ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Tema 1.2:

Tipos de aplicaciones Web.

Introducción

Antes de iniciar este tema, vamos a dar un recorrido por algunos conceptos importantes que son importantes dominar para poder entender el tema **Tipos de aplicaciones web**.

Ya que la información es extensa, es necesario consolidar los conceptos más importantes respecto a la asignatura de Programación Web Avanzada.

En este tema analizaremos conceptos como:

API, Rest, API Rest, Restfull, Contenedores, Web, Aplicaciones Web, Arquitecturas Web, entre otros temas.

Conceptos Importantes.

Sitio Web.

El concepto de Sitio Web ha evolucionado significativamente con el desarrollo de la web. Inicialmente, se concebía como un conjunto de documentos estáticos a los que se accedía a través de enlaces de hipertexto. En esta etapa, conocida como web 1.0, los usuarios tenían un papel pasivo, limitándose a consumir el contenido sin capacidad para interactuar o modificarlo (Velasco & Herrero, 2022).

Con la llegada de la web 2.0, los sitios web se transformaron en entornos dinámicos y participativos. Ahora, los usuarios podían interactuar entre sí y con el contenido, generando una arquitectura de participación extensa. Los sitios web dejaron de ser solo fuentes de información para convertirse en espacios de intercambio y colaboración.

Conceptos Importantes.

Sitio Web.

Finalmente, la web 3.0 o web semántica ha llevado la evolución de los sitios web hacia un nivel superior. En esta etapa, los sitios web no se limitan al significado textual del contenido, sino que utilizan tecnologías avanzadas para comprender y atender las necesidades de los usuarios de manera más precisa. Se almacenan las preferencias de los usuarios y se combinan los contenidos con redes sociales y otras aplicaciones para ofrecer una experiencia más personalizada y funcional.

A pesar de estas transformaciones, el término "sitio web" sigue siendo relevante para conceptualizar estos espacios digitales, ya que abarca todos los elementos construidos a lo largo del tiempo, incluidas las consideraciones relativas a la web 3.0.

Conceptos Importantes.

Sitio Web.

Un Sitio Web es un espacio digital en la web que contiene información y funcionalidades, y que puede variar desde ser estático hasta ser dinámico y altamente interactivo. Su evolución ha estado marcada por la incorporación de nuevas tecnologías y paradigmas que han permitido una mayor participación y personalización de la experiencia del usuario.

Conceptos Importantes.

Portal Web.

Un portal web es una plataforma digital que proporciona acceso a una amplia variedad de información, servicios y recursos en línea, diseñada para interactuar directamente con los usuarios y satisfacer sus necesidades específicas de información y comunicación (Soto Eras, 2021).

Estos portales permiten a los usuarios acceder a información relevante y participar en actividades interactivas utilizando protocolos de comunicación. En la actualidad, existen millones de portales web en todo el mundo, facilitando la comunicación global y el acceso rápido a la información con solo un clic.

Durante la pandemia del COVID-19, muchas organizaciones y empresas han recurrido a los portales web como medio para mantener sus operaciones estables.

Conceptos Importantes.

Portal Web.

Los portales web desarrollados por entidades gubernamentales, instituciones educativas y organizaciones sin fines de lucro, desempeñan un papel crucial en la prestación de servicios y la difusión de información en sus respectivas comunidades.

Estos portales, son herramientas clave para ofrecer servicios, promover proyectos, propuestas y colaborar con la sociedad en diversos ámbitos, incluyendo educación, comercio electrónico, salud, turismo, entre otros.

Conceptos Importantes.

Aplicación Web

Una Aplicación Web es un tipo de software que se ejecuta a través de un navegador web y se accede a través de internet.

A diferencia de los sitios web tradicionales, las aplicaciones web ofrecen funcionalidades y servicios específicos, que pueden variar desde herramientas de productividad hasta redes sociales, servicios bancarios en línea, juegos y mucho más.

Estas aplicaciones son altamente interactivas y dinámicas, lo que significa que permiten una mayor interacción con el usuario y pueden realizar acciones complejas.

Por ejemplo, una aplicación web puede permitir a los usuarios editar documentos en tiempo real, colaborar en proyectos, enviar mensajes instantáneos o realizar transacciones bancarias.

Conceptos Importantes.

Aplicación Web

Una de las características distintivas de las aplicaciones web es su capacidad para ofrecer una funcionalidad más avanzada en comparación con los sitios web estáticos o dinámicos.

Esto se debe a que las aplicaciones web pueden procesar datos en tiempo real, almacenar información del usuario y ofrecer una experiencia más personalizada.

Las aplicaciones web pueden ser tanto simples como complejas. Por ejemplo, una calculadora en línea es una aplicación web simple, mientras que una plataforma de gestión empresarial que permite a los usuarios administrar proyectos, realizar seguimiento de inventario y generar informes detallados es un ejemplo de una aplicación web compleja.

Conceptos Importantes.

Aplicación Web

Las aplicaciones web son una evolución de los sitios web tradicionales, ya que ofrecen una funcionalidad más avanzada, una mayor interactividad y la capacidad de realizar tareas específicas de manera más eficiente, todo dentro de un navegador web.

Conceptos Importantes.

Diferencia entre sitio web, portal web y aplicación web:

- **Sitio Web** es una colección de páginas web relacionadas con un tema específico
- **Portal Web** es una plataforma que proporciona acceso a una amplia variedad de recursos en línea
- **Aplicación Web** es un software accesible a través de un navegador que ofrece funcionalidades específicas, siendo más interactivas y dinámicas que los sitios web tradicionales.

Conceptos Importantes.

REST.

Un servicio REST se define como una agregación de diferentes recursos que pueden ser alcanzados desde un identificador universal de recurso (URI) base.

Un recurso representa a una entidad del mundo real cuyo estado está expuesto y puede cambiarse accediendo a un URI. Una Representación es la descripción de los mensajes enviados o recibidos de un Recurso en términos de un lenguaje tecnológico.

Actualmente XML y JSON son los idiomas más populares para describir estos mensajes

Conceptos Importantes.

SOAP.

La implementación de servicios web por protocolo simple de acceso a objetosse desarrolló como una alternativa al estándar CORBA (Common Object Request Broker Architecture). Para garantizar el transporte de datos en SOAP, se utilizan protocolos como HTTP, SMTP, entre otros, en formato XML.

En este enfoque, un proveedor de servicios publica una descripción del servicio o una interfaz para el registro de servicios, por lo que el solicitante del servicio puede encontrar una instancia de servicio correcta y utilizarla. Algunos problemas de rendimiento en SOAP se producen al formar el mensaje SOAP ya que agrega un encabezado adicional y partes al cuerpo al mensaje.

Los servicios Web basados en SOAP incluyen una variedad de estándares, tales como WSDL, WSBPEL, WS-Security, WS-Addressing Estas normas fueron desarrolladas por organizaciones de normalización, como W3C y OASIS.

Conceptos Importantes.

Integración Continua (CI).

Es una práctica fundamental en el desarrollo de software que implica realizar integraciones automáticas de un proyecto con la mayor frecuencia posible, generalmente a diario.

El objetivo principal de la CI es detectar fallos en el código lo antes posible durante el proceso de desarrollo.

En la integración continua, cada vez que se agrega una nueva parte al sistema, se crean automáticamente casos de prueba para cubrir tanto las partes existentes como las nuevas.

Además, el software se prueba y construye de forma automática, proporcionando una retroalimentación inmediata a los desarrolladores sobre los nuevos códigos integrados.

Conceptos Importantes.

Existen varias prácticas esenciales dentro de la integración continua:

- **Integración frecuente del código:** Se espera que los desarrolladores integren su código en el repositorio compartido al menos una vez al día. Esto asegura que los cambios se incorporen al proyecto de manera regular.
- **No integrar código "roto":** El código "roto" es aquel que contiene errores que podrían afectar el funcionamiento del proyecto. Por lo tanto, no se debe integrar código que presente errores.
- **Corregir compilaciones no funcionales de inmediato:** Cualquier problema que impida que la compilación se realice correctamente debe ser abordado de inmediato. Esto incluye errores en la compilación, la base de datos o la implementación.
- **Escribir pruebas automatizadas:** Es crucial contar con pruebas automatizadas que cubran todas las funcionalidades del proyecto. Estas pruebas garantizan que los cambios no introduzcan nuevos errores en el código existente.

Conceptos Importantes.

- **Aprobar todas las pruebas e inspecciones:** Para que una compilación se considere exitosa, todas las pruebas automatizadas deben pasar satisfactoriamente. Esto es fundamental para garantizar la calidad del software.
- **Ejecutar compilaciones privadas:** Los desarrolladores pueden realizar compilaciones locales del software desde el repositorio compartido en sus propias estaciones de trabajo. Esto les permite verificar la integración antes de enviarla al servidor principal, asegurando que no se presenten problemas.

La integración continua es una práctica que garantiza la calidad y estabilidad del software al realizar integraciones automáticas y frecuentes, junto con pruebas exhaustivas y retroalimentación inmediata sobre cualquier problema detectado.

Conceptos Importantes.

Contenedor de Aplicaciones

Los contenedores de aplicaciones son como cajas mágicas que contienen todo lo necesario para que un software funcione correctamente. Imagina que tienes una caja que contiene no solo el programa que quieres ejecutar, sino también todas las bibliotecas, archivos y configuraciones que necesita.

Esta caja es completamente independiente y puede ejecutarse en cualquier lugar, ya sea en tu computadora portátil, en un servidor de prueba o en un centro de datos.

Ahora, hablemos de Docker. Piensa en Docker como el fabricante de estas cajas mágicas. Docker es como una fábrica que automatiza todo el proceso de empaquetado, entrega y despliegue de estas cajas, llamadas contenedores. Estos contenedores son livianos, portátiles y autónomos, lo que significa que pueden moverse fácilmente de un lugar a otro y ejecutarse en diferentes entornos sin problemas.

Conceptos Importantes.

Contenedor de Aplicaciones

Cada contenedor de Docker es como una pequeña isla privada. Imagina que tienes varias islas, cada una con su propio ecosistema único y aislado del resto.

Del mismo modo, en un servidor, puedes tener varios contenedores ejecutándose, y cada uno está completamente aislado del otro y del sistema operativo subyacente.

Los contenedores de aplicaciones son como cajas mágicas que contienen todo lo necesario para que un software funcione, y Docker es la herramienta que automatiza la creación y gestión de estas cajas mágicas, permitiéndoles ejecutarse de manera confiable y consistente en cualquier lugar.

Tipos de Arquitecturas Web.

Según CTES (2023) existen los siguientes tipos de aplicaciones web.

- Arquitectura Monolíticas.
- Arquitectura Cliente/Servidor.
- Arquitectura de 2, 3, N capas.
- Arquitectura Distribuidas.
- Arquitectura orientada a microservicios.

Arquitecturas Monolíticas.

Son aquellas que las tres partes forman un todo y se ejecutan en el mismo ordenador:

- Interfaz de usuario
- Lógica o reglas de negocios
- Gestión de datos.

Una arquitectura monolítica es una única unidad, que se compone de tres partes:

- Acceso a datos también llamado la capa del Modelo,
- Interfaz de usuario como la vista
- Controlador el cual comunica la capa de la base de datos con la capa de la interfaz.

Arquitecturas Monolíticas.

Las aplicaciones monolíticas son sistemas que emplean una arquitectura en la que todos los componentes están integrados en un solo compilador y lenguaje de programación.

Aunque inicialmente se consideraron ventajosas debido a su ejecución como un solo archivo y su facilidad de desarrollo, hoy en día presentan desafíos.

Al implementar una arquitectura monolítica, las empresas deben sopesar la lógica de negocio y considerar alternativas más escalables e interoperables. Por ejemplo, desarrollar un CRM (Customer Relationship Management) con arquitectura monolítica implica riesgos en el despliegue y actualización debido a la complejidad y la falta de garantía en el mantenimiento del código.

Arquitecturas Monolíticas.

Las aplicaciones monolíticas, aunque históricamente útiles, pueden ser menos apropiadas en entornos empresariales modernos debido a su falta de modularidad y dificultades en el mantenimiento y actualización.

Es importante considerar alternativas más flexibles y adaptativas para satisfacer las necesidades cambiantes de las organizaciones.

Arquitectura Cliente-Servidor.

Es una arquitectura en la que las diferentes tareas son asignadas entre los proveedores y los demandantes o también conocidos como servidores y clientes, que realizan peticiones a uno o más servidores los cuales se encuentran en ejecución para atender la demanda; permitiendo diversificar el trabajo de cada aplicación de esta forma los clientes no se sobrecargan de trabajo (Hernández Berrones, 2020).

La arquitectura Cliente-Servidor tiene algunas caracteristicas:

- El servidor atiende a las peticiones de los clientes con una interfaz única y bien definida
- El cliente no está sujeto a la ubicación de cada servidor, ni el equipo de ejecución; es independiente del sistema operativo.
- Las peticiones de varios clientes pueden ser atendidos por un servidor al mismo tiempo, regulando sus accesos a los recursos.

Arquitectura Cliente-Servidor.

- Siempre los clientes inician la comunicación pidiendo el servicio, esperando pasivamente las peticiones de cada cliente.
- El mensaje es el canal de comunicación de las peticiones de servicio de cada cliente al servidor.
- Los datos del servidor están centralizados, mejorando costos, mantenimiento y el control de la integridad.

Arquitectura Orientada al Servicio (SOA) para diseñar e implementar Web Services.

La Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés) es un enfoque para **diseñar** y **desarrollar** aplicaciones de software, especialmente **Web Services**, que están altamente interconectadas y pueden funcionar juntas de manera eficiente. Los Web Services son piezas de software que permiten la comunicación entre diferentes aplicaciones a través de Internet.

Según el Instituto Nacional de Estándares y Tecnología (NIST), un Web Service se compone de tres capas:

- La capa de Web Services,
- El framework del Web Service
- La Capa Web Server.

Estas capas trabajan juntas para proporcionar funcionalidades y seguridad a los servicios.

Arquitectura Orientada al Servicio (SOA) para diseñar e implementar Web Services.

La **seguridad** es un aspecto importante en el **diseño** de Web Services. La **ISO-WSP,** una arquitectura de flujo de información, divide la seguridad en dos partes:

- T-WSP para manejar datos confidenciales.
- U-WSP para proporcionar funcionalidad general del Web Service.

Un Web Service, según la definición del W3C, es una aplicación de software identificada por una **URI**, cuya **interfaz** y **enlaces** pueden ser definidos, descritos y descubiertos como artefactos **XML**.

Esto significa que los Web Services pueden **comunicarse** entre sí utilizando **mensajes** basados en XML a través de **protocolos** de Internet.

Arquitectura Orientada al Servicio (SOA) para diseñar e implementar Web Services.

La implementación de Web Services ha crecido enormemente, ya que permiten a las organizaciones **conectar diferentes tecnologías y servicios en la web**, lo que facilita la comunicación entre aplicaciones **legadas** y **nuevas**.

Para muchas empresas, adoptar una arquitectura orientada a servicios web les permite **publicar** sus sistemas fácilmente y **utilizarlos** internamente. Esto significa que cada equipo de desarrollo puede elegir las herramientas que prefiera sin afectar la interacción con otros sistemas.

Arquitectura Orientada al Servicio (SOA) para diseñar e implementar Web Services.

Sin embargo, la **alta abstracción** en la **fabricación** y **especificación** de Web Services puede **dificultar** su comprensión. Además, existe el **riesgo** de **exponer vulnerabilidades**, ya que los archivos **WSDL** (Web Service Definition Language) pueden ser utilizados por personas **no autorizadas** para obtener información sobre el **servicio** y realizar **ataques**.

La **flexibilidad**, **interacción** e **integración** requeridas en los sistemas modernos se logran mediante una arquitectura orientada a servicios, que utiliza Web Services para lograr la **integración** y **flexibilidad** de las aplicaciones. Sin embargo, esta flexibilidad también puede llevar a **vulnerabilidades** si no se **controla** adecuadamente, lo que puede resultar en **ataques** de **seguridad** como el **espionaje** de información o la **suplantación** de clientes. Por lo tanto, es crucial mantener un **control riguroso** sobre la seguridad de los Web Services, especialmente en industrias sensibles como la banca o el comercio electrónico.

Aplicación de N capas

En la programación por capas el objetivo principal es la separación de la lógica de negocios de la lógica de diseño; separando de esta manera los datos almacenados en el gestor de bases de datos utilizado (Hernández Berrones, 2020).

Aplicaciones Distribuida.

"Una aplicación distribuida es un sistema compuesto de un conjunto de programas corriendo en múltiples computadoras (host); su arquitectura es un esqueleto de programas diferentes, con una descripción de qué programas están corriendo en qué hosts, cuáles son sus responsabilidades y qué protocolos determinan la forma en la cual diferentes partes del sistema se comunican unas con otras" (Acosta Gonzaga et al, 2006).

Progressive Web Apps (PWA).

Las Progressive Web Apps (PWA) son aplicaciones web desarrolladas con tecnologías específicas y patrones estándar que combinan lo mejor de las aplicaciones nativas y web. Estas aplicaciones se ejecutan en el navegador del dispositivo del usuario y utilizan tecnologías modernas como service workers y manifestos web para brindar una experiencia similar a la de una aplicación nativa.

Una de las características más importantes de las PWA es su capacidad para funcionar sin conexión a Internet o en redes de baja calidad. Esto se logra gracias a los service workers, que son scripts que se ejecutan en segundo plano en el navegador y permiten que la aplicación funcione incluso cuando no hay conexión. Cuando la PWA se carga por primera vez en el navegador, se registran estos service workers para que puedan gestionar la funcionalidad sin conexión en el futuro.

Progressive Web Apps (PWA).

Además, las PWA se sirven a través de HTTPS para garantizar la seguridad de los datos. Esto significa que la comunicación entre el navegador y el servidor web está encriptada, lo que protege la información transmitida entre ellos.

Las Progressive Web Apps son aplicaciones web que ofrecen una experiencia similar a las aplicaciones nativas, con la ventaja de ser rápidas, confiables y accesibles incluso en condiciones de red desfavorables. Utilizan tecnologías como service workers y HTTPS para lograr esto, lo que las convierte en una excelente opción para desarrollar aplicaciones que puedan llegar a una amplia audiencia de usuarios.

Arquitectura orientada a Microservicios.

Las arquitecturas de microservicios se parecen a los patrones SOA en que los servicios son especializados y no tienen conexión directa. Pero, además, descomponen las arquitecturas tradicionales en partes más pequeñas.

Los servicios de la arquitectura de microservicios usan un marco de mensajería común, como las API de RESTful. Utilizan API de RESTful para comunicarse entre sí, sin necesidad de operaciones complejas de conversión de datos ni capas de integración adicionales.

Usar las API de RESTful permite e incluso fomenta una distribución más rápida de nuevas funciones y actualizaciones. Cada servicio es independiente.

Un servicio se puede reemplazar, mejorar o abandonar, sin afectar los demás servicios de la arquitectura. Esta arquitectura liviana optimiza los recursos distribuidos o en la nube y admite la escalabilidad dinámica de los servicios individuales.

Arquitectura orientada a Microservicios.

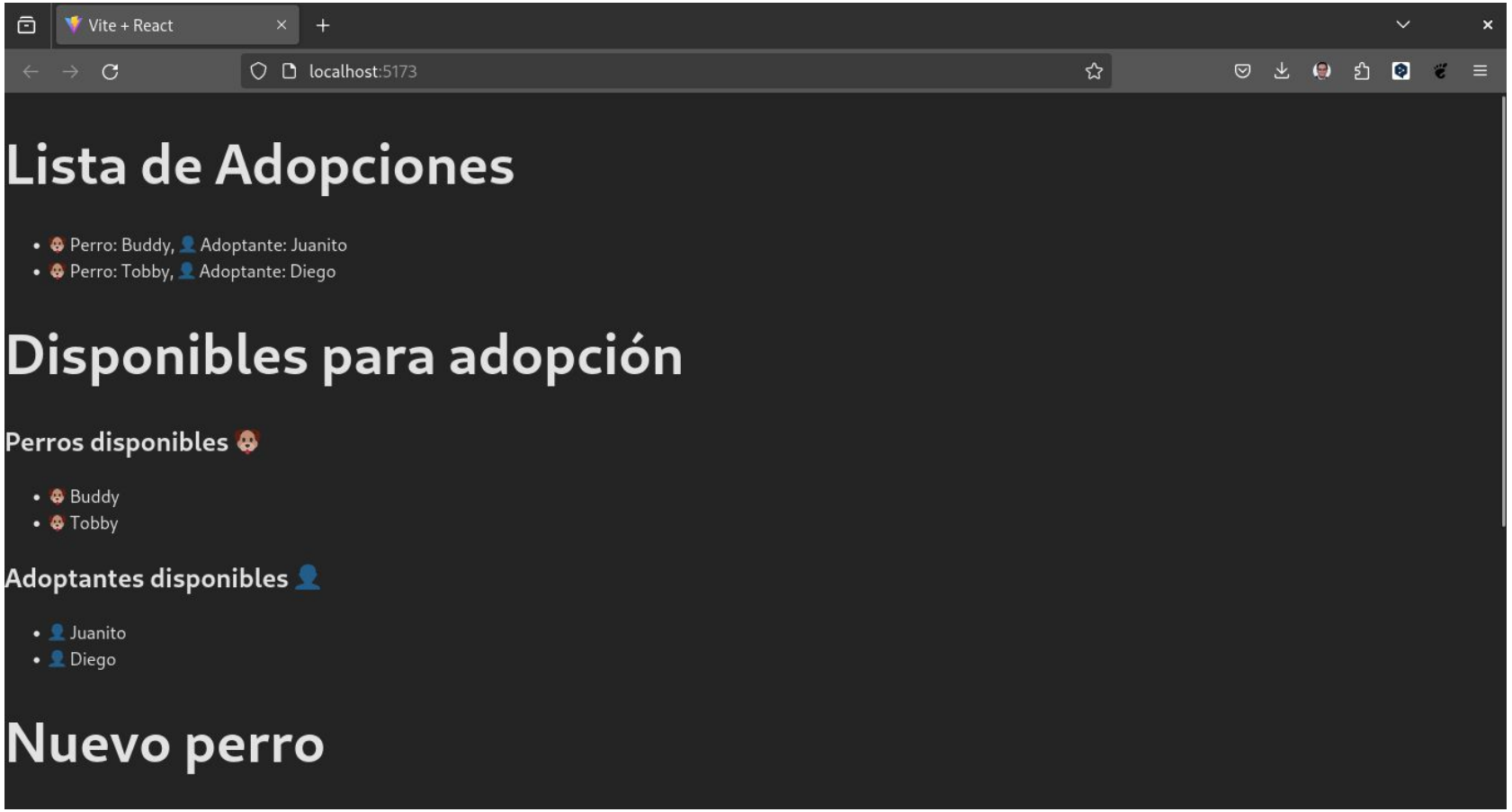
Segím Llamuca et al (2021) "Es un enfoque para el desarrollo de una aplicación única como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y mecanismos ligeros de comunicación, a menudo un recurso de una interfaz de programación de aplicaciones (API) sobre protocolo de transferencia de hipertexto (HTTP). Estos servicios están construidos alrededor de las capacidades del negocio y con independencia de despliegue e implementación totalmente automatizada. Existe un mínimo de gestión centralizada de estos servicios, los que pueden estar escritos en lenguajes de programación diferentes y utilizar diferentes tecnologías de almacenamiento de datos"

Arquitectura orientada a Microservicios.

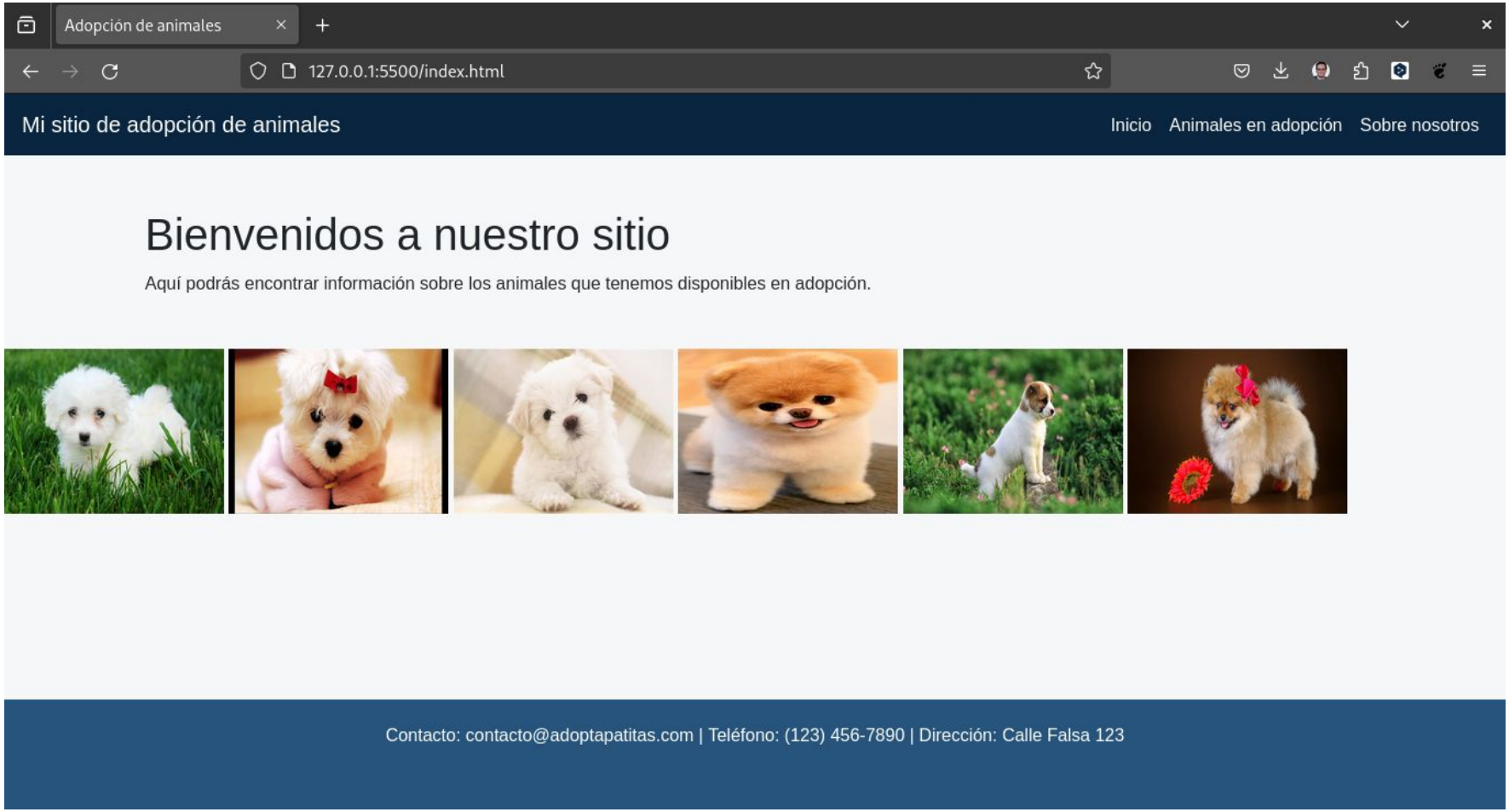
Una de las ventajas de utilizar microservicios es la capacidad de publicar una aplicación grande como un conjunto de pequeñas aplicaciones (microservicios) que se pueden desarrollar, desplegar, escalar, manejar y visualizar de forma independiente.

Los microservicios permiten a las empresas gestionar las aplicaciones de código base grande usando una metodología más práctica donde las mejoras incrementales son ejecutadas por pequeños equipos en bases de código y despliegues independientes. La agilidad, reducción de costes y la escalabilidad granular, traen algunos retos de los sistemas distribuidos y las prácticas de gestión de los equipos de desarrollo que deben ser considerados

Ejemplo


Vite + React
localhost:5173
Lista de Adopciones
Perro: Buddy, Adoptante: Juanito
Perro: Tobby, Adoptante: Diego
Disponibles para adopción
Perros disponibles
Buddy
Tobby
Adoptantes disponibles
Juanito
Diego
Nuevo perro

Ejemplo


Adopción de animales
127.0.0.1:5500/index.html
Mi sitio de adopción de animales
Inicio
Animales en adopción
Sobre nosotros
Bienvenidos a nuestro sitio
Aquí podrás encontrar información sobre los animales que tenemos disponibles en adopción.
Contacto: contacto@adoptapatitas.com | Teléfono: (123) 456-7890 | Dirección: Calle Falsa 123

Bibliografía

Acosta Gonzaga, E., Álvarez Cedillo, J. A., & Gordillo Mejía, A. (2006). Arquitecturas en n-Capas: Un Sistema Adaptivo. Polibits, Recuperado de aquí

CTES. (2023). Tipos de aplicaciones web. CTES: Revista de Ciencia, Tecnología y Sociedad, 8(1), 1-10. Recuperado de aquí.

Hernández Berrones, E. A. (2020). Desarrollo de una aplicación web con el Framework Bootstrap y el precompilador Sass para la gestión de pedidos de productos agrícolas de la Empresa El Chagra. Recuperado de aquí

Llamuca-Quinaloa, J., Vera-Vincent, Y., & Tapia-Cerda, V. (2021). Análisis comparativo para medir la eficiencia de desempeño entre una aplicación web tradicional y una aplicación web progresiva. TecnoLógicas, [Vol. 24(49)], pp. 13-30. Recuperado de aquí

Bibliografía

Soto Eras, W. M. (2021). Desarrollo del portal web de la Fundación Nuestra Señora del Cisne para la gestión de servicios en el Cantón Durán. Recuperado de aquí

Velasco, E., Alonso, O. K., & Pereda Herrero, V. (2022). Diseño y validación de un modelo de análisis de sitios web. Logos (La Serena). Recuperado de aquí