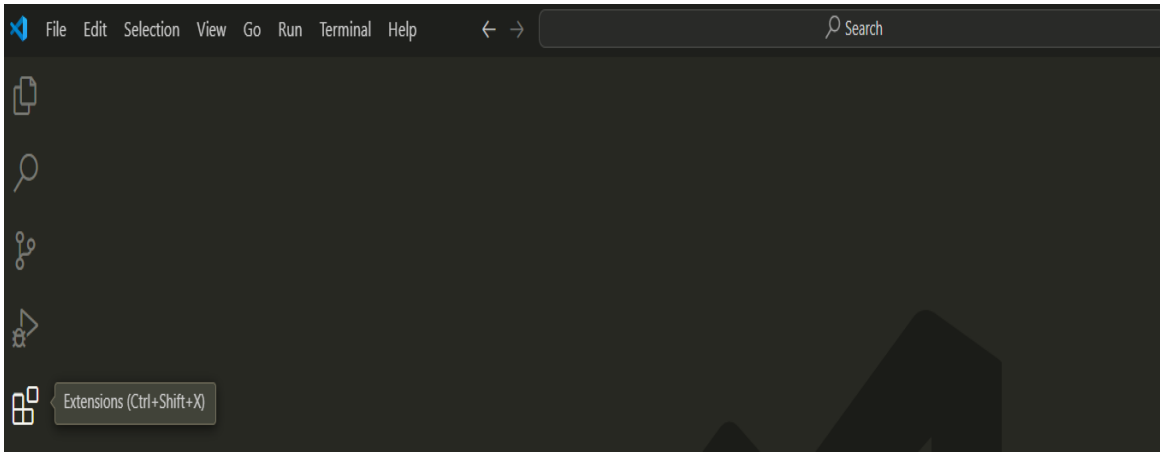
3. Una vez descargado el archivo, ábrelo para iniciar el proceso de instalación.

4. Sigue las instrucciones del instalador, que incluirán aceptar los términos de servicio y elegir la ubicación de instalación.

5. Una vez completada la instalación, puedes abrir Visual Studio Code:

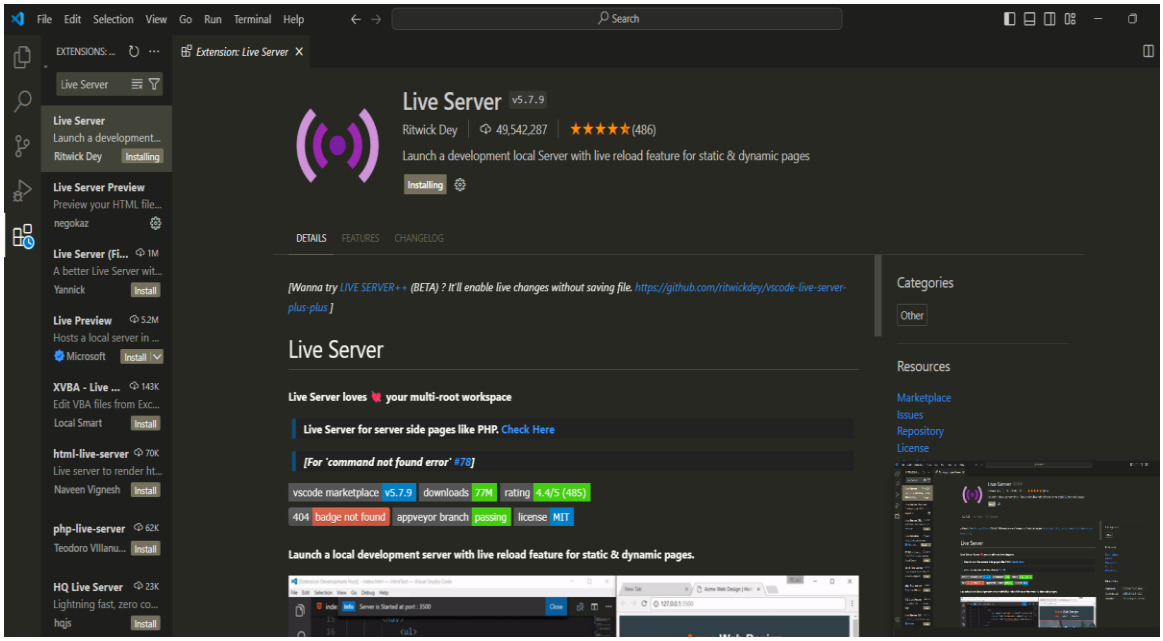


6. Haz clic en el icono de extensiones en la barra lateral izquierda (o presiona Ctrl+Shift+X). :



7. En la barra de búsqueda en la parte superior, escribe el nombre de la extensión que deseas instalar. Por ejemplo, podrías buscar “Live Server”, "HTML Boilerplate" o "Prettier - Code formatter":

8. En los resultados de la búsqueda, haz clic en el botón “Instalar” para la extensión que deseas, por ejemplo “Live Server”:



9. Una vez instaladas las extensiones, reinicia Visual Studio Code para que los cambios surtan efecto.

3. Políticas de desarrollo y pruebas de aplicaciones web.

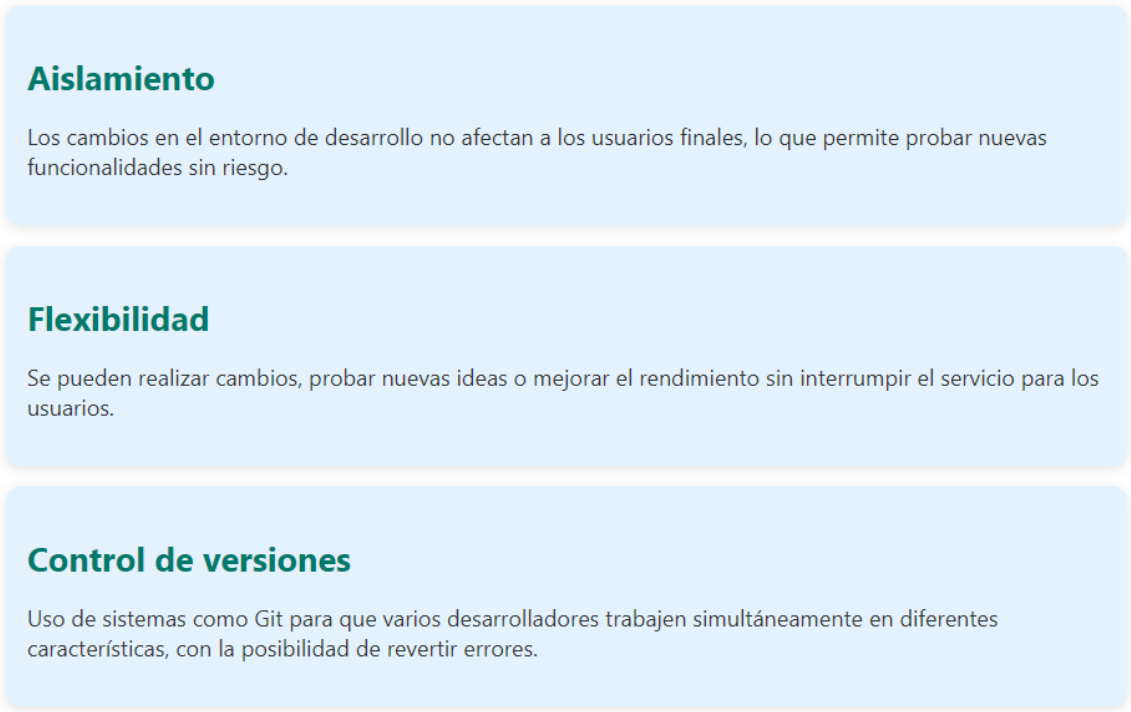
El desarrollo de una aplicación web no se trata solo de escribir código y ponerlo en línea. Es un proceso más estructurado, que involucra diferentes entornos donde se desarrollan, prueban y finalmente se despliegan las aplicaciones. Este enfoque no solo facilita el trabajo en equipo, sino que también minimiza los errores y asegura que todo funcione correctamente antes de que los usuarios finales accedan a la aplicación.

Estos entornos suelen dividirse en tres: el entorno de desarrollo, el entorno de pre-producción o pruebas y el entorno de producción. Cada uno cumple una función específica, y juntos forman una cadena que permite mantener un flujo de trabajo eficiente y seguro.

3.1. Entorno de desarrollo.

El entorno de desarrollo es el lugar donde los desarrolladores escriben y prueban el código por primera vez. Es como un laboratorio, donde se puede experimentar sin preocuparse por causar problemas a los usuarios finales. Aquí es donde se crea la lógica de la aplicación, se diseñan las interfaces, se configuran las bases de datos y se prueban nuevas funcionalidades.

Características del entorno de desarrollo:



- Aislamiento: Todo lo que sucede en este entorno no afecta a los usuarios finales. Por lo general, el entorno de desarrollo está en los ordenadores locales de los desarrolladores o en servidores dedicados, y permite realizar cambios sin riesgo de afectar a la versión pública de la aplicación. Si se está desarrollando una nueva funcionalidad para añadir un sistema de reseñas a una tienda en línea, esta se probaría primero en el entorno de desarrollo. Si hay errores o problemas, se pueden corregir sin que los clientes de la tienda lo noten.

- Flexibilidad: Los desarrolladores pueden hacer cambios en cualquier parte del código, probar nuevas ideas, refactorizar funciones o mejorar el rendimiento sin tener que preocuparse por las interrupciones en el servicio. Un equipo de desarrollo que trabaja en una aplicación de red social puede probar varias formas de implementar el feed de noticias sin preocuparse por dañar la experiencia del usuario.
- Control de versiones: En este entorno es común usar sistemas de control de versiones como Git. Esto permite a los desarrolladores trabajar en diferentes características al mismo tiempo, realizar cambios y revertir errores sin perder el trabajo avanzado. Si dos desarrolladores trabajan en diferentes módulos de una aplicación (uno en el perfil de usuario y otro en el sistema de notificaciones), cada uno puede trabajar en su propia versión del proyecto y fusionar los cambios cuando todo esté listo.

Actividad 13

Indica si las siguientes afirmaciones son Verdadero o Falso:

___
En un entorno de desarrollo, los errores y problemas se pueden corregir sin que afecten a los usuarios finales de la aplicación.

___
El entorno de desarrollo es el mismo que el entorno en el que los usuarios interactúan con la versión pública de la aplicación.

___
Los desarrolladores tienen flexibilidad en el entorno de desarrollo para hacer cambios sin preocuparse por interrumpir el servicio para los usuarios.

___
En un entorno de desarrollo no se utiliza control de versiones, ya que solo un desarrollador puede trabajar en el código a la vez.

___
El sistema de control de versiones como Git permite que varios desarrolladores trabajen en diferentes partes de una aplicación al mismo tiempo sin perder su progreso.

3.2. Entorno de pre-producción o pruebas.

Una vez que el código ha sido desarrollado y parece funcionar correctamente en el entorno de desarrollo, se traslada al entorno de pre-producción o pruebas. Este entorno simula las condiciones de producción, es decir, cómo funcionará la aplicación en el mundo real, pero sin estar disponible para los usuarios finales.

¿Por qué es importante este entorno? Porque muchas veces lo que funciona bien en el entorno de desarrollo no necesariamente funcionará de la misma manera en un entorno con más restricciones y condiciones reales, como la carga de usuarios o la integración con otros servicios.

Características del entorno de pre-producción:

- Simulación del entorno real: El entorno de pre-producción debe ser lo más parecido posible al entorno de producción, tanto en configuración del servidor, como en la base de datos, para que las pruebas sean realistas. Si hay diferencias entre ambos entornos, pueden surgir problemas en producción que no se detectaron durante las pruebas. Si la aplicación está destinada a manejar 10.000 usuarios simultáneos, el entorno de pruebas debe estar configurado para probar cómo se comportará bajo esa carga. De lo contrario, podrías descubrir problemas solo cuando ya esté en manos de los usuarios.
- Pruebas exhaustivas: En este entorno se realizan pruebas funcionales (para verificar que la aplicación hace lo que debe hacer), pruebas de estrés (para ver cómo se comporta bajo condiciones extremas) y pruebas de integración (para asegurarse de que todas las partes de la aplicación funcionan correctamente juntas). Un equipo que desarrolla una aplicación bancaria podría realizar pruebas de integración para asegurarse de que el sistema de pagos, el historial de transacciones y las notificaciones por correo electrónico funcionen sin problemas.
- Automatización de pruebas: En muchos casos, se usan herramientas de pruebas automatizadas que permiten ejecutar tests sobre el código para verificar que todas las funciones sigan funcionando después de cada cambio. Esto reduce el tiempo de prueba y permite detectar errores rápidamente. Si una tienda online ha desarrollado una nueva

función de búsqueda avanzada, un sistema de pruebas automatizado puede revisar cada variación posible de las búsquedas para asegurarse de que no se presentan fallos al usuario.

- Revisión por equipos: En este entorno, los equipos de QA (Control de calidad), desarrolladores y otros actores importantes revisan los resultados de las pruebas y deciden si el código está listo para ser enviado a producción. Si se detecta un error durante una prueba, el equipo de desarrollo regresa al entorno de desarrollo para corregirlo y vuelve a probarlo hasta que todo funcione correctamente.

3.3. Entorno de producción.

El entorno de producción es la fase final del desarrollo. Es donde la aplicación se despliega para ser utilizada por los usuarios finales. Una vez que el código ha pasado todas las pruebas en pre-producción, se mueve a este entorno, que es donde se espera que todo funcione correctamente sin interrupciones.

Características del entorno de producción:

- Disponibilidad: La aplicación debe estar disponible para los usuarios en todo momento. Cualquier error o caída en el sistema puede afectar directamente la experiencia del usuario y, en el caso de aplicaciones críticas como servicios bancarios o de comercio electrónico, causar pérdidas económicas significativas. Si una tienda online sufre una caída del servidor durante un día de rebajas importantes, los usuarios no podrán acceder para hacer sus compras, lo que puede resultar en una pérdida considerable de ingresos.

- Optimización y seguridad: En producción, el rendimiento y la seguridad son prioritarios. Deben aplicarse optimizaciones para que la aplicación cargue rápido y maneje adecuadamente grandes cantidades de tráfico. También es vital que los datos de los usuarios estén protegidos y que se prevengan posibles vulnerabilidades de seguridad. Un sitio de streaming de vídeo debe optimizar su entorno de producción para que pueda manejar grandes volúmenes de usuarios simultáneos sin ralentizarse, mientras que un sistema de banca en línea debe priorizar la protección de la información sensible.
- Monitorización constante: Las aplicaciones en producción deben estar bajo una monitorización continua para detectar problemas antes de que afecten a los usuarios. Esto puede incluir el seguimiento del rendimiento, la carga del servidor, errores inesperados o incluso ataques de seguridad. Si una red social experimenta una carga inesperada, los sistemas de monitorización pueden alertar a los administradores para que tomen medidas (como escalar los recursos) antes de que los usuarios noten problemas.
- Despliegues controlados: El despliegue en producción debe realizarse con cuidado. Muchas veces se usan técnicas como el despliegue gradual o canary releases (desplegar la nueva versión solo para un pequeño porcentaje de usuarios primero) para asegurarse de que no se introducen errores en el sistema que puedan afectar a todos los usuarios de una vez. Si una aplicación de mensajería introduce una nueva funcionalidad, puede activarla primero para el 5% de sus usuarios. Si todo funciona bien, se despliega gradualmente para el resto.

¿Qué pasa si algo falla en producción?

A pesar de todas las pruebas, a veces los errores no se detectan hasta que la aplicación está en producción. En estos casos, es importante tener un plan de contingencia para revertir los cambios y solucionar el problema sin que los usuarios sufran demasiadas interrupciones. Si un nuevo sistema de pagos causa errores inesperados, el equipo de desarrollo puede revertir el despliegue al estado anterior mientras solucionan el problema, minimizando el impacto en los clientes.

4. Organización de recursos en una aplicación web.

Cuando se desarrolla una aplicación web, no solo se trata de escribir código. Hay muchos recursos que forman parte de la estructura de una aplicación: programas, imágenes, hojas de estilo, archivos de configuración, documentos, entre otros. Organizar todos estos elementos de manera eficiente es fundamental para que la aplicación sea fácil de mantener, rápida y escalable. Aquí se explica cómo organizar estos recursos y qué papel juega cada uno dentro de una aplicación web.

4.1. Programas.

Los programas o archivos de código son el núcleo de la aplicación. Estos archivos pueden estar escritos en varios lenguajes dependiendo del tipo de aplicación, como JavaScript, Python, PHP, Ruby, o Node.js. Cada uno de estos lenguajes sirve para diferentes propósitos dentro del desarrollo web.

JavaScript es el lenguaje que se utiliza principalmente en el frontend, es decir, la parte de la aplicación que los usuarios ven e interactúan. Por ejemplo, si al hacer clic en un botón aparece un mensaje emergente, probablemente esté controlado por JavaScript. También se usa en el backend con Node.js para manejar el servidor. En una aplicación de comercio electrónico, JavaScript podría validar que el usuario haya completado todos los campos de un formulario antes de enviarlo.

PHP/Python/Ruby son lenguajes que se suelen usar en el backend, la parte de la aplicación que se ejecuta en el servidor. Aquí es donde se maneja la lógica de negocio, se interactúa con bases de datos y se gestionan las solicitudes del usuario. Cuando un usuario inicia sesión en una aplicación, el backend (por ejemplo, en PHP o Python) valida las credenciales ingresadas comparándolas con la base de datos y responde si son correctas o no.

Anotación

Estos archivos de código suelen estar organizados en carpetas separadas para mantener claro cuál es la lógica del frontend y cuál pertenece al backend, así como para facilitar la actualización y la búsqueda de errores.

4.2. Hojas de estilos.

Las hojas de estilos son archivos CSS que se encargan de la apariencia de la aplicación web. Aquí se define todo lo relacionado con la presentación visual, como los colores, las fuentes, el tamaño de los elementos, el diseño responsivo (es decir, cómo se adapta la página a diferentes tamaños de pantalla), entre otros aspectos.

CSS (Cascading Style Sheets) es el lenguaje utilizado para darle estilo a las páginas web. Define cómo se deben ver los diferentes elementos del HTML, desde los botones hasta los textos y las imágenes. Si quieres que todos los títulos de la página tengan el mismo color y fuente, creas una

regla en el archivo CSS que lo defina, y esa regla se aplicará automáticamente a todos los títulos. Por ejemplo:

```css
/* Regla para todos los títulos */
h1, h2 {
    color: #00796b; /* Color verde oscuro */
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; /* Fuente deseada */
}
```

Es común tener varias hojas de estilo, por ejemplo, una para el diseño general de la página y otras para temas específicos, como versiones móviles o modos oscuros. Una buena práctica es mantener los archivos CSS organizados en carpetas, para que sea fácil encontrar el archivo que corresponde a cada parte de la aplicación. Por ejemplo:

```
/mi-proyecto-web
|
├── index.html
|
├── /css
|   ├── styles.css            # Hoja de estilo principal
|   ├── mobile.css            # Hoja de estilo para versión móvil
|   └── dark-theme.css        # Hoja de estilo para modo oscuro
|
└── /js
    └── script.js             # Archivo JavaScript (opcional)
```

4.3. Ficheros de configuración.

Los ficheros de configuración permiten definir parámetros importantes sin tener que modificar el código fuente. Estos archivos configuran aspectos como las conexiones a bases de datos, los parámetros de seguridad, las credenciales de acceso a servicios externos, entre otros.

En lenguajes como Node.js, se utilizan archivos JSON o archivos .env (variables de entorno). En frameworks como Django (Python), se suelen usar archivos .py específicos para configuraciones. Por ejemplo, si tienes una base de datos en un servidor externo, el archivo de configuración contendrá la dirección del servidor, el nombre de usuario y la contraseña necesarios para conectar la aplicación con la base de datos. La siguiente imagen es un ejemplo de un archivo de configuración usando un archivo .env, que se utiliza comúnmente en aplicaciones Node.js para almacenar variables de entorno:

```
# .env

# Configuración de la base de datos
DB_HOST=localhost
DB_PORT=5432
DB_USER=mi_usuario
DB_PASS=mi_contraseña
DB_NAME=mi_base_de_datos

# Configuración de la aplicación
SECRET_KEY=mi_clave_secreta
API_URL=https://api.ejemplo.com
PORT=3000
```

- DB_HOST: La dirección del servidor de la base de datos.
- DB_PORT: El puerto en el que escucha la base de datos.
- DB_USER: Nombre de usuario para acceder a la base de datos.
- DB_PASS: Contraseña para el acceso a la base de datos.
- DB_NAME: Nombre de la base de datos a la que se conecta la aplicación.
- SECRET_KEY: Clave secreta utilizada para la seguridad (por ejemplo, para firmar tokens).
- API_URL: URL base para una API externa que la aplicación podría estar utilizando.
- PORT: Puerto en el que la aplicación se ejecuta, que se puede cambiar según el entorno (desarrollo, producción, etc.).

La organización de estos archivos es clave. Se suelen almacenar en carpetas específicas y muchas veces están protegidos para evitar que alguien externo acceda a información sensible (como claves API o contraseñas).

4.4. Imágenes.

Las imágenes son una parte fundamental de la mayoría de las aplicaciones web, ya que ayudan a crear una experiencia visual atractiva para los usuarios. Sin embargo, si no se gestionan correctamente, pueden hacer que la aplicación sea lenta, sobre todo si las imágenes no están optimizadas o si no se organizan adecuadamente.

Es importante comprimir las imágenes para que ocupen menos espacio y se carguen más rápido en la web. Existen formatos como JPEG y PNG que son adecuados dependiendo del tipo de imagen (las fotos suelen ir en JPEG, mientras que los gráficos con transparencia o vectores se guardan en PNG).

Las imágenes se suelen almacenar en carpetas específicas dentro de la estructura de la aplicación, separándolas por tipo (por ejemplo, iconos, fotos de productos, banners, etc.) para facilitar su uso y gestión:

```
/mi-aplicacion
│
├── /images                 # Carpeta principal para todas las imágenes
│   ├── /icons              # Carpeta para iconos
│   │   ├── icono1.png
│   │   ├── icono2.png
│   │   └── icono3.png
│   │
│   ├── /productos          # Carpeta para fotos de productos
│   │   ├── producto1.jpg
│   │   ├── producto2.jpg
│   │   └── producto3.jpg
│   │
│   ├── /banners            # Carpeta para banners
│   │   ├── banner1.jpg
│   │   ├── banner2.jpg
│   │   └── banner3.jpg
│   │
│   └── /fondos            # Carpeta para imágenes de fondo
│       ├── fondo1.jpg
│       └── fondo2.jpg
│
└── index.html              # Archivo principal de la aplicación
```

4.5. Documentos.

Los documentos en una aplicación web pueden incluir archivos como PDFs, hojas de cálculo, o cualquier otro tipo de archivo que los usuarios necesiten descargar o consultar desde la aplicación.

Estos documentos pueden almacenarse directamente en el servidor o utilizar servicios en la nube, como Amazon S3, para reducir la carga en el servidor y mejorar la escalabilidad. Por ejemplo, en un portal educativo, los documentos que contienen materiales de clase (como presentaciones en PDF) deben estar organizados por temas o asignaturas para que los estudiantes puedan encontrarlos fácilmente.

Es importante definir quién puede acceder a estos documentos. Por ejemplo, si la aplicación maneja documentos confidenciales, debe haber mecanismos de autenticación para asegurarse de que solo los usuarios autorizados puedan ver o descargar esos archivos.

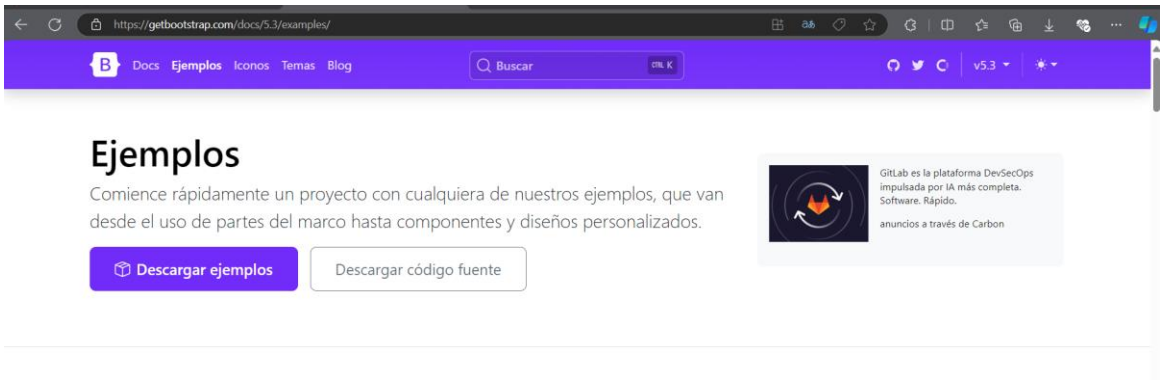
4.6. Bibliotecas de componentes (librerías).

Las librerías son colecciones de código reutilizable que ayudan a los desarrolladores a no tener que reinventar la rueda cada vez que necesitan realizar una tarea común, como interactuar con una API o manipular el DOM (Document Object Model) en una página web.

Algunas de las bibliotecas JavaScript: más usadas son jQuery, Lodash o Axios. Estas librerías proporcionan funciones útiles que facilitan el desarrollo web. Si tu aplicación necesita hacer

peticiones a una API para obtener datos, puedes usar Axios, que te facilita mucho la tarea en comparación con hacerlo directamente con JavaScript nativo.

Muchas veces, las aplicaciones web usan bibliotecas de componentes de interfaz de usuario (UI), como Bootstrap o Material-UI, que proporcionan estilos y componentes listos para usar, como botones, menús o formularios. Por ejemplo, si quieres que tu aplicación tenga un diseño moderno y responsivo sin tener que diseñar todo desde cero, podrías usar Bootstrap, que te ofrece una gran cantidad de componentes visuales y estilos predefinidos.



Pie de imagen: Ejemplos descargables de Bootstrap.

Actividad 14

En esta actividad, utilizarás jQuery para realizar una acción básica de manipulación del DOM.

Crea una página HTML con un botón que diga "Cambiar color" y un párrafo con algún texto.
En el archivo HTML, enlaza la biblioteca jQuery añadiendo el siguiente enlace en la sección <head>:

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

Usa jQuery para que, al hacer clic en el botón "Cambiar color", el color del texto del párrafo cambie a rojo. Ejemplo de código jQuery:

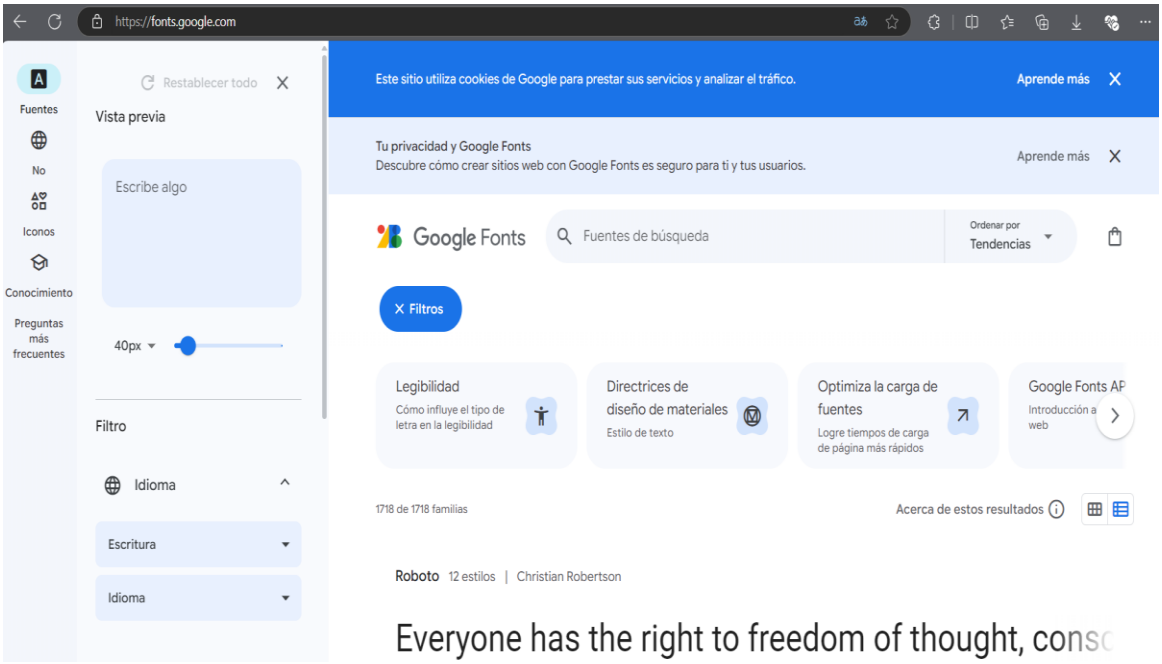
```
$(document).ready(function(){
    $("#changeColorButton").click(function(){
        $("p").css("color", "red");
    });
});
```

Abre el archivo HTML en tu navegador y verifica que el texto cambie de color al hacer clic en el botón.

4.7. Otros archivos

Además de los recursos principales mencionados, existen otros archivos que son importantes en el desarrollo y despliegue de una aplicación web:

- Archivos de fuentes: Si la aplicación utiliza tipografías personalizadas, estas deben estar incluidas en la carpeta de recursos y vinculadas correctamente en las hojas de estilo. También se pueden usar servicios externos como Google Fonts para cargar las fuentes directamente desde la web. Por ejemplo, si tu aplicación tiene un diseño que incluye una tipografía especial para los títulos, debes asegurarte de que esa fuente esté disponible para todos los usuarios, sin importar el navegador o el dispositivo que usen.



Pie de imagen: Sitio web de Google Fonts.

- Archivos de logs: Los logs son archivos que registran lo que ocurre en la aplicación, como errores, accesos o eventos importantes. Estos archivos son muy útiles para el mantenimiento y la depuración de problemas en el servidor. Si un usuario reporta que no puede iniciar sesión en la aplicación, puedes revisar los archivos de logs para ver qué error ocurrió en el backend y solucionarlo.
- Archivos de traducción: Si la aplicación es multilingüe, los archivos de traducción permiten gestionar los diferentes idiomas de la aplicación. Estos archivos suelen estar organizados por idioma y contienen todas las cadenas de texto que se muestran en la interfaz. En una tienda online que funciona en varios países, el archivo de traducción en español podría contener textos como "Añadir al carrito", mientras que el archivo en inglés tendría la traducción correspondiente, "Add to cart".

5. Seguridad en una aplicación web.

La seguridad en una aplicación web es esencial para proteger tanto los datos de los usuarios como la integridad de la aplicación misma. Una vulnerabilidad de seguridad puede tener consecuencias graves, como la pérdida de datos, la exposición de información sensible o el mal funcionamiento del sistema. Por eso, en el desarrollo de aplicaciones web es necesario implementar múltiples niveles de seguridad y seguir ciertos estándares que aseguran que la aplicación sea segura frente a ataques.

5.1. Niveles de seguridad. Estándares.

Al hablar de seguridad en aplicaciones web, existen diferentes niveles que deben ser considerados. Cada uno de estos niveles protege distintos aspectos del sistema:

- Nivel de aplicación: En este nivel se protege el código de la aplicación y las interacciones entre el usuario y la aplicación. Aquí es donde se utilizan técnicas para prevenir ataques como inyección SQL, Cross-Site Scripting (XSS) o Cross-Site Request Forgery (CSRF). Por ejemplo, para evitar una inyección SQL, que es cuando un atacante intenta manipular una consulta a la base de datos a través de un formulario, se deben usar técnicas como la sanitización de entradas, que limpian los datos que provienen del usuario antes de procesarlos.
- Nivel de transporte: En este nivel, se asegura que la información que viaja entre el cliente (el navegador del usuario) y el servidor esté protegida. Aquí es donde entra en juego el uso de HTTPS, que cifra la información para que no pueda ser interceptada fácilmente por atacantes. Por ejemplo, un sitio de compras en línea debe usar HTTPS para proteger la información sensible, como los números de tarjetas de crédito, mientras viajan entre el navegador del usuario y el servidor.
- Nivel de red y servidor: Aquí se protege el servidor donde se aloja la aplicación y la red que la conecta con los usuarios. Es común utilizar firewalls y otros sistemas para bloquear accesos no autorizados y evitar ataques de denegación de servicio (DDoS), que pueden sobrecargar el servidor y hacerlo inaccesible. Por ejemplo, un firewall puede configurarse para bloquear el tráfico desde direcciones IP sospechosas o para limitar la cantidad de solicitudes que puede hacer un mismo usuario en un periodo corto de tiempo, lo que previene ataques de fuerza bruta.

Estándares de seguridad como OWASP (Open Web Application Security Project) ofrecen guías y recomendaciones para asegurar las aplicaciones web contra las vulnerabilidades más comunes. Estas guías son seguidas por la mayoría de los desarrolladores para mantener la seguridad de sus aplicaciones al día con las mejores prácticas.

Saber más...

A través del siguiente enlace puedes descargar una “Una Guía para Construir Aplicaciones y Servicios Web Seguros” proporcionada por OWASP (Open Web Application Security Project):

https://wiki.owasp.org/images/b/b2/OWASP_Development_Guide_2.0.1_Spanish.pdf

5.2. Conceptos y técnicas de identificación, autenticación y autorización o control de acceso.

Dentro de la seguridad de una aplicación web, tres conceptos clave son la identificación, la autenticación y la autorización. Estos términos a veces se confunden, pero cada uno tiene un papel distinto:

- Identificación: Es simplemente el proceso de identificar al usuario. En términos simples, es cuando un usuario indica quién es, por ejemplo, introduciendo un nombre de usuario o una dirección de correo electrónico. Por ejemplo, cuando un usuario ingresa su nombre de usuario en una aplicación, está identificándose ante el sistema.
- Autenticación: Este proceso verifica que la persona que se ha identificado es realmente quien dice ser. Normalmente se hace a través de una contraseña, pero también puede incluir métodos más avanzados como autenticación de dos factores (2FA) o huellas dactilares. Por ejemplo, tras ingresar su nombre de usuario, el sistema le pedirá al usuario su contraseña. Si coincide con la que está almacenada en la base de datos, el usuario ha sido autenticado.
- Autorización: Una vez que el sistema ha identificado y autenticado al usuario, el paso siguiente es determinar qué puede o no puede hacer dentro de la aplicación. Este proceso se llama autorización y define qué recursos o funcionalidades están disponibles según los permisos del usuario. Por ejemplo, un usuario autenticado como "administrador" podrá acceder a más funciones (como eliminar usuarios o modificar configuraciones), mientras que un usuario "estándar" solo podrá ver su perfil o hacer operaciones básicas.

Estas tres etapas forman el control de acceso, que garantiza que cada usuario acceda solo a la información o acciones que le corresponden según su rol en la aplicación.

5.3. Identificación y autenticación avanzada. Certificados digitales.

Con el aumento de amenazas en línea, se ha vuelto común implementar métodos avanzados de autenticación.

Uno de los más utilizados es la autenticación de dos factores (2FA), donde el usuario debe proporcionar algo que sabe (como una contraseña) y algo que tiene (como un código enviado a su teléfono). Este método requiere que, después de ingresar la contraseña, el usuario introduzca un código que se le envía a través de SMS, correo electrónico o mediante una aplicación de autenticación como Google Authenticator. Por ejemplo, un banco en línea podría requerir que, después de que el usuario introduzca su contraseña, ingrese un código enviado a su teléfono para asegurarse de que es quien dice ser:

Otra técnica avanzada es el uso de certificados digitales, que son archivos emitidos por autoridades certificadoras (CA) y que verifican la identidad del usuario o del servidor. Los certificados digitales son archivos que aseguran que una comunicación o una transacción se está realizando con una identidad legítima. Los servidores web utilizan certificados SSL/TLS para garantizar que los usuarios están conectados a un servidor legítimo y que la información intercambiada está cifrada. Un sitio web seguro muestra un candado en la barra de direcciones del navegador, indicando que el sitio tiene un certificado SSL válido y que las comunicaciones están cifradas.

Actividad 15

Visita un sitio web que utilice HTTPS (por ejemplo, una tienda en línea o tu banco).

Haz clic en el candado que aparece en la barra de direcciones de tu navegador.

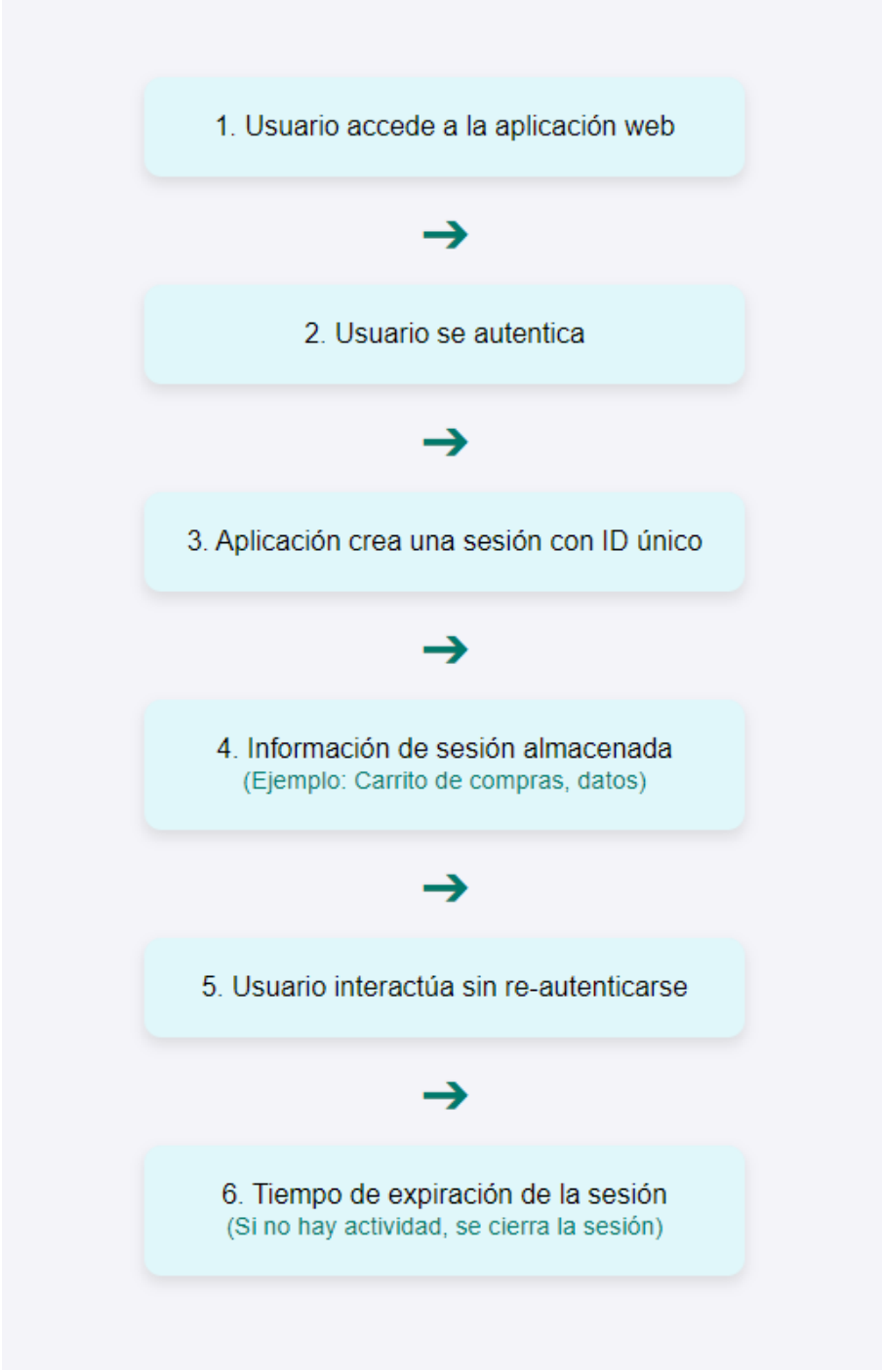
Explica qué información encontraste sobre el certificado SSL (emisor, validez, tipo de cifrado) y cómo puedes asegurarte de que es un sitio seguro.

5.4. Concepto de sesión. Conservación de sesiones.

Cuando un usuario accede a una aplicación web y se autentica, se inicia una sesión. La sesión es un período en el que el usuario puede interactuar con la aplicación sin necesidad de volver a autenticarse cada vez que realiza una acción. La aplicación almacena información temporal sobre el usuario mientras dura esta sesión.

La sesión se mantiene a través de un identificador único que se asigna al usuario al iniciar sesión. Este identificador puede almacenarse en cookies en el navegador o en el servidor. La conservación de la sesión es importante para que el usuario no tenga que iniciar sesión cada vez que navega entre diferentes páginas dentro de la misma aplicación. Por ejemplo, si un usuario inicia sesión en una tienda en línea y añade artículos a su carrito, la aplicación mantiene su sesión abierta para que pueda completar la compra sin tener que volver a autenticarse.

Una sesión puede tener un tiempo de expiración, lo que significa que si el usuario no realiza ninguna acción durante un tiempo determinado, la sesión se cierra automáticamente por motivos de seguridad.



Pie de imagen: Gestión de sesiones en aplicaciones web.

5.5. Sistemas de uso común para la conservación de las sesiones en aplicaciones web. Single Sign-on y Single Sign-out.

Existen diferentes técnicas y sistemas para la conservación de sesiones en aplicaciones web. Dos de los métodos más comunes son el Single Sign-on (SSO) y el Single Sign-out (SSO):

- Single Sign-on (SSO): Este sistema permite que un usuario inicie sesión una sola vez para acceder a múltiples aplicaciones relacionadas sin tener que autenticarse de nuevo en cada una. Es muy útil en grandes organizaciones donde los empleados necesitan acceder a varios sistemas, pero no quieren tener que recordar diferentes contraseñas o iniciar sesión varias veces. Por ejemplo, si trabajas en una empresa que usa SSO, podrías iniciar sesión en tu cuenta de correo corporativo y automáticamente tener acceso al sistema de gestión de proyectos o al sistema de recursos humanos sin tener que iniciar sesión otra vez en cada aplicación.
- Single Sign-out (SSO): Este sistema, a su vez, permite que cuando un usuario cierre sesión en una aplicación, se cierre automáticamente en todas las demás aplicaciones relacionadas. Esto es útil para garantizar que la sesión se cierra completamente en todos los sistemas y que no quede ninguna sesión abierta por error. Por ejemplo, si has accedido a varias aplicaciones empresariales a través de SSO y cierras sesión en una de ellas, el sistema cerrará tu sesión en todas las demás aplicaciones para mayor seguridad.

6. Despliegue de aplicaciones web.

El despliegue de una aplicación web es el proceso por el cual una aplicación, que ha sido desarrollada y probada, se pone en funcionamiento para que los usuarios finales puedan acceder a ella. Este es un paso crítico en el ciclo de vida del desarrollo, ya que implica mover la aplicación desde un entorno controlado (como el entorno de pruebas o pre-producción) a un entorno de producción, donde estará disponible para todo el mundo. Pero, ¿cómo se realiza este despliegue? Vamos a verlo paso a paso.

6.1. Características del proceso de despliegue.

El proceso de despliegue de una aplicación web tiene varias características que lo hacen especial. Aunque cada proyecto puede tener sus particularidades, hay algunos aspectos que se repiten en la mayoría de los casos.

Para evitar errores humanos y reducir el tiempo de despliegue, es común que este proceso se automatice en la medida de lo posible. Esto significa que se usan herramientas y scripts que ejecutan los pasos del despliegue de forma automática. De esta manera, los desarrolladores no tienen que hacerlo manualmente, lo que podría causar problemas si olvidan algún paso o lo ejecutan mal. Imagina que trabajas en una empresa que actualiza su aplicación web varias veces al mes. En lugar de hacer el despliegue manualmente cada vez, se puede configurar un sistema que, con solo ejecutar un comando, cargue todos los archivos en el servidor, actualice las bases de datos y reinicie los servicios de forma automática.

En muchos casos, especialmente en aplicaciones que deben estar disponibles las 24 horas, se intenta realizar un despliegue continuo o despliegue sin interrupciones. Esto significa que la aplicación se actualiza sin necesidad de que los usuarios experimenten tiempos de inactividad. Hay técnicas como el despliegue canario o los blue-green deployments que permiten hacer esto. Por ejemplo, si gestionas una tienda online y lanzas una nueva versión de la web, no puedes permitir que esté inactiva durante el despliegue, ya que los clientes no podrían comprar durante ese tiempo. Con el despliegue continuo, se puede actualizar el sistema en segundo plano, sin que los usuarios se den cuenta.

El despliegue debe realizarse de manera segura, especialmente si la aplicación maneja datos sensibles. Deben tomarse precauciones para asegurarse de que las credenciales (como las claves API, contraseñas de bases de datos, etc.) no se expongan durante el proceso de despliegue. Además, se debe garantizar que la nueva versión de la aplicación esté protegida contra vulnerabilidades de seguridad.

No siempre todo sale bien al realizar un despliegue. Por eso, es fundamental contar con un sistema de rollback, que permita volver rápidamente a la versión anterior si se detecta algún problema en la nueva versión desplegada. Esto evita que los usuarios experimenten errores por mucho tiempo. Por ejemplo, si después de desplegar una nueva funcionalidad en una aplicación de mensajería instantánea los usuarios comienzan a experimentar problemas al enviar mensajes, un rollback permitiría revertir la aplicación a la versión anterior rápidamente para solucionar el problema.

Pie de imagen: Proceso de despliegue de aplicaciones web

6.2. Definición del proceso de despliegue de aplicaciones web.

El proceso de despliegue de una aplicación web no es simplemente "subir" archivos al servidor. Hay una serie de pasos que deben seguirse para garantizar que todo funcione como se espera. A continuación, se describe un proceso típico de despliegue:

1. Preparación del entorno

Antes de realizar el despliegue, es importante asegurarse de que el entorno de producción esté listo para recibir la nueva versión de la aplicación. Esto incluye verificar que los servidores tengan suficiente espacio, que las bases de datos estén en buen estado y que no haya problemas en la red que puedan interferir con el proceso.

Por ejemplo, si tu aplicación web utiliza una base de datos MySQL, antes de desplegar una nueva versión es recomendable verificar que la base de datos no tenga problemas de rendimiento o que no esté a punto de quedarse sin espacio.

2. Compilación y empaquetado

Muchas aplicaciones modernas, especialmente las que usan frameworks como React o Angular, requieren un paso de compilación antes del despliegue. Esto significa que el código fuente de la aplicación se convierte en un formato más eficiente (normalmente archivos JavaScript y CSS

minificados). Luego, todo se empaqueta en un solo archivo o conjunto de archivos listos para ser desplegados en el servidor.

Por ejemplo. si estás desarrollando una aplicación web con Angular, debes ejecutar un comando como ng build que compilará y optimizará tu código para producción, generando los archivos listos para ser cargados al servidor.

Saber más...

Para implementar el proceso de construcción y despliegue de una aplicación web desarrollada con Angular, se pueden seguir estos pasos. A continuación, se expone un ejemplo de cómo se hace esto utilizando la línea de comandos:

Asegúrate de tener Angular CLI instalado. Si no lo tienes, puedes instalarlo globalmente con el siguiente comando:

```
npm install -g @angular/cli
```

Si aún no tienes una aplicación creada en Angular, puedes crear una nueva usando el siguiente comando:

```
ng new mi-aplicacion
```

Esto creará una nueva carpeta llamada mi-aplicacion con la estructura básica de un proyecto Angular.

Accede al directorio de tu aplicación:

```
cd mi-aplicacion
```

Utiliza el siguiente comando para compilar tu aplicación. Esto generará archivos optimizados para producción en la carpeta dist/mi-aplicacion.

```
ng build --prod
```

--prod: Este flag indica que la aplicación debe ser compilada en modo de producción, lo que incluye optimizaciones como minificación y tree shaking.

Después de compilar, los archivos listos para la producción estarán disponibles en dist/mi-aplicacion. Puedes cargar estos archivos en tu servidor web. En el siguiente ejemplo se expone cómo hacerlo usando un servidor HTTP simple con Node.js.

Si deseas probar tu aplicación Angular localmente después de compilarla, puedes usar el siguiente código para crear un servidor básico que sirva los archivos estáticos.

Instalar Express (si no lo tienes):

```
npm install express
```

Crear un archivo server.js en la raíz de tu proyecto:

```
// server.js
const express = require('express');
const path = require('path');

const app = express();
const PORT = process.env.PORT || 8080;

// Servir los archivos estáticos desde la carpeta 'dist'
app.use(express.static(path.join(__dirname, 'dist/mi-aplicacion')));

// En cualquier ruta que no se haya definido, devolver 'index.html'
app.get('/*', (req, res) => {
    res.sendFile(path.join(__dirname, 'dist/mi-aplicacion/index.html'));
});

// Iniciar el servidor
app.listen(PORT, () => {
    console.log(`Servidor escuchando en http://localhost:${PORT}`);
});
```

Para iniciar el servidor y servir tu aplicación compilada, ejecuta:

```
node server.js
```

Abre tu navegador y visita http://localhost:8080 para ver tu aplicación Angular en acción.

3. Despliegue en el servidor

Este es el paso donde los archivos compilados y empaquetados se suben al servidor. Dependiendo de la infraestructura utilizada, esto puede implicar subir archivos a un servidor web o desplegarlos en un sistema de contenedores como Docker. Si la aplicación tiene una base de datos, también se deben realizar las actualizaciones necesarias (nuevas tablas, cambios en la estructura de datos, etc.).

Por ejemplo, si tu aplicación web está alojada en Amazon Web Services (AWS), podrías usar una herramienta como AWS CodeDeploy para automatizar el proceso de despliegue, de manera que los archivos correctos se suban al servidor y la aplicación se reinicie automáticamente.

4. Verificación post-despliegue

Después de realizar el despliegue, es fundamental hacer una verificación para asegurarse de que todo funcione como se espera. Esto incluye comprobar que las páginas se cargan correctamente, que las nuevas funcionalidades están disponibles y que no hay errores visibles para los usuarios. También es buena idea realizar pruebas adicionales en el entorno de producción para detectar cualquier problema que no haya aparecido durante las pruebas previas.

Por ejemplo, después de actualizar una aplicación web de reservas de hoteles, los desarrolladores pueden realizar pruebas simulando el proceso de reserva desde diferentes dispositivos para asegurarse de que los usuarios puedan completar una reserva sin problemas.

5. Monitoreo y gestión de errores

Una vez que la aplicación está en producción, se debe monitorizar constantemente para asegurarse de que no haya problemas de rendimiento o errores que afecten a los usuarios. Hay muchas herramientas que permiten rastrear el estado de la aplicación en tiempo real y enviar alertas si algo va mal.

New Relic o Sentry son herramientas que permiten monitorear aplicaciones en producción, alertando a los desarrolladores si ocurre algún error o si hay problemas de rendimiento, como tiempos de carga lentos:

La verificación es una fase clave dentro del despliegue, ya que permite asegurarse de que todo ha funcionado correctamente antes de dar por concluido el proceso. Esta verificación implica revisar que todos los elementos de la aplicación se han desplegado correctamente y que no hay errores críticos que puedan afectar la experiencia del usuario.

Tipos de verificación:

- Pruebas de regresión: Se realizan pruebas para asegurarse de que las funcionalidades que ya existían antes del despliegue siguen funcionando correctamente con la nueva versión. A veces, una actualización puede afectar a partes de la aplicación que no estaban directamente relacionadas, y estas pruebas ayudan a detectar esos problemas.
- Pruebas funcionales: Se verifica que las nuevas funcionalidades introducidas en la aplicación funcionan como se espera. Esto puede incluir pruebas manuales o automatizadas. Por ejemplo, si has añadido un nuevo método de pago a tu tienda online, es esencial hacer pruebas en producción para asegurarte de que los usuarios pueden completar una compra usando ese nuevo método sin problemas.

7. Prueba de autoevaluación.

¿Qué herramienta se usa comúnmente para el control de versiones en el desarrollo web?

a) Visual Studio Code

b) Git

c) Webpack

¿Cuál de estas herramientas es un framework para el desarrollo de backend en JavaScript?

a) Express.js

b) Django

c) Ruby on Rails

El entorno de pre-producción en el desarrollo web se utiliza principalmente para:

a) Desarrollar nuevas funcionalidades

b) Simular condiciones reales de producción y realizar pruebas

c) Monitorear el rendimiento en tiempo real

El sistema Single Sign-on (SSO) permite a los usuarios:

a) Iniciar sesión una vez para acceder a múltiples aplicaciones

b) Usar la misma contraseña para todas las cuentas

c) Acceder a una aplicación sin autenticarse

En el despliegue continuo de aplicaciones web, el objetivo principal es:

a) Actualizar la aplicación sin interrumpir el servicio

b) Optimizar el uso de imágenes en la web

c) Garantizar la personalización de la interfaz de usuario

El patrón de diseño _____ divide una aplicación en modelo, vista y controlador.

En el entorno de _____ se prueban las funcionalidades bajo condiciones similares a las de producción.

Los archivos _____ permiten definir parámetros importantes como las conexiones a bases de datos sin modificar el código fuente.

El framework _____ es utilizado para desarrollar aplicaciones en Python.

En el despliegue de aplicaciones, es común utilizar herramientas como AWS CodeDeploy para _____ el proceso.

Verificación de aplicaciones web