



Desarrollo de Aplicaciones Web

Tema 1 Desarrollo Web Cliente-Servidor



Carrera en Línea
Tecnologías de la Información



1.1. La importancia del desarrollo Web en las organizaciones

El desarrollo de aplicaciones web es el proceso de construir sitios web o aplicaciones web para resolver un problema de procesamiento de datos o los requerimientos de software de una empresa. Estos sitios web serán ejecutados en el Internet (Word Wide Web) o en una intranet. El desarrollo Web va desde simples páginas web estáticas, pasando por aplicaciones web complejas, incluyendo comercio electrónico y redes sociales.

En la actualidad, el desarrollo de aplicaciones web ha evolucionado y poco a poco ha ido remplazando a las aplicaciones de escritorio, tanto de modo texto (ya casi en la obsolescencia o inexistentes) como de modo gráfico, por sus múltiples ventajas tecnológicas. Por ejemplo, las aplicaciones web tienen una gran ventaja frente a las aplicaciones distribuidas. Cuando se desarrolla una aplicación distribuida de escritorio o de consola, se debe programar, inclusive, las comunicaciones, utilizando componentes de programación llamados sockets (capas de red y de transporte). Adicionalmente, el programador debe analizar la implementación de su propio protocolo de comunicaciones a nivel de su aplicación, por ejemplo, que hacer luego de recibir un mensaje, que hacer cuando la red de datos falla, como enviar un acknowledgement luego de la ejecución de un proceso ejecutado satisfactoriamente o de que ese proceso haya fallado.

Para las organizaciones y los negocios, el desarrollo de sitios web se lo puede ver desde dos puntos de vista, el diseño de contenido (CMS-Content Management System) o la implementación de sistemas completos orientados a la web (Internet o intranet) y contratan a una empresa que provea esos servicios de desarrollo de software. Las grandes organizaciones por otro lado, consideran tener un equipo de desarrolladores web interno y seguir metodologías de desarrollo ágil, como SCRUM por ejemplo para administrar el proceso de desarrollo y su recurso humano.



1.1. La importancia del desarrollo Web en las organizaciones

Otro grupo de profesionales, necesario para cumplir con su objetivo son los diseñadores gráficos y de contenido, quienes no cumplen la tarea de implementación de las aplicaciones y de las páginas web, sino más bien, su tarea es el diseño de las interfaces de usuario para que estas sean amigables, pero más que nada, usables. En este mundo del desarrollo, no es únicamente importante conocer o dominar las tecnologías de desarrollo básicas como son HTML, JavaScript y CSS, o tecnologías avanzadas como nodejs, React, Vuejs, entre muchas otras. También es importante conocer lo que el usuario desea y como comunicar la información al usuario a través de la aplicación Web, esto es crucial para el uso de la aplicación, pues en la mayoría de casos, por no decir en todos, el usuario estará solo frente a la computadora cuando use el sistema. Todo deberá ser intuitivo.

Por otro lado, y para evitar que el web y su desarrollo sea desordenado y evitar de alguna manera la anarquía tecnológica, aparece desde los inicios del web el “World Wide Web Consortium” (W3C) el cual desde 1994 es liderado por Tim Berners-Lee, provee estándares para el World Wide Web. El W3C dirige el camino de las tecnologías principales del web, y mantiene constante comunicación con la industria y los líderes investigación. La visión del W3C es una web que incluya participación, conocimiento compartido, confiabilidad. Ofrece una Web de interacción rica, un Web de datos y servicios, una Web confiable.



1.1. La importancia del desarrollo Web en las organizaciones

El desarrollo de aplicaciones web es el proceso de construir sitios web o aplicaciones web para resolver un problema de procesamiento de datos o los requerimientos de software de una empresa. Estos sitios web serán ejecutados en el Internet (Word Wide Web) o en una intranet. El desarrollo Web va desde simples páginas web estáticas, pasando por aplicaciones web complejas, incluyendo comercio electrónico y redes sociales.

En la actualidad, el desarrollo de aplicaciones web ha evolucionado y poco a poco ha ido remplazando a las aplicaciones de escritorio, tanto de modo texto (ya casi en la obsolescencia o inexistentes) como de modo gráfico, por sus múltiples ventajas tecnológicas. Por ejemplo, las aplicaciones web tienen una gran ventaja frente a las aplicaciones distribuidas. Cuando se desarrolla una aplicación distribuida de escritorio o de consola, se debe programar, inclusive, las comunicaciones, utilizando componentes de programación llamados sockets (capas de red y de transporte). Adicionalmente, el programador debe analizar la implementación de su propio protocolo de comunicaciones a nivel de su aplicación, por ejemplo, que hacer luego de recibir un mensaje, que hacer cuando la red de datos falla, como enviar un acknowledgement luego de la ejecución de un proceso ejecutado satisfactoriamente o de que ese proceso haya fallado.

Para las organizaciones y los negocios, el desarrollo de sitios web se lo puede ver desde dos puntos de vista, el diseño de contenido (CMS-Content Management System) o la implementación de sistemas completos orientados a la web (Internet o intranet) y contratan a una empresa que provea esos servicios de desarrollo de software. Las grandes organizaciones por otro lado, consideran tener un equipo de desarrolladores web interno y seguir metodologías de desarrollo ágil, como SCRUM por ejemplo para administrar el proceso de desarrollo y su recurso humano.



1.2. Introducción al desarrollo de aplicaciones Web



Al igual que todo tipo de aplicaciones de software, ya sean estas de escritorio, móviles, embebidas, gráficas, de consola. El desarrollo de aplicaciones web debe seguir un conjunto de pasos agrupados en una metodología y se deben implementar en plataformas de desarrollo robustas y seguras, y que además, permitan un desarrollo rápido. Una de las metodologías más usadas en la actualidad es SCRUM, pero no es la única solución a seguir. El desarrollador deberá estar en capacidad de aplicar el método apropiado de acuerdo los requerimientos y los recursos a su disposición. El ciclo de vida debe respetarse ya sea de manera secuencial (Waterfall) o iterativa (Ágil).

1.2.1. El proceso de desarrollo Web

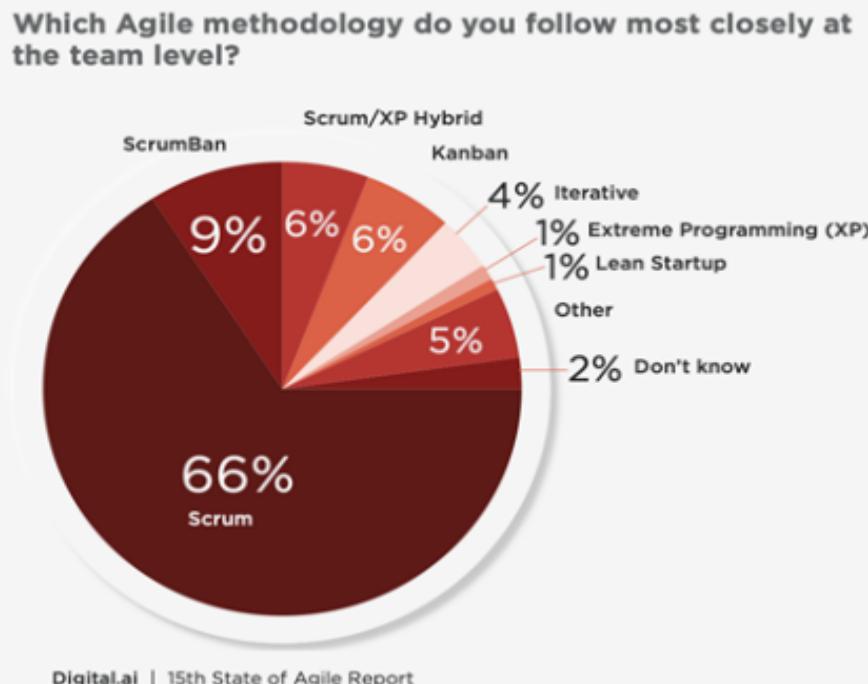


Figura 1. Most Followed Agile Methodology

Fuente: <https://digital.ai/resource-center/analyst-reports/state-of-agile-report>

Independiente de la arquitectura y las tecnologías a ser usadas para la implementación de una aplicación Web, los desarrolladores deben seguir un conjunto de pasos para la obtención de un producto final funcional y de calidad. Esencialmente, existen dos tendencias de ciclos de vida de desarrollo de software (SDLC), el método tradicional o de cascada (Sekgweleo, 2018) y el método Ágil (Lascano, 2009), de los cuales, el más representativo y usado es Scrum. Las etapas básicas se mantienen, pero se ejecutan en diferente orden y con un enfoque distinto para cada metodología. En ambos casos, se da inicio al proceso con uno o más requerimientos de procesamiento, un análisis de los mismos, para posteriormente realizar un diseño de la estructura de los datos y de las interfaces de usuario, también se debe realizar un diseño arquitectónico de la solución, luego se procede a la construcción y pruebas de unidad (funcionalidad), de usabilidad, las validaciones y verificaciones correspondientes, para luego dar inicio a la puesta en marcha (deployment) de la aplicación. Estos pasos, en la actualidad se adaptan más a un desarrollo iterativo por features (componentes/módulos), globalmente, una de las metodologías ágiles más usadas por las empresas de desarrollo es Scrum (15th Annual State Of Agile Report | Digital.Ai, s. f.), Figura 1.



1.2.1. El proceso de desarrollo Web

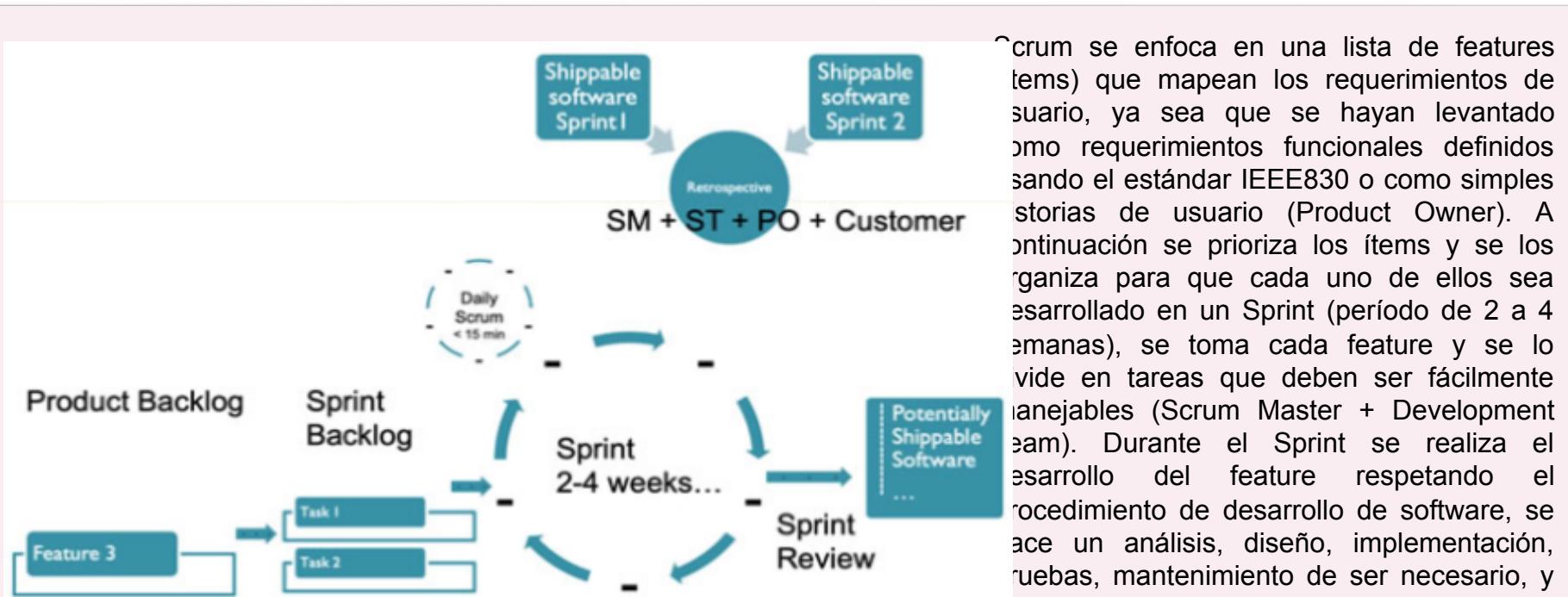


Figura 2. El proceso Scrum

Scrum se enfoca en una lista de features (items) que mapean los requerimientos de usuario, ya sea que se hayan levantado como requerimientos funcionales definidos usando el estándar IEEE830 o como simples historias de usuario (Product Owner). A continuación se prioriza los ítems y se los organiza para que cada uno de ellos sea desarrollado en un Sprint (período de 2 a 4 semanas), se toma cada feature y se lo divide en tareas que deben ser fácilmente manejables (Scrum Master + Development Team). Durante el Sprint se realiza el desarrollo del feature respetando el procedimiento de desarrollo de software, se hace un análisis, diseño, implementación, pruebas, mantenimiento de ser necesario, y se obtiene un software entregable a ser revisado y recibido por el cliente. Se repite el proceso (Sprint) para cada feature, y se va integrando cada componente al final de cada iteración, se revisa el procedimiento a través de un Sprint Review y se continua.

1.2.1. El proceso de desarrollo Web

Para la administración del proceso, se puede usar herramientas como Trello (<https://www.trello.com>) y para el versionamiento de los entregables, principalmente para el código, se usa Sistemas de Control de Versionamiento (VCS). El más usado en la actualidad es Git, una de sus implementaciones es GitHub (<https://github.com/>), que ofrece cuentas académicas sin costo alguno con todas sus funcionalidades al alcance de los estudiantes.

El desarrollo de aplicaciones web se divide principalmente en el desarrollo del back-end (procesos + datos) y en el desarrollo del front-end (páginas web, formularios). Para desarrollar el back-end, existen varias posibilidades, una de ellas es la generación de aplicaciones tipo CGI (Perl, Python, nodejs, ASP, PHP, JSP) y a través de ellos se puede acceder a datos en una base de datos, SQL o No-SQL. Otra opción para el desarrollo del back-end son los Servicios Web tipo REST, para los cuales se debe diseñar las URLs (más adelante se cubrirá este tópico), y el formato JSON de los datos.

Para el desarrollo del Front-End web, se debe bosquejar las páginas que formarán parte de la aplicación y su navegabilidad de página a página, pero más importante es la usabilidad del sitio completo.



1.2.1. El proceso de desarrollo Web

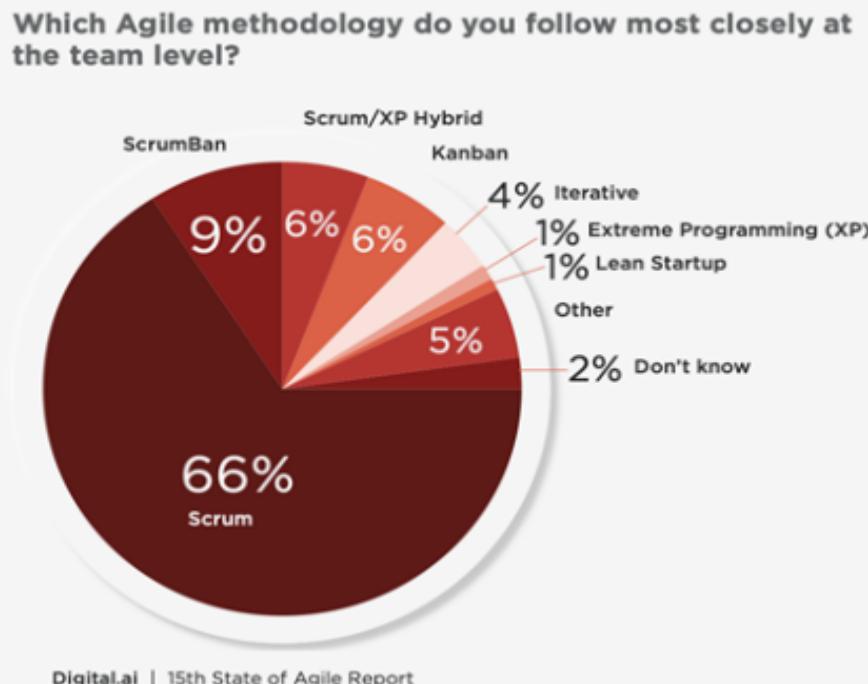


Figura 1. Most Followed Agile Methodology

Fuente: <https://digital.ai/resource-center/analyst-reports/state-of-agile-report>

Independiente de la arquitectura y las tecnologías a ser usadas para la implementación de una aplicación Web, los desarrolladores deben seguir un conjunto de pasos para la obtención de un producto final funcional y de calidad. Esencialmente, existen dos tendencias de ciclos de vida de desarrollo de software (SDLC), el método tradicional o de cascada (Sekgweleo, 2018) y el método Ágil (Lascano, 2009), de los cuales, el más representativo y usado es Scrum. Las etapas básicas se mantienen, pero se ejecutan en diferente orden y con un enfoque distinto para cada metodología. En ambos casos, se da inicio al proceso con uno o más requerimientos de procesamiento, un análisis de los mismos, para posteriormente realizar un diseño de la estructura de los datos y de las interfaces de usuario, también se debe realizar un diseño arquitectónico de la solución, luego se procede a la construcción y pruebas de unidad (funcionalidad), de usabilidad, las validaciones y verificaciones correspondientes, para luego dar inicio a la puesta en marcha (deployment) de la aplicación. Estos pasos, en la actualidad se adaptan más a un desarrollo iterativo por features (componentes/módulos), globalmente, una de las metodologías ágiles más usadas por las empresas de desarrollo es Scrum (15th Annual State Of Agile Report | Digital.Ai, s. f.), Figura 1.

1.2.2. Tipos de aplicaciones web

Se considera una aplicación a aquella que puede ser vista a través de un navegador web. Actualmente existen navegadores web para todo dispositivo con conexión a Internet, por ejemplo: Computadoras personales, laptops, teléfonos inteligentes, medios de transporte, electrodomésticos. Vamos a mencionar nueve tipos de aplicaciones web (9 Types of Web Applications to Simplify Your Business [+Examples], s. f.), de las cuales se presenta un resumen a continuación, definiendo sus características, y usos: (El término del tipo de aplicación se mantendrá en Inglés): Static Web Apps, Dynamic Web Apps, Single Page Apps, Multiple Page Apps, Animated Web Apps, Content Management System, E-Commerce Apps, Portal Web Apps, Progressive Web Apps.

Static Web Apps (Web pages). Entregan directamente el contenido al usuario final, pero sin recuperar datos de una base de datos, por lo general contienen texto, imágenes, animaciones gif, banners para atraer a sus visitantes, para su creación se utiliza HTML, CSS, JavaScript. Se las usa para presentaciones de empresas, para libros.

Dynamic Web Applications. Es una aplicación que genera datos en tiempo real, de acuerdo a los pedidos del usuario, la aplicación generará una respuesta del servidor, el contenido se genera dinámicamente, por ejemplo: Facebook. Sus principales usos: industria TI, salud, transporte, logística, para su creación se utiliza, además de las tecnologías anteriormente mencionadas. Angular, React, nodejs, Java, PHP.

Los tipos de aplicaciones que se presentan a continuación, son diferentes versiones de aplicaciones Web Dinámicas, y como tal se las puede desarrollar con la misma tecnología en su forma básica. Pero también se cuenta con muchas más opciones.

1.2.2. Tipos de aplicaciones web

Single Page Applications. Corren completamente en un navegador (browser) y no requieren que su página se recargue, se debe manejar muy bien la lógica de presentación, por ejemplo una aplicación de correo electrónico, PayPal. Se implementan por lo general siguiendo el concepto de AJAX.

Multiple Page Applications. Esa es una aplicaciones que se compone de múltiples páginas y estas se recargan cada que el usuario navega a un enlace diferente. Por ejemplo Amazon, cuando se realiza una búsqueda de productos para comprar, y se hace click en uno de esos productos. Amazon lleva al usuario a otra página. Su uso se da principalmente en el comercio electrónico, su mayor ventaja es que su indexación en los motores de búsqueda es optimizada. Además, permite a los usuarios la visita individual de cada página a través de sus enlaces.

Animated Web Applications. Es una aplicación web que soporta animación y sincronización en la plataforma web. Por ejemplo: www.squadeeasy.com , www.mikimottes.com . sus principales usos son en el área de animación, educación y juegos. Su desarrollo se complementa con el uso de HTML5, Flash y SVG.

Content Management System. Este software ayuda al usuario común y corriente manejar contenido digital. A la mejora de la administración y la producción de contenido se le conoce como CMS (Content Management System). Un CMS ayuda crear, modificar y administrar contenido sin necesidad de conocimiento técnico de aplicaciones web. No hace falta código ni HTML. Muy útil para blogs, márgenes y portales de noticias. Ejemplos de CMS: WordPress, Drupal, Joomla.

1.2.2. Tipos de aplicaciones web

E-Commerce Applications. Estas aplicaciones ayudan a vender o comprar bienes/mercadería utilizando el Internet, es esencial la integración de las transacciones con su pago, por ejemplo, Amazon, eBay. Algunas tecnologías en las que se basa el e-commerce son: dinero electrónico, sistemas de inventario, comercio móvil, Internet Market, gestión de cadenas de suministro.

Portals. Son aplicaciones web que integran diferentes fuentes para cumplir con su objetivo, por ejemplo, correo, foros, motores de búsqueda. Portales más avanzados se desarrollan bajo el concepto de Mashups, combinación de lógica de negocio propia del desarrollador y funcionalidad de proveedores de servicios (APIs) públicos. Las tecnologías para desarrollar APIs son las usadas para programar el Back-end de las aplicaciones web: nodejs, Java, Python, etc. Las tecnologías para programar el consumo de las APIs son las usadas para Front-end: React, HTML, JavaScript, Angular, vue.js, etc. Se los encuentra en portales del gobierno, portales de instituciones educativas. Udemy (www.udemy.com), Corsera (www.coursera.org), son ejemplos de portales.

Progressive Web Applications. Conocidas como aplicaciones Web cross-platform que usan las APIs (Application Programming Interface) de browsers de última generación. Su principal característica es que ofrecen una experiencia similar a la de una aplicación móvil nativa. Se construyen con tecnologías web estándar como HTML, CSS, y JavaScript. Una de las principales razones para escoger una aplicación web progresiva es que mejora la velocidad y la adaptabilidad, puede funcionar on-line y off-line. Entonces, inclusive con una conexión de Internet mala resulta fácil acceder a la información. Por ejemplo, Starbucks, BMW, Spotify. Sus principales usos: procesos bajo demanda, venta al por menor, comercio electrónico, transporte y logística, social media, salud, industria TI.

1.2.2. Tipos de aplicaciones web

Se considera una aplicación a aquella que puede ser vista a través de un navegador web. Actualmente existen navegadores web para todo dispositivo con conexión a Internet, por ejemplo: Computadoras personales, laptops, teléfonos inteligentes, medios de transporte, electrodomésticos. Vamos a mencionar nueve tipos de aplicaciones web (9 Types of Web Applications to Simplify Your Business [+Examples], s. f.), de las cuales se presenta un resumen a continuación, definiendo sus características, y usos: (El término del tipo de aplicación se mantendrá en Inglés): Static Web Apps, Dynamic Web Apps, Single Page Apps, Multiple Page Apps, Animated Web Apps, Content Management System, E-Commerce Apps, Portal Web Apps, Progressive Web Apps.

Static Web Apps (Web pages). Entregan directamente el contenido al usuario final, pero sin recuperar datos de una base de datos, por lo general contienen texto, imágenes, animaciones gif, banners para atraer a sus visitantes, para su creación se utiliza HTML, CSS, JavaScript. Se las usa para presentaciones de empresas, para libros.

Dynamic Web Applications. Es una aplicación que genera datos en tiempo real, de acuerdo a los pedidos del usuario, la aplicación generará una respuesta del servidor, el contenido se genera dinámicamente, por ejemplo: Facebook. Sus principales usos: industria TI, salud, transporte, logística, para su creación se utiliza, además de las tecnologías anteriormente mencionadas. Angular, React, nodejs, Java, PHP.

Los tipos de aplicaciones que se presentan a continuación, son diferentes versiones de aplicaciones Web Dinámicas, y como tal se las puede desarrollar con la misma tecnología en su forma básica. Pero también se cuenta con muchas más opciones.

1.2.3. Web móvil

El desarrollo de aplicaciones web móviles ha evolucionado a la par de la evolución de la circuitería que ha permitido cada vez tener teléfonos celulares más pequeños, pero sobre todo más poderosos, en los cuales se puede instalar suites de aplicaciones completas, como por ejemplo office, y muchas más. En la actualidad, en los celulares, y en casi cualquier dispositivo se puede instalar un navegador de Internet (browser).

Las tecnologías móviles en sus inicios permiten el acceso a datos del Internet a través de un teléfono utilizando WAP (Wireless Access Protocol) que es introducido en 1999, y es ampliamente usado de los 2000, pero deja de ser necesario en los 2010, cuando los dispositivos móviles inician su soporte a protocolos modernos tales como http. Las principales tecnologías de la época fueron WML (Wireless Markup Language) y XHTML-MP (XHTML Mobile Profile). Sistemas operativos móviles como Android y iOS dominan actualmente el mercado, y permiten el desarrollo de apps nativas, pero también proveen diferentes browsers con soporte reducido, pero completo hacia HTML, CSS y JavaScript.



1.2.3. Web móvil

Sin embargo, y a pesar del poder de los celulares y su soporte de browsers estándares en el mercado. Los diseñadores de aplicaciones Web para móviles deben considerar varios aspectos, por ejemplo los fisiológicos, los dispositivos de entrada de datos (pantalla táctil), el tamaño de la pantalla. Entonces, las páginas deben ser automáticamente ajustadas para que el usuario pueda ver la misma información en un celular o en un browser de escritorio, y que también su interacción sea placentera y útil, provocando que la aplicación web asegure usabilidad y satisfacción. Este diseño que apunta que una página web se renderice de buena forma independiente del dispositivo se conoce como Diseño Web Responsivo. Esto se logra usando CSS media queries.

En conclusión el diseño web móvil depende de ciertas limitaciones como el tamaño de la pantalla, el medio de ingreso de datos el soporte de browsers que el dispositivo tenga. Pero como no se puede diseñar una página por dispositivo, se debe usar el concepto de diseño responsive y de diseño adaptativo. El back-end para aplicaciones web móviles es el mismo que para aplicaciones web estándar, puede ser una tecnología CGI, o servicios REST, y los datos pueden estar en una base de datos local o remota.

1.2.3. Web móvil

El desarrollo de aplicaciones web móviles ha evolucionado a la par de la evolución de la circuitería que ha permitido cada vez tener teléfonos celulares más pequeños, pero sobre todo más poderosos, en los cuales se puede instalar suites de aplicaciones completas, como por ejemplo office, y muchas más. En la actualidad, en los celulares, y en casi cualquier dispositivo se puede instalar un navegador de Internet (browser).

Las tecnologías móviles en sus inicios permiten el acceso a datos del Internet a través de un teléfono utilizando WAP (Wireless Access Protocol) que es introducido en 1999, y es ampliamente usado de los 2000, pero deja de ser necesario en los 2010, cuando los dispositivos móviles inician su soporte a protocolos modernos tales como http. Las principales tecnologías de la época fueron WML (Wireless Markup Language) y XHTML-MP (XHTML Mobile Profile). Sistemas operativos móviles como Android y iOS dominan actualmente el mercado, y permiten el desarrollo de apps nativas, pero también proveen diferentes browsers con soporte reducido, pero completo hacia HTML, CSS y JavaScript.



1.2.4. Progressive Web Applications

“Las opciones como PWA no son adecuadas para las nuevas empresas. Es bastante eficiente si una empresa estable y el propietario del producto sabe quiénes son sus usuarios finales y qué tipo de experiencia esperan (por ejemplo, la mayoría de ellos son usuarios de Android). Sin embargo, el soporte de aplicaciones web progresivas no es tan extenso. Recientemente, Firefox (que aún tiene una participación del 6,3% en el mercado estadounidense) dejó de aceptar PWA, lo que demuestra que este tipo de arquitectura web aún es inestable.” (Luchaninov, s. f.) Sergey Rykov, JavaScript Team Leader en MobiDev

En 2017 Twitter decide implementar su aplicación para el cliente como una Progressive Web Application, con esto logran mejorar sus tiempos de carga, rebajar el consumo de datos, y mejorar el compromiso del usuario con el sistema. Esto se traduce en un incremento de Tweets en un 75%. Todo eso lo lograron creando Twitter Lite, la PWA de Twitter. Otra ventaja aún mayor fue que el rendimiento de la PWA fue mejor que el de su aplicación móvil nativa, y apenas usando el 3% de almacenamiento en comparación con la App de Twitter.

1.2.4. Progressive Web Applications

En pocas palabras las PWA reducen la brecha diferencial entre las aplicaciones web móviles y las aplicaciones nativas. Hace un poco más de una década con la llegada disruptiva del iPhone, todo se mudo del web a las apps, situación similar sucedió con la disruptión de las aplicaciones web con respecto a las de escritorio. Tecnologías que han avanzado para permitir este rápido cambio son HTML, CSS y JavaScript. Igualmente lo han hecho los browsers que han añadido service workers, web manifestations y el uso de APIs modernas, los cuales se han convertido en requerimientos principales de las PWA.

Sin embargo, su implementación va más allá del uso de tecnología modernos, en sí es un cambio de enfoque, de paradigma para el desarrollo web. Entre las cosas que se debería aprender para implementar PWA están los service workers de cache, las push notifications, y el background synchronization. Un libro interesante disponible en la base de datos ProQuest Ebook Central que está disponible digitalmente a través de la biblioteca de la Universidad es (Progressive Web Application Development by Example, s. f.). Se lo puede descargar desde mieSPE en la URL <https://ebookcentral.proquest.com/lib/espeec/detail.action?docID=5477666>.



1.2.4. Progressive Web Applications

“Las opciones como PWA no son adecuadas para las nuevas empresas. Es bastante eficiente si una empresa estable y el propietario del producto sabe quiénes son sus usuarios finales y qué tipo de experiencia esperan (por ejemplo, la mayoría de ellos son usuarios de Android). Sin embargo, el soporte de aplicaciones web progresivas no es tan extenso. Recientemente, Firefox (que aún tiene una participación del 6,3% en el mercado estadounidense) dejó de aceptar PWA, lo que demuestra que este tipo de arquitectura web aún es inestable.” (Luchaninov, s. f.) Sergey Rykov, JavaScript Team Leader en MobiDev

En 2017 Twitter decide implementar su aplicación para el cliente como una Progressive Web Application, con esto logran mejorar sus tiempos de carga, rebajar el consumo de datos, y mejorar el compromiso del usuario con el sistema. Esto se traduce en un incremento de Tweets en un 75%. Todo eso lo lograron creando Twitter Lite, la PWA de Twitter. Otra ventaja aún mayor fue que el rendimiento de la PWA fue mejor que el de su aplicación móvil nativa, y apenas usando el 3% de almacenamiento en comparación con la App de Twitter.

1.3. Arquitectura de Aplicaciones Web

Una aplicación web es una aplicación Cliente-Servidor donde hay un browser (el cliente) y un servidor web. La lógica del sitio esta distribuida entre el cliente y el servidor, y la información esta almacenada en una base de datos, ya sea local, en otra computadora en la red, o en la nube o el Internet. Existen diferentes formas de organizar el Front-End, el Back-End y los datos de un sistema para la Web, esta estructura y sus relaciones componen la arquitectura de la aplicación. La arquitectura define como opera, se desempeña y escala la aplicación. Una aplicación web se caracteriza por su interacción, integración, y autenticación (Luchaninov, s. f.).



1.3.1. Client-Server

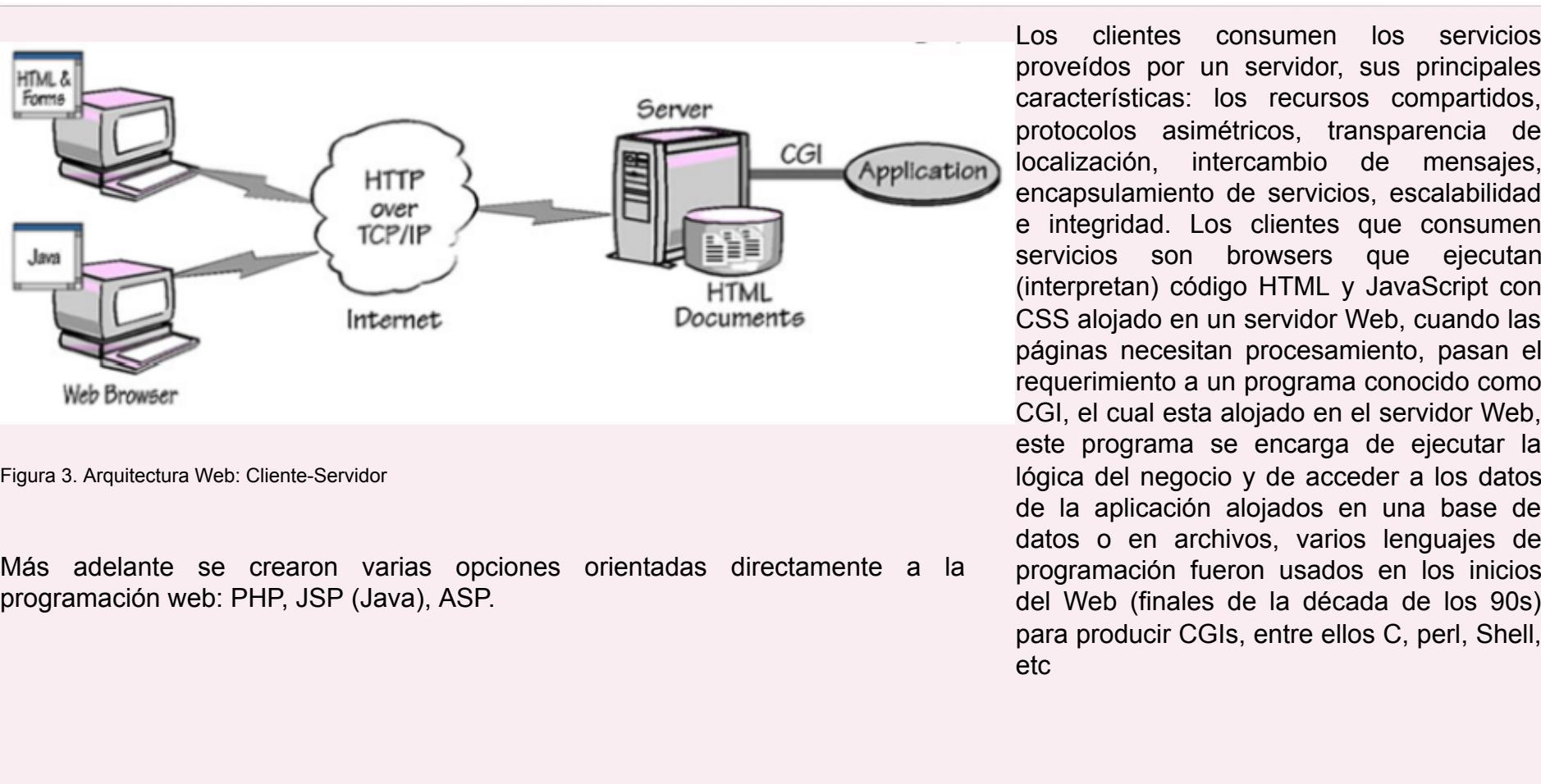


Figura 3. Arquitectura Web: Cliente-Servidor

Más adelante se crearon varias opciones orientadas directamente a la programación web: PHP, JSP (Java), ASP.

1.3.2. Tier-architecture

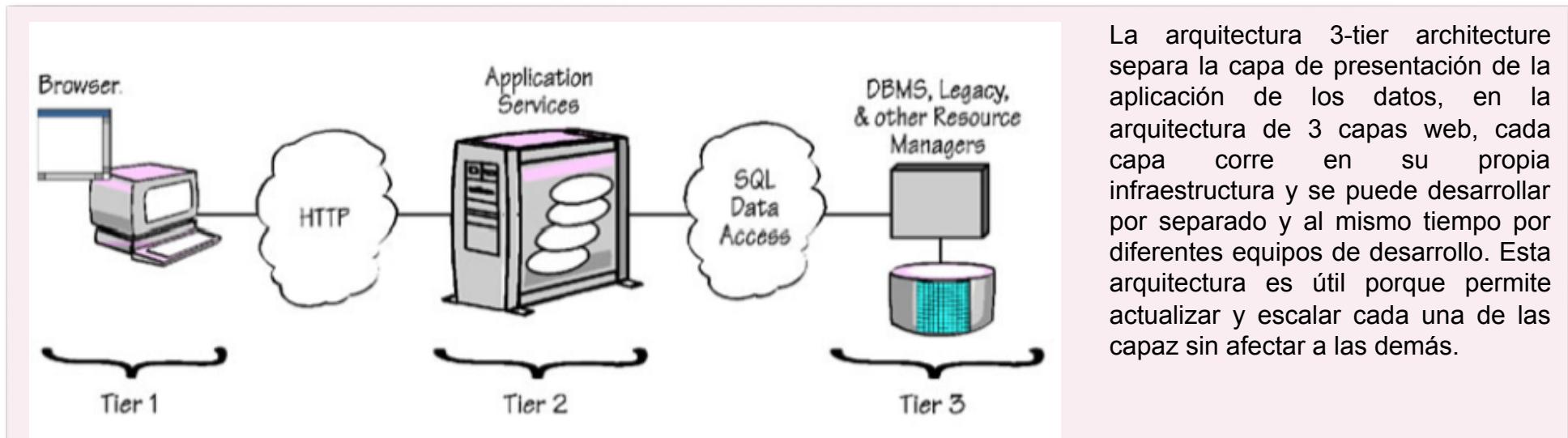


Figura 4. 3-tier architecture

A continuación se 4 arquitecturas modernas que describen en detalle las necesidades del negocio

1.3.3. Server-side rendering

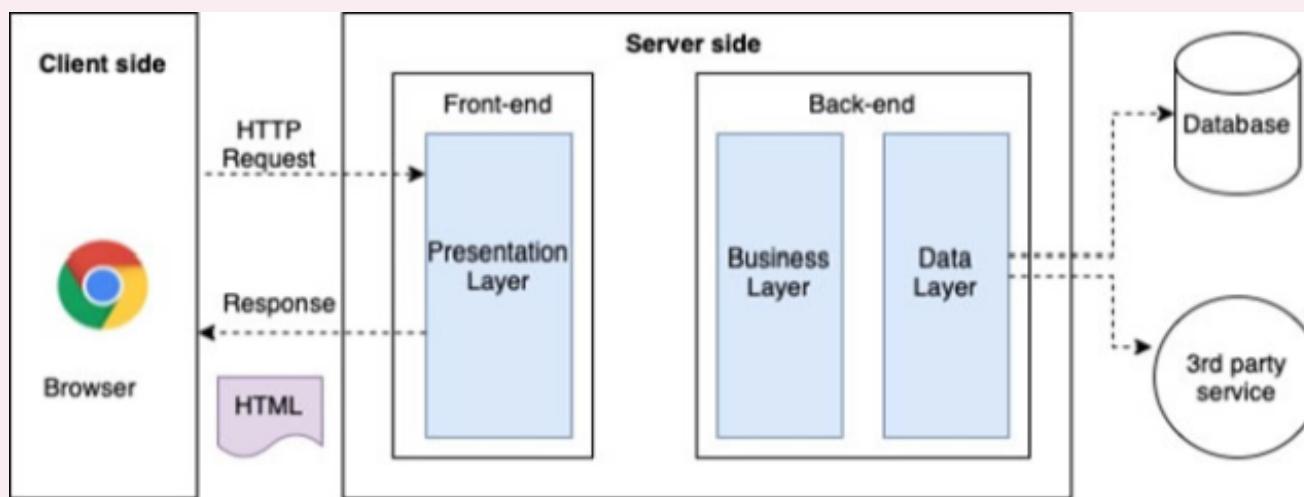


Figura 5. Server side rendering

Esta arquitectura ayuda para una correcta indexación de las páginas (HTML) generadas, se monta el archivo HTML antes de que sea enviado al browser, en otras palabras el browser no recibe o procesa JavaScript, no ofrecen mucha interacción pero son más rápidas y resultan en una mejor experiencia de usuario por ejemplo, las redes sociales.

Los principios básicos de la arquitectura cliente-servidor establecen que un cliente requiere contenido del servidor, en donde existe un servidor que contiene las reglas de negocio y una base de datos, una página estática usando JavaScript solicita un pedido a un servicio (muy probablemente una API), el servicio responde al pedido, entregando los datos y los despliega como página Web al cliente. La página HTML es renderizada en el lado del servidor.

1.3.4. Static site generation

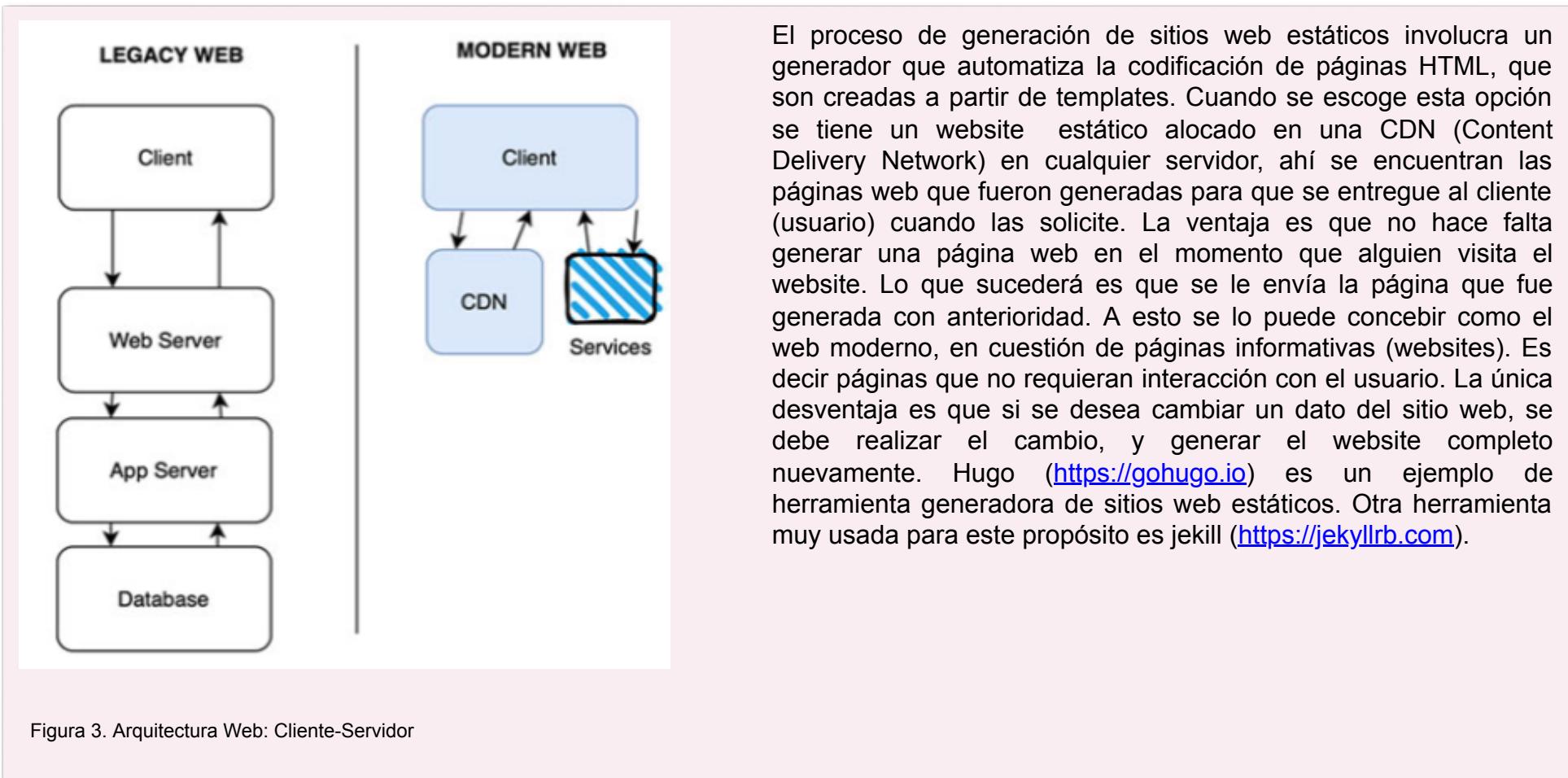


Figura 3. Arquitectura Web: Cliente-Servidor

1.3.5. Micro Front-End

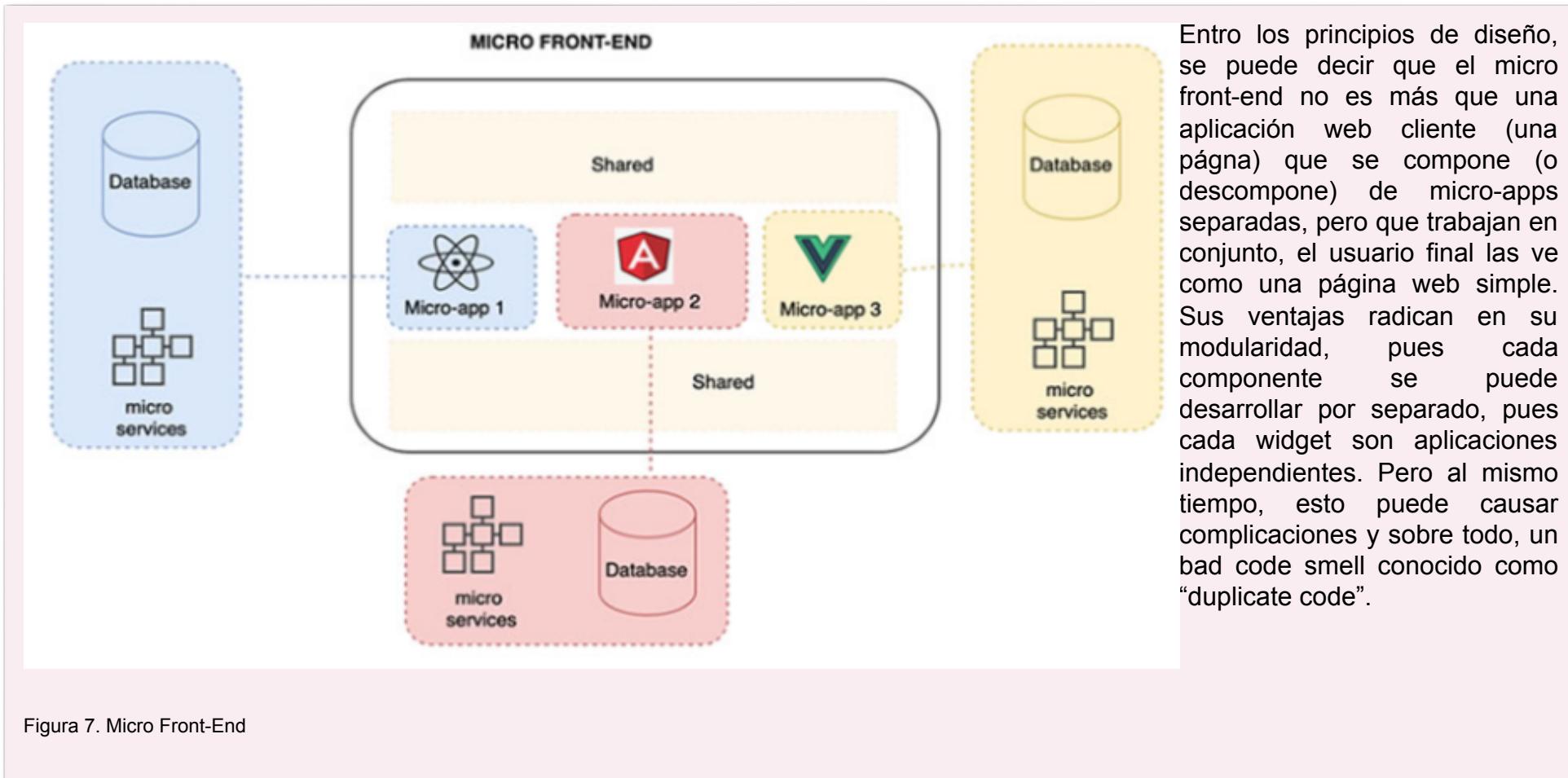


Figura 7. Micro Front-End

1.3.6. Cloud Architecture



Por lo general las aplicaciones web, independiente de la tecnología utilizada para su desarrollo deben publicarse en un servidor web que por lo general es provisto por un servicio de hosting. En el caso de la nube la arquitectura de la nube, esto implica que los servidores serán administrados por un proveedor de servicios en la Nube (IaaS o PaaS), por ejemplo Amazon AWS, Microsoft Azure, Google Cloud u otros. En el caso de requerir IaaS, el desarrollador estará en la obligación de instalar las aplicaciones necesarias para que la aplicación se pueda poner en producción. Si el desarrollador, no desea perder el tiempo configurando el servidor web, o el servidor de bases de datos, entonces solicitará PaaS (Platform as a Service), de esta manera la computadora virtual estará lista para desplegar la aplicación web. La instalación y configuración corre por parte del proveedor del servicio.

1.3.7. Selección de la arquitectura apropiada

La selección de la arquitectura para la aplicación a desarrollarse dependerá de varios aspectos, tanto técnicos como de usuario. Por ejemplo, necesita una aplicación que almacene datos y, que en general, solo realice las cuatro operaciones básicas (CRUD), una arquitectura cliente-servidor será más que suficiente. Si la aplicación exige la implementación de varias reglas de negocio, como por ejemplo cálculos de facturas, control de stock en la bodega, cómputo e promedios de alumnos, la mejor solución será la 3-tier architecture, y mucho si es una n-tier architecture/ para que cada capa se encargue de sus proceso correspondiente, implementando de esta manera una aplicación web modular en el back-end.

Si se necesita que el contenido de un website sea indexado, pero se tiene procesamiento de datos, una arquitectura híbrida podría ser la solución, una n-tier architecture para el procesamiento, pero se usaría un server-side rendering para mejorar la indexación del contenido generado.

Si por otro lado, lo que se necesita es velocidad de transmisión de los datos, y además los datos no cambian muy seguido en el tiempo (caso particular de los web sites), la solución sería seguir la arquitectura static site generation, siempre y cuando los datos sean informativos y no transaccionales.

Si desea mostrar en una sola página información dinámica proveniente de diferentes fuentes de datos como bases de datos o APIs propias o de terceros (públicas), la solución será Micro Front-End.

Finalmente, con respecto a la arquitectura en la nube, se daría en caso de necesidad del cliente. Los recursos que provee la nube son fácilmente asequibles, ya a costos razonables. Una de sus mayores ventajas es la alta disponibilidad, entonces en primera instancia, la decisión de optar por la nube de computación es más orientada a la puesta en marcha del proyecto o a la necesidad de contar en poco tiempo con ambientes de desarrollo listos para la codificación y Deployment.



1.4. Back-End para el Web

Como se dijo anteriormente, para desarrollar una aplicación web, es necesario diseñar y programar una o más páginas cliente para presentar y solicitar información para y del usuario (back-end), mientras que la lógica del negocio y el acceso a los datos se programa en el back-end. Básicamente, se debe tener el servidor web y el servidor de base de datos que pueden ser físicos (servidor), virtuales (VM) o estar en la nube (Instancias en la nube). Los servidores por lo general se encuentran en grandes granjas de servidores. Tanto el front-end como el back-end se deben publicar en un servidor web con acceso de entrada y salida al Internet, para que de esta manera sea visible para todos los usuarios. Sin embargo, se podría desarrollar una aplicación web para que únicamente sea accedida en la red local de la empresa, en este caso debe implementar firewalls para evitar accesos indebidos a la aplicación local de la empresa.

Si la aplicación web es desarrollada para ser pública, se la puede publicar en un servidor local en la empresa y se configura el servidor para que acepte pedidos a través del puerto 80 si es un http, y si se implementó un sitio seguro (es un estándar en la actualidad), entonces se debe permitir las entradas por el puerto 443, dependiendo de la tecnología usada para desarrollar la aplicación web, se debe instalar un software para servidor web en la máquina, por ejemplo si se desarrolló con Java, los servidores que soportan Java pueden ser Apache Tomcat, Glassfish, Payara. Si se usó PHP, se debe instalar Apache. En el caso de programar usando Node.js, no se debe instalar un servidor web, pues node.js cuenta con librerías, por ejemplo Express.js, que permite implementar un servidor web junto con el código de la aplicación, Figura 8.

```
1 var http = require('http');
2 var server = http.createServer(function(req, res) {
3     //handle incoming requests
4     if(req.url == '/') {
5         res.writeHead(200, { 'Content-Type': 'text/html'});
6     }
7 }
```

Como se dijo anteriormente, para desarrollar una aplicación web, es necesario diseñar y programar una o más páginas cliente para presentar y solicitar información para y del usuario (back-end), mientras que la lógica del negocio y el acceso a los datos se programa en el back-end. Básicamente, se debe tener el servidor web y el servidor de base de datos que pueden ser físicos (servidor), virtuales (VM) o estar en la nube (Instancias en la nube). Los servidores por lo general se encuentran en grandes granjas de servidores. Tanto el front-end como el back-end se deben publicar en un servidor web con acceso de entrada y salida al Internet, para que de esta manera sea visible para todos los usuarios. Sin embargo, se podría desarrollar una aplicación web para que únicamente sea accedida en la red local de la empresa, en este caso debe implementar firewalls para evitar accesos indebidos a la aplicación local de la empresa.

Si la aplicación web es desarrollada para ser pública, se la puede publicar en un servidor local en la empresa y se configura el servidor para que acepte pedidos a través del puerto 80 si es un http, y si se implementó un sitio seguro (es un estándar en la actualidad), entonces se debe permitir las entradas por el puerto 443, dependiendo de la tecnología usada para desarrollar la aplicación web, se debe instalar un software para servidor web en la máquina, por ejemplo si se desarrolló con Java, los servidores que soportan Java pueden ser Apache Tomcat, Glassfish, Payara. Si se usó PHP, se debe instalar Apache. En el caso de programar usando Node.js, no se debe instalar un servidor web, pues node.js cuenta con librerías, por ejemplo Express.js, que permite implementar un servidor web junto con el código de la aplicación, Figura 8.

```
1 var http = require('http');
2 var server = http.createServer(function(req, res) {
3     //handle incoming requests
4     if(req.url == '/') {
5         res.writeHead(200, { 'Content-Type': 'text/html'});
6
7         res.write('<html><body><h1> Jorge Edison Lascano Web Site</h1><hr/><p>');
8         res.write('Hello everybody from web root of Edison Lascano\'s web server');
9         res.write('</p><hr/>');
10        res.write('</body></html>');
11
12        res.end();
13    }
14    //another route (URL), or end point
15    else if (req.url == "/bye") {
16        res.writeHead(200, { 'Content-Type': 'text/html'});
17
18        res.write('Bye bye Future Engineers');
19
20        res.end();
21    }
22    else
23        res.end('Invalid request (The usual 404)');
24 });
25
26 port = 8081
27
28 server.listen(port);
29
30 console.log('Edison\'s Node.js web server is running on port --> ' + port );
```

Figura 1. un servidor web programado en node.js

Node.js se programa usando JavaScript, y para su ejecución se lo hace usando el intérprete node, Figura 9, que debió haber sido instalado anteriormente.

```
Edisons-MacBook-Pro:WS16nodejs edisonlascano$ node server.js
Edison's Node.js web server is running on port --> 8081
```

Figura 1. Ejecución de un programa en node.js

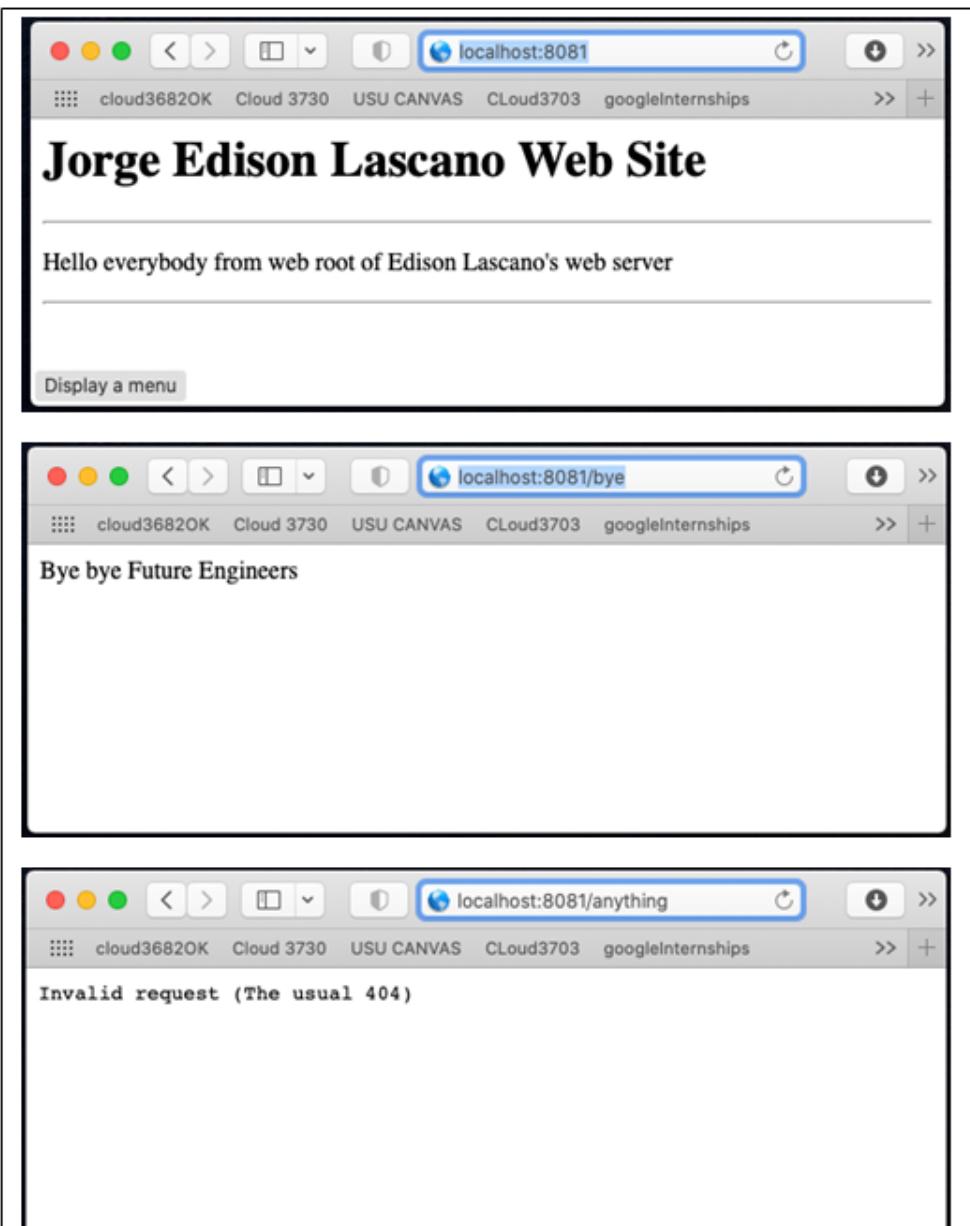


Figura 1. Visitando una página Web

Una vez levantado el servicio, se puede visitar el sitio web utilizando un browser (navegador de Internet), la URL se compone del nombre de la máquina donde se

ejecuta el servidor, por ejemplo `http://localhost` seguida de : y del número del puerto donde se levantó el servicio, y se complementa con el contenido de la variable `req.url`, (/ o /bye), las URLs completas serían `http://localhost:8081/` y `http://localhost:8081/bye` Figura 10. En caso de escribir cualquier otra URL, el servidor Web responderá con el mensaje “*Invalid request (The usual 404)*”.

1.4.1. Tecnologías Back-End

Diferentes tecnologías permiten el desarrollo de la lógica del negocio y la conexión a las bases de datos para aplicaciones Web. Alrededor de cada tecnología, existe no sólo un lenguaje de programación, sino también frameworks, bibliotecas, APIs.

Las tecnologías de back-end necesitan de un componente especial instalado en el servidor web seleccionado. En los inicios del web, los programas de servidores web, se implementan con un solo objetivo, atender requerimientos de archivos de un software cliente web (browser). Cuando el servidor web recibe el pedido a través de una dirección URL (request), este envía el archivo/documento (por lo general HTML) solicitado hacia el cliente, sin antes realizar un barrido del código HTML en búsqueda de archivos que se puedan adjuntar en el pedido, por ejemplo, si existe un , el servidor web adjuntará el archivo logo.jpg a la respuesta (response), si en la página web, hay un enlace a un archivo local (al servidor web), ese archivo no se enviará, sino hasta cuando se haga click en el enlace correspondiente main page. Por otro lado, si la página contiene varias imágenes o elementos multimedia como animaciones, videos o sonido, todas serán enviadas al cliente.

Una respuesta http, por lo general se compone de: una línea de estado, campos de cabecera, una línea en blanco y el cuerpo de la respuesta. La línea de estado, es la que más nos interesa a los desarrolladores, esta línea dice la versión de http usada y un código de estado de 3 dígitos, por ejemplo: "HTTP/1.1 200 OK". Para saber que dos dice el código se debe empezar por interpretar el primer dígito del código, 1 -> Información, 2 -> éxito, 3 -> redirección, 4-> error en el cliente, 5 -> error en el servidor.

1.4.1. Tecnologías Back-End

De los códigos de estado más comunes están: 404 Not Found, 200 OK, 500 server error. Desde el punto de vista de; back-end esos códigos se los puede interpretar así: 404: Recurso no encontrado, por ejemplo un empleado, un producto. 200: el procesamiento se realizó con éxito. 500, existe algún error de sintaxis o un bug en el servidor.

Para procesar información, los servidores web necesitan extensiones, que en la actualizada están casi embebidos en cada servidor web, por ejemplo Apache+PHP, Payara+JSP, IIS+ASP.NET. eso no significa que JPS no pueda ser ejecutado en un servidor Apache o IIS, en esos casos se debería instalar las extensiones correspondientes.

En general, existen varios lenguajes de programación y tecnologías que permiten implementar el back-end web, aquí se mencionan algunos lenguajes de programación y varis tecnologías.



< 17/33 >



1.4.1. Tecnologías Back-End

Lenguajes de programación.

- PHP
- Java
- JavaScript
- TypeScript
- CoffeeScript
- Ruby
- Python
- Go
- Scala
- C#

Tecnologías.

- Laravel – PHP Framework
- JSP
- Servlets
- Node.js
- Express.js = Node.js para desarrollar apps y APIs
- Ruby on Rails - Framework
- ASP.NET
- YII – PHP5
- Meteor JS – Node.js para aplicaciones en tiempo real
- Zend – PHP, MVC
- Django – Framework MVC, Python



1.4.1. Tecnologías Back-End

Diferentes tecnologías permiten el desarrollo de la lógica del negocio y la conexión a las bases de datos para aplicaciones Web. Alrededor de cada tecnología, existe no sólo un lenguaje de programación, sino también frameworks, bibliotecas, APIs.

Las tecnologías de back-end necesitan de un componente especial instalado en el servidor web seleccionado. En los inicios del web, los programas de servidores web, se implementan con un solo objetivo, atender requerimientos de archivos de un software cliente web (browser). Cuando el servidor web recibe el pedido a través de una dirección URL (request), este envía el archivo/documento (por lo general HTML) solicitado hacia el cliente, sin antes realizar un barrido del código HTML en búsqueda de archivos que se puedan adjuntar en el pedido, por ejemplo, si existe un , el servidor web adjuntará el archivo logo.jpg a la respuesta (response), si en la página web, hay un enlace a un archivo local (al servidor web), ese archivo no se enviará, sino hasta cuando se haga click en el enlace correspondiente main page. Por otro lado, si la página contiene varias imágenes o elementos multimedia como animaciones, videos o sonido, todas serán enviadas al cliente.

Una respuesta http, por lo general se compone de: una línea de estado, campos de cabecera, una línea en blanco y el cuerpo de la respuesta. La línea de estado, es la que más nos interesa a los desarrolladores, esta línea dice la versión de http usada y un código de estado de 3 dígitos, por ejemplo: "HTTP/1.1 200 OK". Para saber que dos dice el código se debe empezar por interpretar el primer dígito del código, 1 -> Información, 2 -> éxito, 3 -> redirección, 4-> error en el cliente, 5 -> error en el servidor.

1.4.2. Servicios Web



Los servicios Web son un método de comunicación para las aplicaciones Web, estas aplicaciones pueden ejecutarse en el Internet, una intranet, una extranet, inclusive en el escritorio o en un dispositivo móvil. En este apartado nos limitaremos a decir que un servicio web es una API o forma parte de una API a nivel de la capa de aplicación, y que por lo general se ejecuta sobre le protocolo HTTP y el puerto 80, pero se podría cambiar, tanto el puerto como el protocolo. Son parte importante del ecosistema web actual, y componente primordial del Web 2.0 (el web interactivo y social). En el apartado 2, se trata en detalle acerca de los servicios Web.

1.4.3. CGIs y Scripts de Servidor JSP, PHP, ASP

Para implementar tantos las aplicaciones web como los servicios Web se debe usar plataformas/tecnologías/lenguajes del lado del servidor. En el semestre anterior, en la asignatura “Programación Web” usted aprendió a desarrollar este tipo de aplicaciones. En esta sección hemos reforzado ese conocimiento, y hemos topado temas adicionales como introducción al siguiente contenido (punto 2.0). En la actualidad, los CGIs han quedado en desuso, pero su concepto se mantiene en el uso de los scripts de servidor como PHP, o JSP. Usando cualquiera de las dos opciones se puede construir una aplicación web Cliente servidor. Por ejemplo, la Figura 11 muestra una aplicación Cliente-Servidor n-capas (no n-tiers, eso es diferente y lo descubriremos más adelante), en la aplicación diagramada, existe una puerta de entrada al sitio web (Home.jsp), luego tenemos la página principal con el menú del sistema (MainPage.jsp), cuando se necesita ejecutar alguna operación en la base de datos, del menú principal vamos a la página jsp necesaria para la acción correspondiente, por ejemplo si se quisiera insertar datos, se llamaría a la página create.jsp.



1.4.3. CGIs y Scripts de Servidor JSP, PHP, ASP

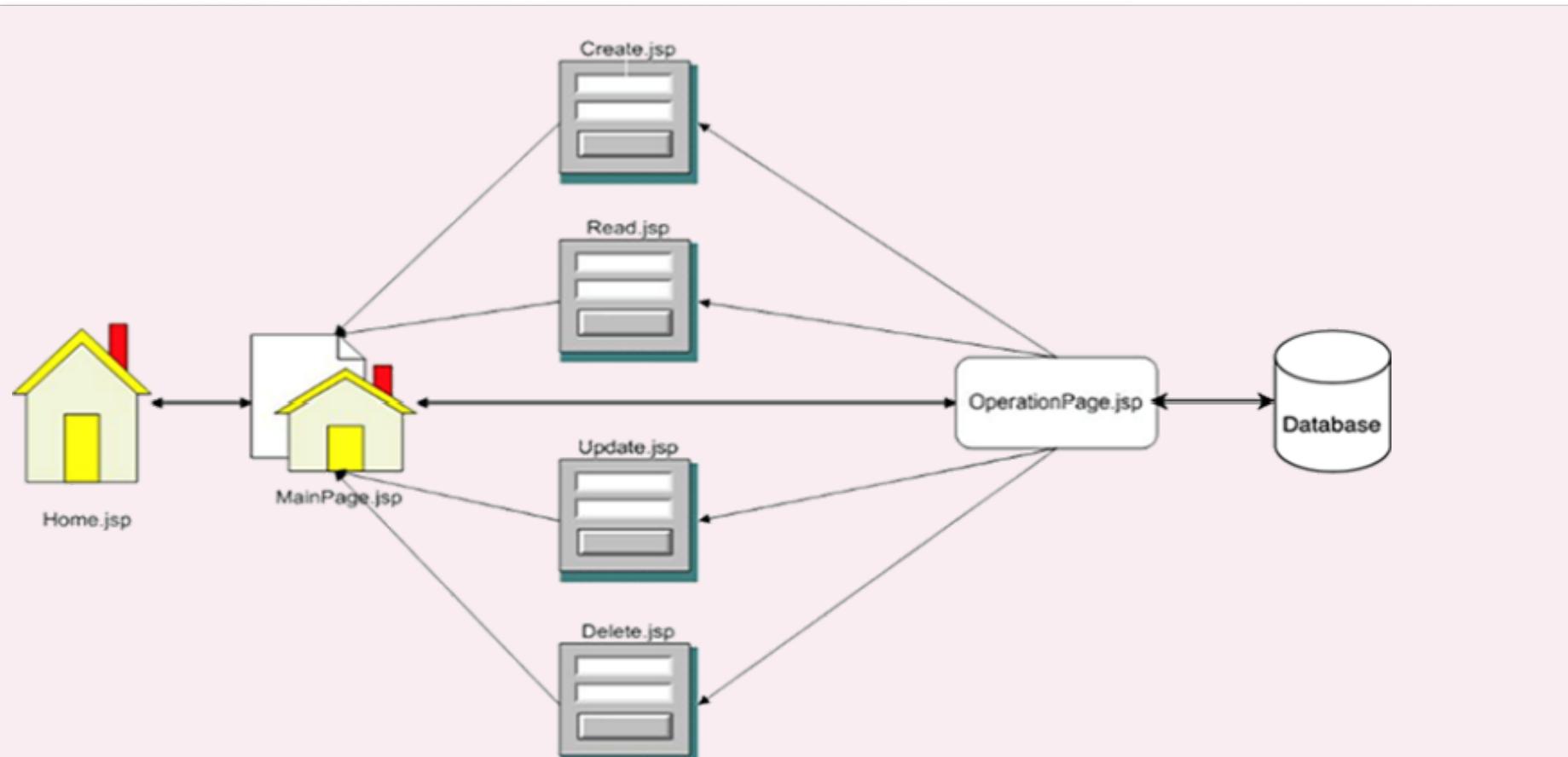


Figura 11. Arquitectura Cliente-Servidor n-capas Web usando JSP

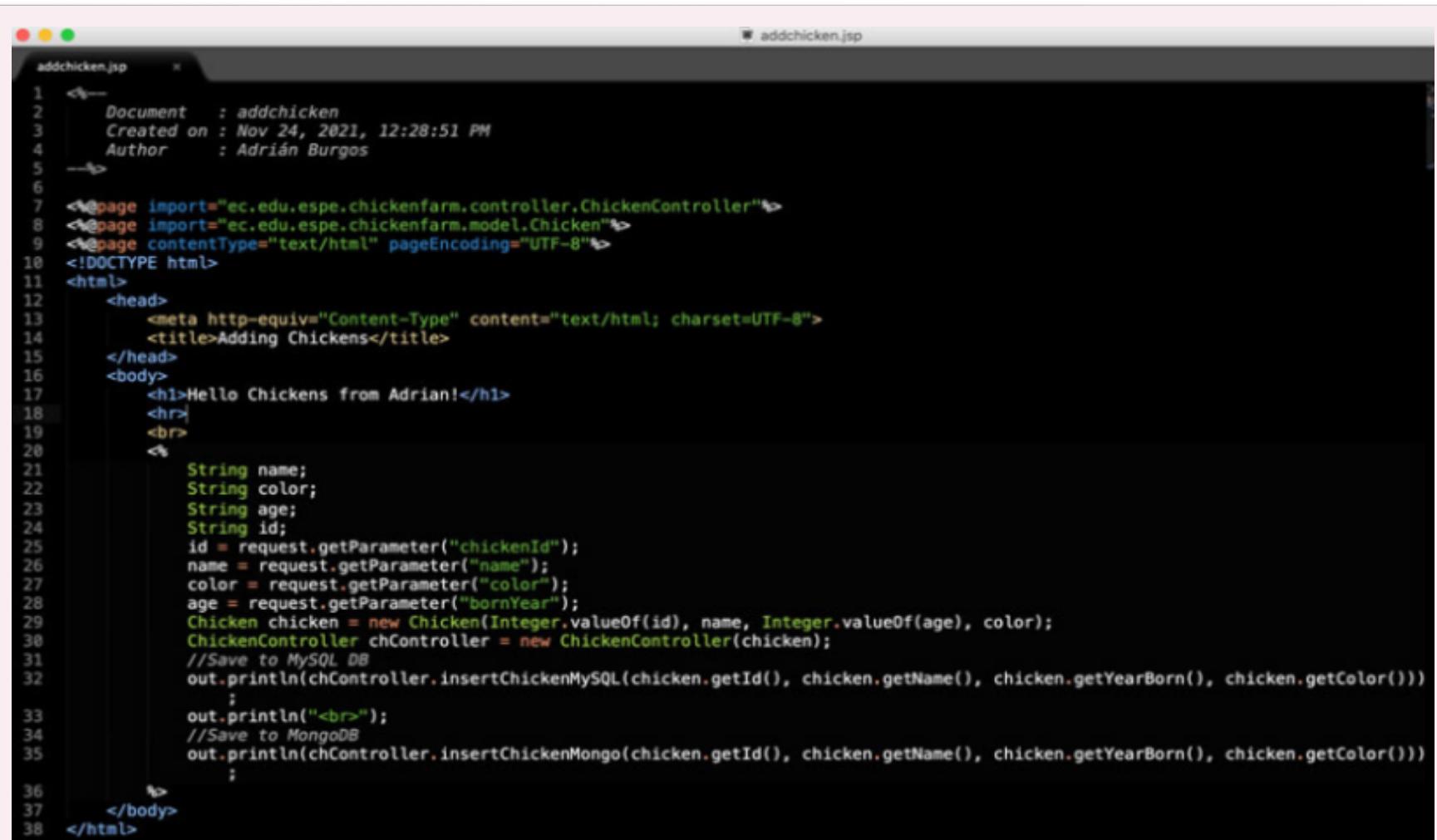
1.4.3. CGIs y Scripts de Servidor JSP, PHP, ASP

Las aplicaciones web que se crean usando scripts de servidor tienen una característica en común: en el mismo archivo mezclan código de interface de usuario (JavaScript, CSS, HTML) con el código de la aplicación (JSP, PHP, ASP).

Cuando se programa este tipo de aplicaciones, se recomienda separar el código de la mejor manera posible, manteniendo el código HTML, JavaScript y CSS en sus archivos correspondientes. En determinado punto, no es posible separar el código de servidor (PHP o JSP) del código HTML, entonces, la habilidad del programador entra en juego a la hora de generar código de calidad para este tipo de aplicaciones. En la , se pude observar un archivo JSP, en el cual se incluye código HTML, embebido en el código JSP, en ese mismo código, las reglas de negocio o manejo de datos de la base de datos se encuentra bien implementado en controladores, en el paquete controlador con código programado en Java.



1.4.3. CGIs y Scripts de Servidor JSP, PHP, ASP



```
addchicken.jsp
1 <%-- Document      : addchicken
2   Created on : Nov 24, 2021, 12:28:51 PM
3   Author       : Adrián Burgos
4 -->
5
6
7 <%@page import="ec.edu.espe.chickenfarm.controller.ChickenController"%>
8 <%@page import="ec.edu.espe.chickenfarm.model.Chicken"%>
9 <%@page contentType="text/html" pageEncoding="UTF-8"%>
10 <!DOCTYPE html>
11 <html>
12   <head>
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14     <title>Adding Chickens</title>
15   </head>
16   <body>
17     <h1>Hello Chickens from Adrian!</h1>
18     <br>
19     <br>
20     <%
21       String name;
22       String color;
23       String age;
24       String id;
25       id = request.getParameter("chickenId");
26       name = request.getParameter("name");
27       color = request.getParameter("color");
28       age = request.getParameter("bornYear");
29       Chicken chicken = new Chicken(Integer.valueOf(id), name, Integer.valueOf(age), color);
30       ChickenController chController = new ChickenController(chicken);
31       //Save to MySQL DB
32       out.println(chController.insertChickenMySQL(chicken.getId(), chicken.getName(), chicken.getYearBorn(), chicken.getColor()));
33       ;
34       out.println("<br>");
35       //Save to MongoDB
36       out.println(chController.insertChickenMongo(chicken.getId(), chicken.getName(), chicken.getYearBorn(), chicken.getColor()));
37       ;
38     %>
39   </body>
40 </html>
```

Figura 12. archivo JSP

1.4.3. CGIs y Scripts de Servidor JSP, PHP, ASP

Para implementar tantos las aplicaciones web como los servicios Web se debe usar plataformas/tecnologías/lenguajes del lado del servidor. En el semestre anterior, en la asignatura “Programación Web” usted aprendió a desarrollar este tipo de aplicaciones. En esta sección hemos reforzado ese conocimiento, y hemos topado temas adicionales como introducción al siguiente contenido (punto 2.0). En la actualidad, los CGIs han quedado en desuso, pero su concepto se mantiene en el uso de los scripts de servidor como PHP, o JSP. Usando cualquiera de las dos opciones se puede construir una aplicación web Cliente servidor. Por ejemplo, la Figura 11 muestra una aplicación Cliente-Servidor n-capas (no n-tiers, eso es diferente y lo descubriremos más adelante), en la aplicación diagramada, existe una puerta de entrada al sitio web (Home.jsp), luego tenemos la página principal con el menú del sistema (MainPage.jsp), cuando se necesita ejecutar alguna operación en la base de datos, del menú principal vamos a la página jsp necesaria para la acción correspondiente, por ejemplo si se quisiera insertar datos, se llamaría a la página create.jsp.



1.4.4. Desarrollo de una aplicación web siguiendo la arquitectura Cliente – Servidor (HANDS-ON-LAB)

A continuación se presenta un taller hands-on-lab (no evaluado), que el alumno debe realizar antes de implementar la actividad 1 del curso, para aprender la tecnología y los lenguajes de programación orientados al desarrollo de aplicaciones web Cliente-Servidor. El presente proyecto fue desarrollado por Michael Cobacango bajo la supervisión del docente autor Jorge Edison Lascano

i. Tema:

Desarrollo de una aplicación web siguiendo la arquitectura Cliente – Servidor.

ii. Objetivo:

Desarrollar una aplicación web (CRUD) y una regla de negocio aplicando la arquitectura Cliente – Servidor.

iii. Descripción:

La presente aplicación web tiene como objetivo gestionar el inventario de los productos que se encuentran en la bodega de una Ferretería, además de calcular las ganancias que se obtendrían en base a la cantidad y precio de un producto en específico. La aplicación web será desarrollada utilizando distintas tecnologías:

- *JSP*: Permite crear páginas web de manera dinámica, las páginas JSP se programan en el Lenguaje de programación Java, y utilizan HTML y XML (es similar a PHP), se puede incluir CSS y JavaScript.
- *Bootstrap*: permite aplicar diseños (de interfaces de usuario) a la aplicación web.

Sexto Es uno clase desarrollado en el horario de programación, lo permitió seguir la aplicación.

A continuación se presenta un taller hands-on-lab (no evaluado), que el alumno debe realizar antes de implementar la actividad 1 del curso, para aprender la tecnología y los lenguajes de programación orientados al desarrollo de aplicaciones web Cliente-Servidor. El presente proyecto fue desarrollado por Michael Cobacango bajo la supervisión del docente autor Jorge Edison Lascano

i. Tema:

Desarrollo de una aplicación web siguiendo la arquitectura Cliente – Servidor.

ii. Objetivo:

Desarrollar una aplicación web (CRUD) y una regla de negocio aplicando la arquitectura Cliente – Servidor.

iii. Descripción:

La presente aplicación web tiene como objetivo gestionar el inventario de los productos que se encuentran en la bodega de una Ferretería, además de calcular las ganancias que se obtendrían en base a la cantidad y precio de un producto en específico. La aplicación web será desarrollada utilizando distintas tecnologías:

- *JSP*: Permite crear páginas web de manera dinámica, las páginas JSP se programan en el Lenguaje de programación Java, y utilizan HTML y XML (es similar a PHP), se puede incluir CSS y JavaScript.
- *Bootstrap*: permite aplicar diseños (de interfaces de usuario) a la aplicación web.
- *Servlet*: Es una clase desarrollada en el lenguaje de programación Java permitirá servir a la aplicación web.
- *MySQL*: Es un sistema de gestión de bases de datos, permitirá manejar los datos de los productos.
- *Java*: Lenguaje de programación, permitirá programar algoritmos, lógica a la aplicación web y el acceso a los datos almacenados en MySQL.
- *MVC*: Conocido como el patrón -Modelo Vista Controlador- es un patrón de arquitectura de software, que nos indica que se debe separar el modelo de los datos, la lógica del negocio su representación; y el módulo que se encarga de la gestión de los eventos y de las comunicaciones.
- *Cliente – Servidor*: Es un modelo de diseño de software donde las tareas se dividen entre los proveedores de recursos o servicios (Servidores) y los demandantes (Clientes). Un cliente realiza peticiones al servidor y el servidor envía respuestas al cliente.

iv. Análisis:

Requisitos Funcionales

La aplicación web deberá cumplir con los siguientes requerimientos:

1. El sistema deberá poder agregar nuevos productos al inventario.
2. El sistema deberá poder mostrar los productos existentes en la bodega de la ferretería.
3. El sistema deberá poder modificar los productos ya existentes.
4. El sistema deberá poder eliminar un producto existente.
5. El sistema deberá calcular de manera automática y guardar en la base de datos las ganancias que se obtendrían al ingresar un nuevo producto con su respectiva cantidad y precio.

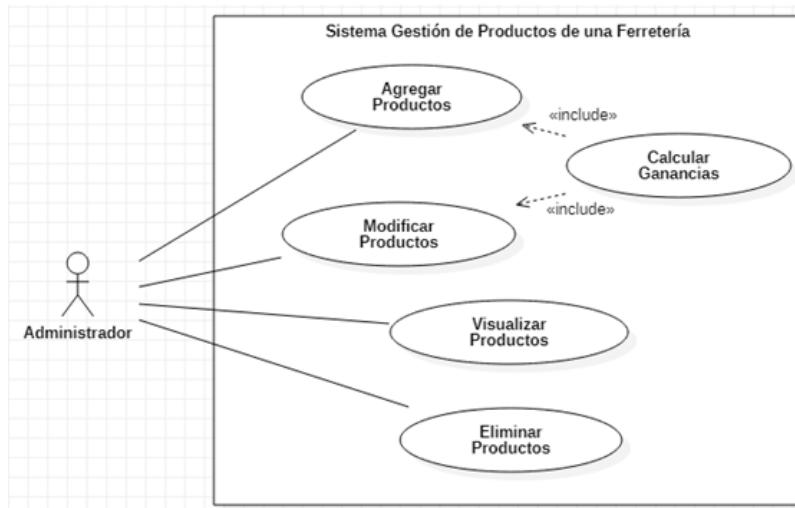
Requisitos No Funcionales

1. El sistema deberá poder ser utilizado desde cualquier navegador web.
2. El sistema deberá almacenar sus datos en una base de datos relacional.
3. El sistema deberá ser diseñado bajo una arquitectura Cliente – Servidor
4. El sistema deberá, en lo posible, de seguir el patrón MVC

v. Diseño:

v.1 Diagrama de Casos de Uso.

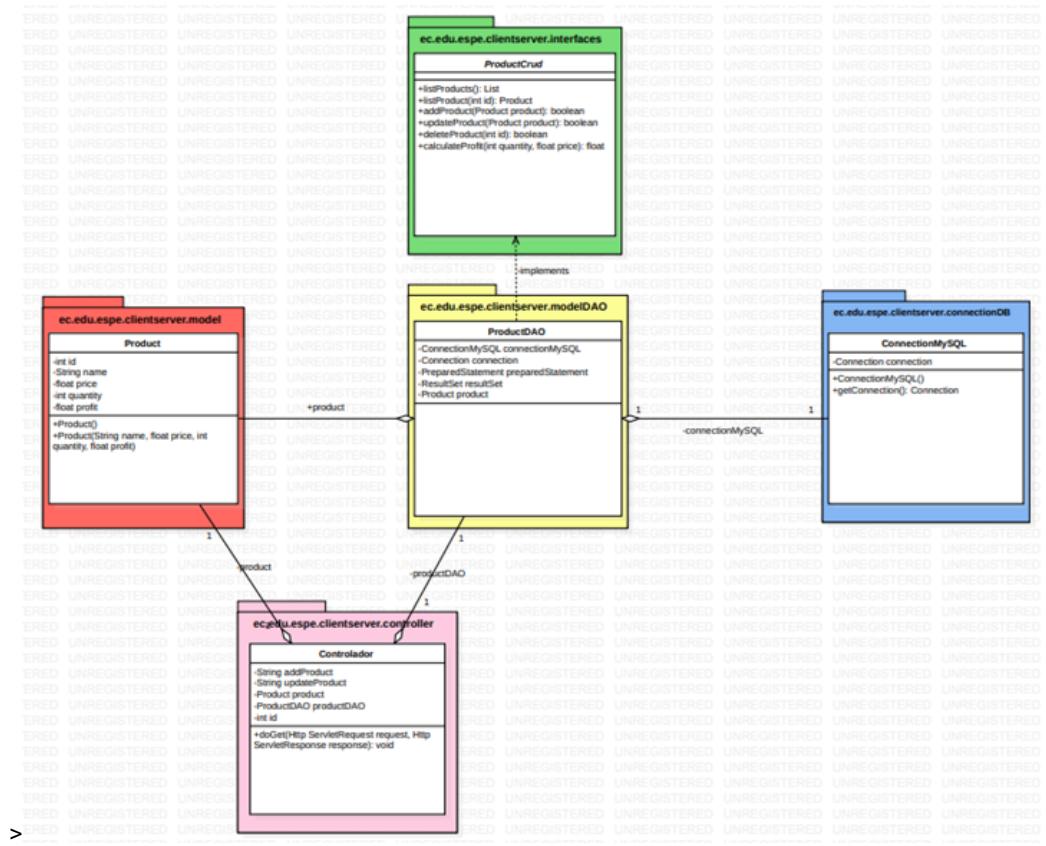
A continuación, se presenta el **Diagrama de Casos de Uso** donde se tiene un actor que es el **Administrador**, quien se encarga de toda la gestión de los productos de la ferretería, los casos de uso representan cada método del CRUD como son los de **Agregar Productos**, **Modificar Productos**, **Visualizar Productos** y **Eliminar Productos**, además se tiene el método de la regla de negocio que es el **Calcular Ganancias**, donde se incluye en los casos de uso de agregar y modificar productos.



v.2 Diagrama de Clases.

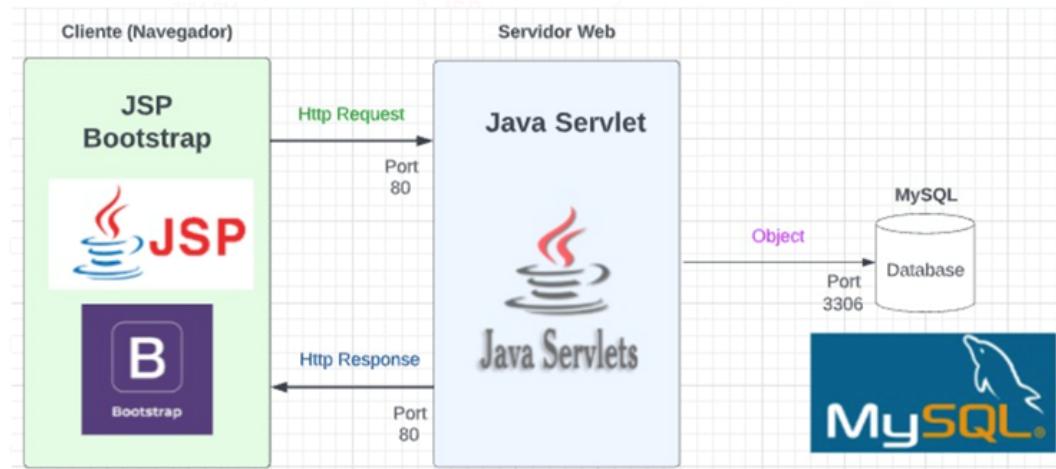
A continuación, se presenta el **Diagrama de Clases**, que muestra la estructura del Sistema de Gestión de Productos de una Ferretería. Se puede diferenciar los distintos paquetes conforme lo indica el patrón de diseño **MVC** (Modelo Vista Controlador). En el paquete **ec.edu.espe.clientserver.interfaces** se encuentra la Interfaz **ProductCrud**, en el paquete **ec.edu.espe.clientserver.modelDAO** se encuentra la clase **ProductDAO** la cual implementa los métodos de la clase **ProductCrud** además una relación de tipo agregación de multiplicidad 1 a 1 con la clase **Product** también tiene una relación de tipo agregación de multiplicidad 1 a 1 con la clase **ConnectionMySQL**, en el paquete **ec.edu.espe.clientserver.model** se encuentra la clase **Product** la misma que permite mantener la persistencia de los datos, en el paquete **ec.edu.espe.clientserver.connectionDB** se encuentra la clase **ConnectionMySQL** que se encarga de realizar las conexiones a la Base de Datos y por último en el paquete **ec.edu.espe.clientserver.controller** se encuentra la clase **Controlador** la misma que se encarga de recibir peticiones y enviar respuestas al cliente, se comunica con el modelo por lo mismo que tiene dos relaciones del tipo agregación de multiplicidad 1 a 1 con las clases **Product** y **ProductDAO**.

Las vistas son realizadas utilizando **JSP** que no es parte de un Diagrama de Clases puesto que un JSP es una tecnología que ayuda a los desarrolladores a crear páginas web dinámicas, más no son clases.



v.3 Arquitectura.

La arquitectura que se va a utilizar se está basada en 2 capas, i.e., **Cliente-Servidor**, donde el **Cliente** es todo aquello que el usuario va a visualizar, para el desarrollo del mismo se va a utilizar JSP y Bootstrap. El cliente realiza una petición al **Servidor** por medio del **puerto 80**, el servidor se encarga de enviar las respuestas al Cliente, además de enviar los productos a la base de datos que los recibe en forma de Objeto por medio del **puerto 3306 (MySQL)**.



v.4 Diseño de la Aplicación.

La lista de los productos que se encuentran en la Ferretería se va a visualizar en una tabla con los siguientes datos: **id**, **Nombre**, **Precio**, **Cantidad**, **Ganancias** además de las acciones: **Editar** y

Eliminar.

También en la interfaz gráfica principal se encontrará la opción de poder agregar un nuevo producto, el diseño se puede visualizar en la siguiente imagen.

The screenshot shows a web browser window with the URL "localhost:8080/index.jsp". The title of the page is "Lista de Productos". Below the title is a table with columns labeled "ID", "Nombre", "Precio", "Cantidad", "Ganancias", "Editor", and "Eliminar". There is one row in the table. Below the table is a single button labeled "Agregar".

Para poder agregar un nuevo producto se pedirá datos como: **Nombre, Precio y Cantidad**, teniendo en cuenta que las ganancias se van a autocalcular por medio de la regla de negocio y el id del producto es de autoincremento y se encarga el Gestor de la Base de Datos además se tendrá la opción de agregar el nuevo producto como se muestra en la siguiente imagen.

The screenshot shows a web browser window with the URL "localhost:8080/addProduct.jsp". The title of the page is "Agregar Producto". Below the title are three input fields labeled "Nombre:", "Precio:", and "Cantidad:". Below these fields is a single button labeled "Agregar".

Para poder modificar un producto se tendrá que elegir la opción de editar desde la tabla que muestra todos los productos, al seleccionar un producto para ser modificado se cargarán los datos de ese producto con el fin de poder modificarlos (**a excepción del id y de las ganancias**) y además contará con una opción de confirmar como se muestra en la siguiente figura.

localhost:8080/updateProduct.jsp

Modificar Producto

Nombre: Tornillos Precio: 0.63 Cantidad: 100 Ganancia: 7.56

A continuación, se ejecuta un **Script** (SELECT * FROM 'productos') para poder visualizar los productos, el resultado muestra el registro de los 2 productos que se encuentran en la base de datos:

```
✓ Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0015 segundos.)
SELECT * FROM `productos`
 Perfilando [ Editar en línea ] [ Editar ] [ Explicar SQL ] [ Crear código PHP ] [ Actualizar ]
 Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla
+ Opciones
← → ▾ id name price quantity profit
    2 Clavos 0.35 100 4.2
    3 Tornillos 0.63 100 7.56
```

vi. Herramientas y Entornos de Desarrollo:

NOTA: Es posible que cuando usted realice este ejercicio, las versiones de las aplicaciones de software que instalen se hayan actualizado, sin embargo, este hands-on lab considera que el software y sus versiones son compatibles y estarán estables por un período de tiempo razonable. Se le aconseja al lector, analizar las versiones más actuales. Además, el estudiante debe estar consciente que la instalación dependerá del Sistema Operativo de su elección, Windows, MacOS, Linux, etc.

IDE: Apache NetBeans IDE 12.5

Descarga e Instalación:

Link de la descarga Apache NetBeans :

<https://netbeans.apache.org/download/nb125/nb125.html>

Para poder realizar la descarga de **Apache NetBeans 12.5** se tiene que ir al link indicado anteriormente donde se mostrará la siguiente pantalla (ver imagen), en esta pantalla se selecciona el tipo de descarga basado en el sistema operativo que se disponga, en este caso se selecciona **Windows**.

Downloading Apache NetBeans 12.5

Apache NetBeans 12.5 was released September 13, 2021. See Apache NetBeans 12.5 Features for a full list of features.

Apache NetBeans 12.5 is available for download from your closest Apache mirror.

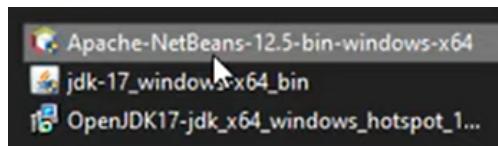
- Binaries: [netbeans-12.5-bin.zip \(SHA-512, PGP ASC\)](#)
- Installers:
 - [Apache-NetBeans-12.5-bin-windows-x64.exe \(SHA-512, PGP ASC\)](#)

Link de descarga de jdk: <https://adoptium.net/>

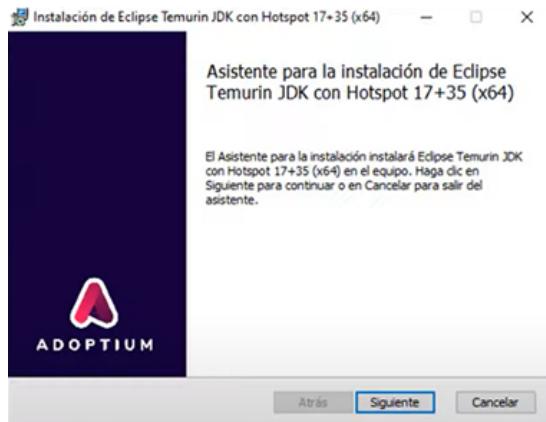
Para poder realizar la descarga del **JDK** (compilador de java) que se utilizará, se tiene que ir al link indicado anteriormente donde. Se mostrará la siguiente pantalla (ver imagen) se selecciona la versión que se va a utilizar, en este caso la de **Temurin 17 (LTS)**, y se procede a darle clic en **Latest release**.



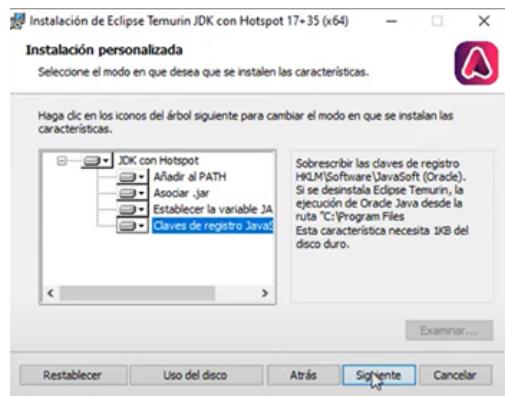
Una vez finalizadas las descargas se tiene que dirigir al sitio donde se escogió para que se guarden los archivos.



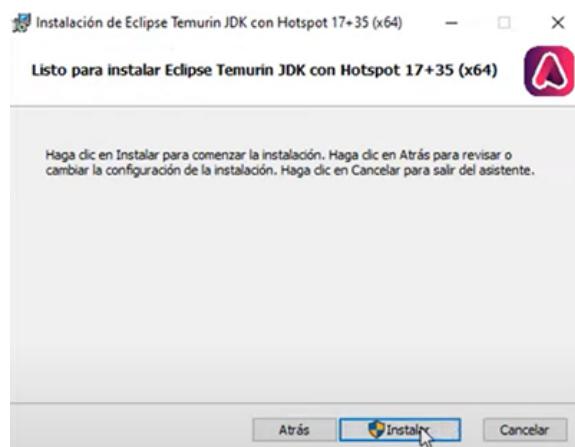
Lo primero que se instalará es el **JDK** para ello se tiene que dar clic derecho sobre el archivo **OpenJDK17-jdk_x64_windows_hotspot** y se tiene que ejecutar como administrador, una vez realizado ese proceso se mostrará la siguiente pantalla, donde se debe de dar clic en **Siguiente**.



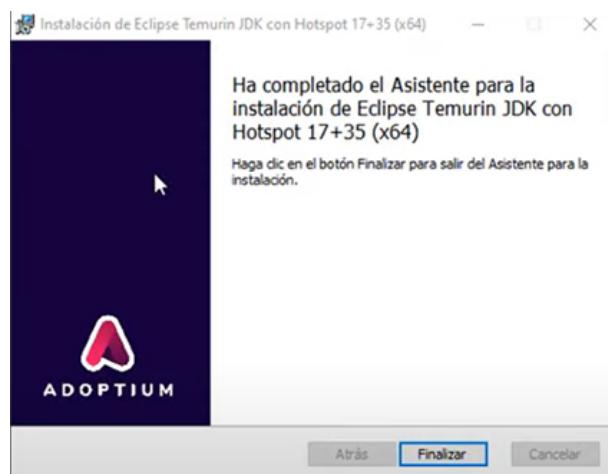
Una vez que se dio clic al botón siguiente en la pantalla anterior se muestra la siguiente pantalla, donde **no se modifica nada** y únicamente se hace clic en el botón **Siguiente**.



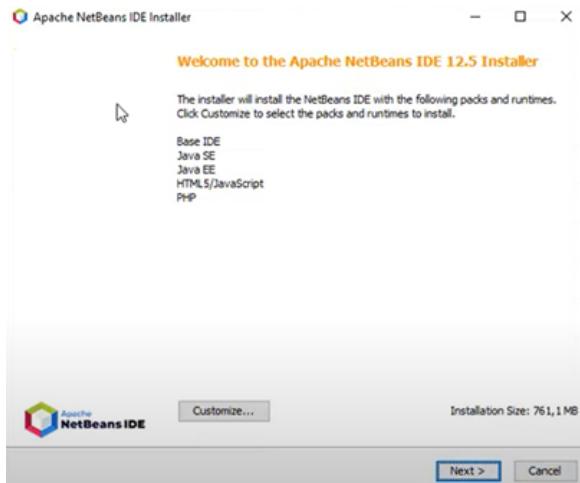
Una vez que se dio clic en el botón siguiente en la pantalla anterior, se mostrará la siguiente pantalla donde únicamente se hace clic al botón **Instalar**.



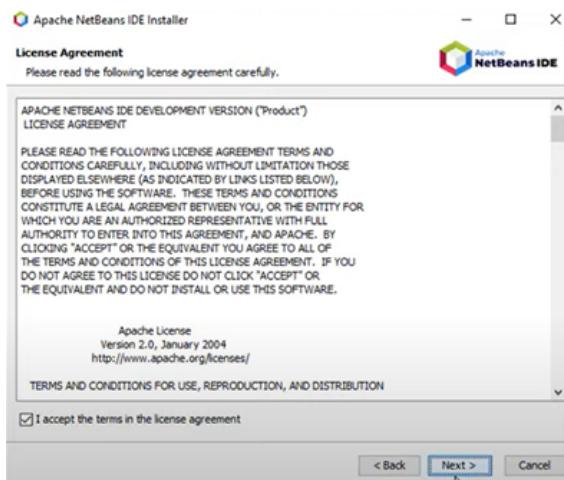
Cuando la instalación haya finalizado se mostrará la siguiente pantalla donde se dará clic al botón **Finalizar**.



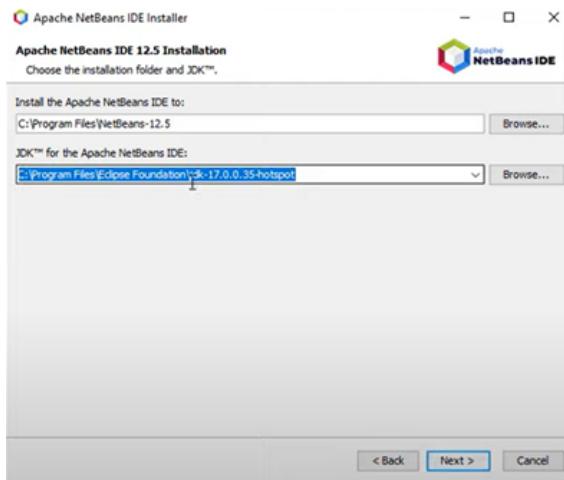
Ahora se procede a la instalación de **Apache NetBeans 12.5** para lo ello se tiene que dirigir a la ruta de descargas y con un clic derecho sobre el ejecutable de NetBeans que se descargó anteriormente se hace clic en ejecutar como administrador, se mostrará la siguiente pantalla y se hace clic en el botón **Next**.



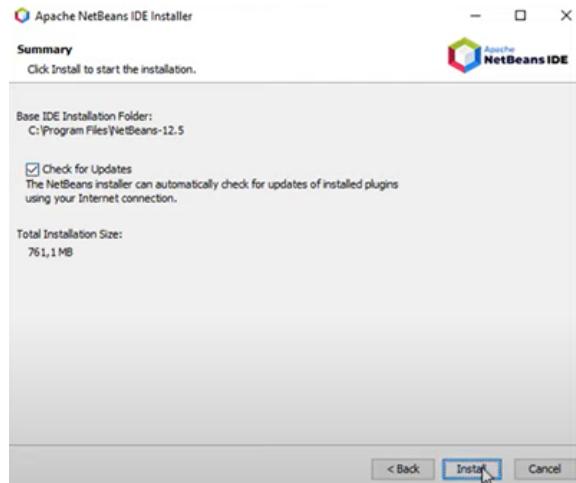
Una vez que se haya dado clic en el botón **Next** en la pantalla anterior, se mostrará la siguiente pantalla, donde se tiene que **aceptar los términos** y se hace clic en el botón **Next**.



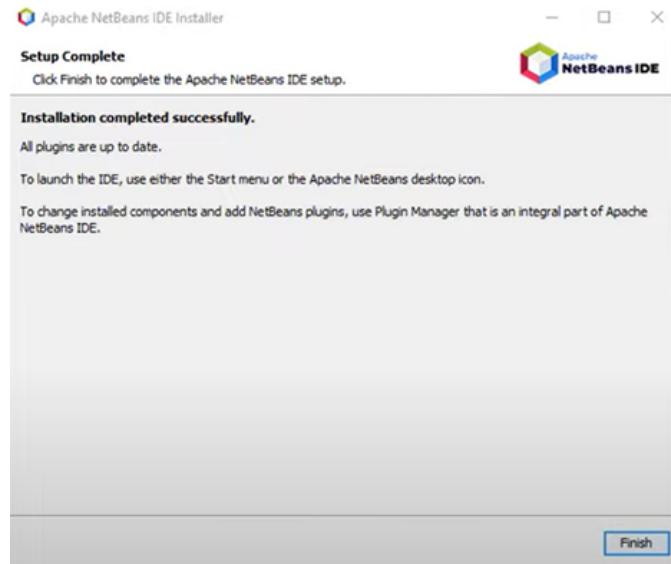
A continuación, se mostrará la siguiente pantalla, donde se deberá elegir la **ruta la instalación** y la **ruta del JDK**, aunque Apache NetBeans detecta automáticamente la ruta del JDK instalado anteriormente, se podría elegir otro JDK de ser necesario. Posteriormente, se hace clic en el botón **Next**.



Una vez que se haya elegido la ruta de instalación y la ruta del JDK, se mostrará la siguiente pantalla donde se tiene que marcar **Check for Updates** y se procede a hacer clic en el botón **Install**.



Cuando haya finalizado la instalación, se mostrará la siguiente pantalla donde finalmente se hace clic en el botón **Finish**.



XAMPP

Descarga e Instalación:

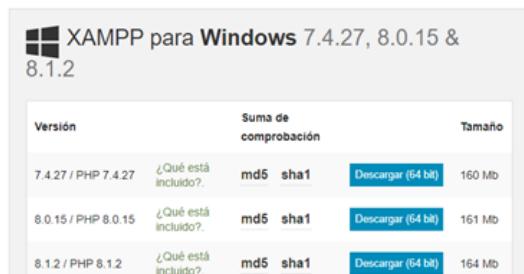
Link de descarga de XAMPP:

<https://www.apachefriends.org/es/download.html>

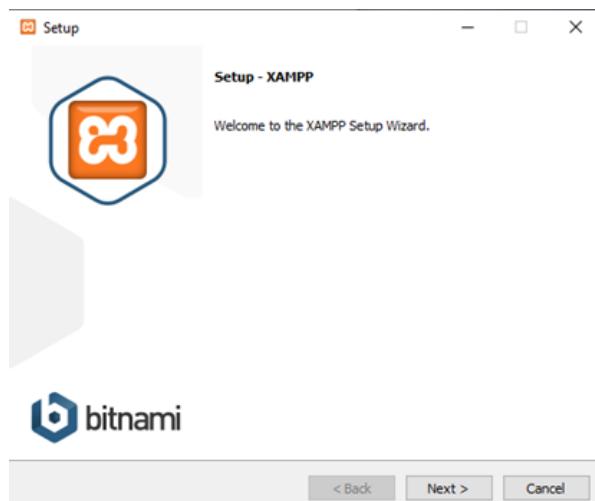
Para poder descargar e instalar **XAMPP** primero se tiene que dirigir al enlace especificado anteriormente, donde se mostrará la siguiente pantalla, se elige la versión que se requiere descargar en este caso la **7.4.27 / PHP 7.4.27**.

Descargar

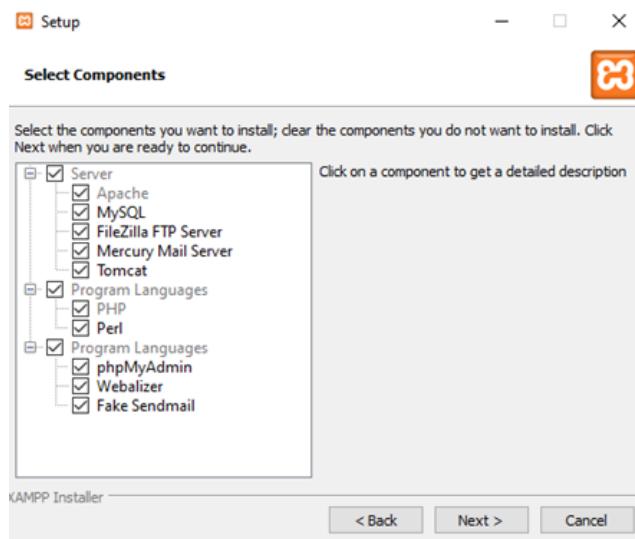
XAMPP es una distribución de Apache fácil de instalar que contiene MariaDB, PHP y Perl. Simplemente descarga y ejecuta el instalador. ¡Es así de fácil!



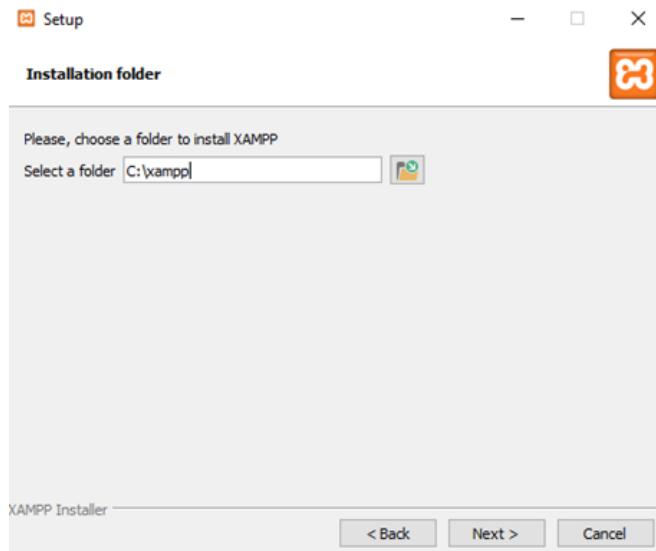
Cuando haya finalizado la descarga, se tiene que dirigir a la ruta de descargas y con un clic derecho sobre el archivo se elige la opción ejecutar como administrador y se mostrará la siguiente pantalla, donde se hace clic en el botón **Next**.



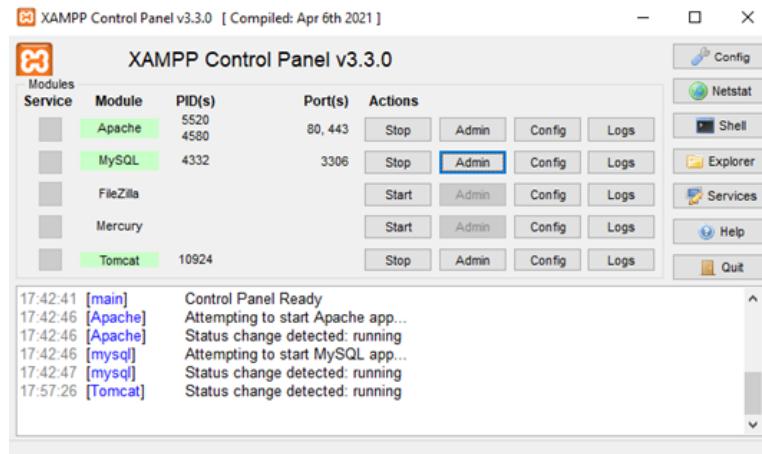
En la siguiente pantalla **no se tiene que modificar nada**, y únicamente se hace clic en el botón **Next**.



En la siguiente pantalla se pedirá la **ruta en la que se quiere instalar XAMPP**, lo recomendable es dejarle en la **raíz del disco C**, como se muestra en la siguiente pantalla y se procede a hacer clic en el botón **Next**.



Una vez finalizada la instalación, se deberá ejecutar la aplicación y se mostrará la siguiente pantalla donde se tiene que **activar Apache y MySQL**. En este paso, se puede decidir levantar los dos servidores manualmente o permitir que el sistema operativo los levante como servicios, se debe considerar que si se selecciona esta opción, los servicios siempre se levantarán al encender la computadora.



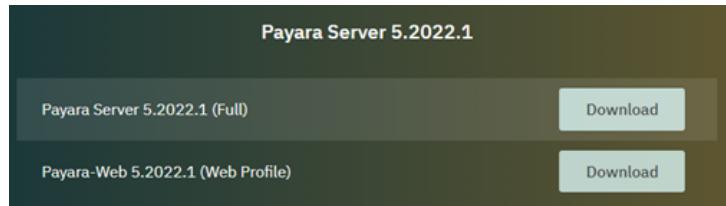
Payara Server

Descarga e Instalación:

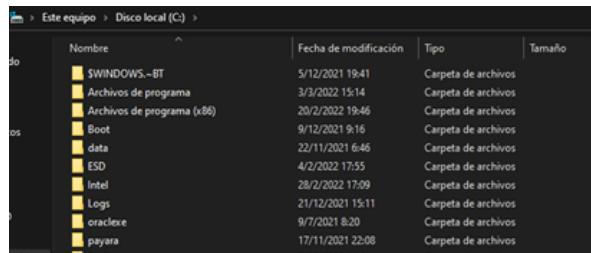
Link de descarga de Payara Server:

<https://www.payara.fish/downloads/payara-platform-community-edition/>

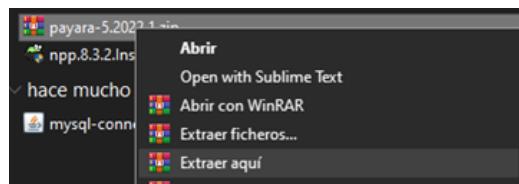
Para poder descargar **Payara Server 5.2022.1** se tiene que dirigir al enlace especificado anteriormente, donde se mostrará la siguiente pantalla y se deberá seleccionar la opción de **Payara Server 5.2022.1 (Full)**, finalmente se tiene que hacer clic en el botón **Download**.



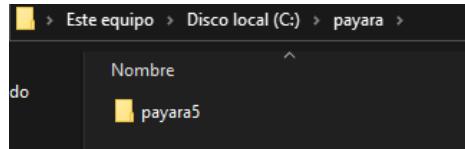
Una vez finalizada la descarga, se tiene que dirigir al equipo y al **Disco Local C** donde se deberá **crear una nueva carpeta llamada payara** como se muestra en la siguiente imagen.



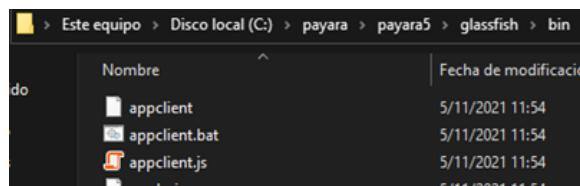
Una vez creada la carpeta se tiene que dirigir a la ruta de las descargas o al sitio donde se guardó el archivo de payara, el archivo que se descarga es un archivo de extensión .zip, donde finalmente se procede a **Extraer** sus datos como se muestra en la siguiente imagen.



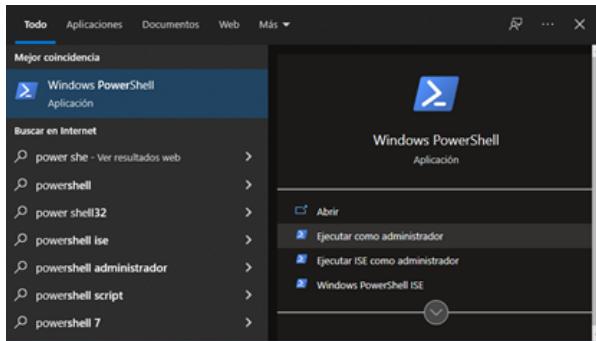
Una vez que se extraen los datos, se procede a copiar y pegar en la carpeta que fue creada anteriormente en el **Disco Local C** llamada **payara**.



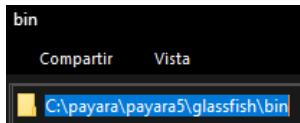
Ahora se tiene que dirigir a la ruta **payara/payara5/glassfish/bin** como se muestra en la siguiente imagen.



Además, se deberá dirigir al Inicio de Windows y buscar **Windows PowerShell** y se deberá ejecutar como administrador.



Se deberá copiar la ruta a la cual se dirigió anteriormente como se muestra en la siguiente pantalla.



En **PowerShell**, se debe dirigir a la ruta que se copió anteriormente haciendo uso del siguiente comando “**cd la ruta que se copió**” sin las comillas, como se muestra en la siguiente imagen.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

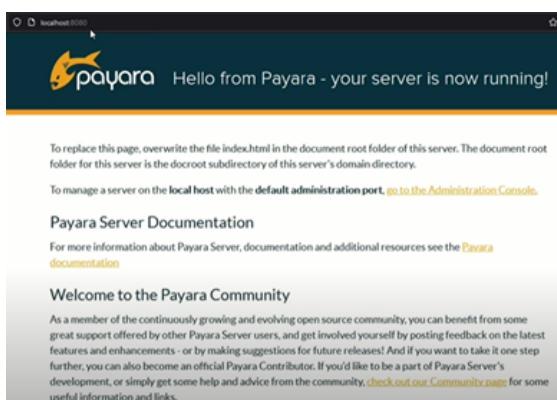
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Windows\system32> cd C:\payara\payara5\glassfish\bin
PS C:\payara\payara5\glassfish\bin>
```

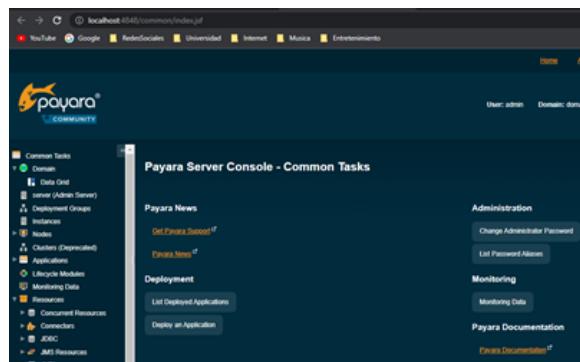
Una vez que se haya dirigido a la ruta, se procede a ejecutar el siguiente comando **./asadmin start-domain** para levantar el Servidor Payara.

```
PS C:\payara\payara5\glassfish\bin> ./asadmin start-domain
Waiting for domain1 to start ..... .
Successfully started the domain : domain1
domain1 Location: C:\payara\payara5\glassfish\domains\domain1
Log File: C:\payara\payara5\glassfish\domains\domain1\logs\server.log
Admin Port: 4848
Command start-domain executed successfully.
PS C:\payara\payara5\glassfish\bin>
```

Cuando se haya levantado el servidor Payara se debe dirigir al navegador de preferencia y en la barra de búsqueda se escribe **http://localhost:8080** y se mostrará la siguiente pantalla.



Para poder visualizar el administrador de Payara nuevamente se tiene que dirigir a un navegador de preferencia y escribir **localhost:4848** y se mostrará la siguiente pantalla.



Para poder detener los servicios de Payara, se deberá dirigir a la ventana de PowerShell y ejecutar el siguiente comando **./asadmin stop-domain** cuando los servicios hayan finalizado se mostrará el siguiente mensaje:

```
PS C:\payara\payara5\glassfish\bin> ./asadmin stop-domain
Waiting for the domain to stop .
Command stop-domain executed successfully.
PS C:\payara\payara5\glassfish\bin> -
```

vii. Drivers:

1. MySQL Driver JDBC, es necesario para que el programa desarrollado en Java pueda conectarse a la base de datos MySQL

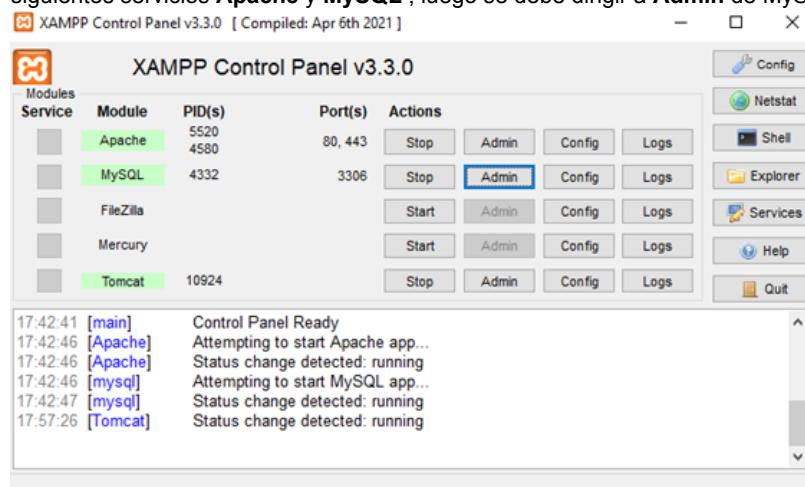
Link de descarga: https://es.osdn.net/projects/sfnet_id2d/downloads/jdbc%20drivers/mysql-connector-java-5.1.15-bin.jar/

viii. Desarrollo:

Creación de la base de datos

1. Activación los servicios de XAMPP

Para comenzar con la creación de la base de datos, debe dirigirse a la aplicación **XAMPP** y activar los siguientes servicios **Apache** y **MySQL**, luego se debe dirigir a **Admin** de MySQL.



Una vez iniciado los servicios y que se haya hecho clic en la opción **Admin** de MySQL se mostrará la siguiente pantalla donde se tiene la ventana inicial del Gestor de Base de Datos.



2. Creación de la base de datos.

Se procede hacer clic en la opción **Nueva** como se muestra en la siguiente pantalla.



Se especifica el nombre a la base de datos en este caso será **ferreteria** y se elige el tipo de Cotejamiento el cual es **utf8_general_ci** finalmente se hace clic en el botón **Crear**.

Bases de datos

A screenshot of the 'Crear base de datos' (Create Database) form. It has a text input field containing 'ferreteria', a dropdown menu set to 'utf8_general_ci', and a 'Crear' (Create) button.

3. Creación de la tabla para almacenar los datos de los productos

En la siguiente pantalla donde se debe dar un nombre a la tabla, en este caso **productos**, y en número de columnas se tiene que elegir **5** ya que esos son los atributos o datos que se almacenarán de cada producto de acuerdo al diagrama de clases. Finalmente se hace clic en el botón **Continuar**.

A screenshot of the 'Crear tabla' (Create Table) form. It has a 'Nombre:' (Name:) input field with 'productos', a 'Número de columnas:' (Number of columns:) input field with '5', and a 'Continuar' (Continue) button.

En la siguiente pantalla se debe escribir el nombre de los atributos, el tipo de dato de cada atributo, la longitud y el tipo de cotejamiento de los mismos que en todos los casos será **utf8_general_ci** además se tiene que seleccionar que **id** será de **autoincremento**, mismo que se convertirá la **clave primaria** al ser una base de datos relacional como se muestra en la siguiente imagen.

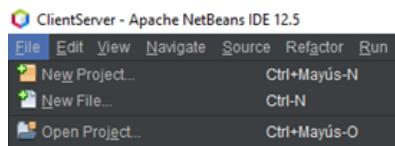
Nombre	Tipo	Largo/Valores	Predeterminado	Colejamiento	Atributos	Not. Índice	A.J. Comentarios
id	INT	4	Ninguna	utf8_general_ci		PRIMARY	
name	VARCHAR	35	Ninguna	utf8_general_ci			
price	FLOAT	4	Ninguna	utf8_general_ci			
quantity	INT	4	Ninguna	utf8_general_ci			
profit	FLOAT	4	Ninguna	utf8_general_ci			
Estructura							

Una vez finalizada la creación de la **tabla productos** se puede visualizar la estructura de la misma, donde deberá ser igual a la que se muestra en la siguiente imagen.

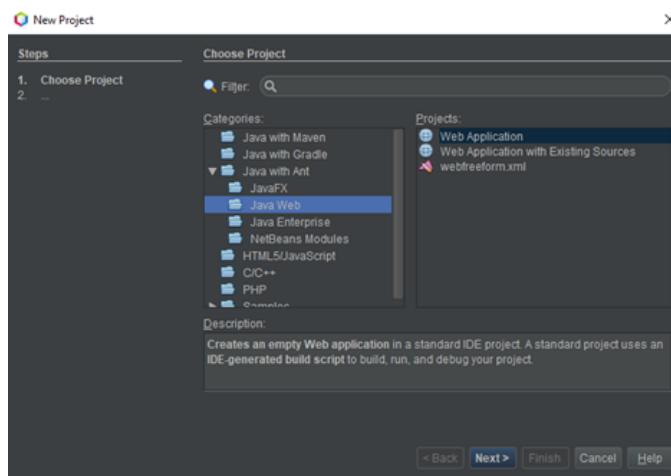
Estructura de tabla						
#	Nombre	Tipo	Colejamiento	Atributos	Nulo	Predeterminado
1	id	int(4)		No	Ninguna	AUTO_INCREMENT
2	name	varchar(35)	utf8_general_ci	No	Ninguna	
3	price	float		No	Ninguna	
4	quantity	int(4)		No	Ninguna	
5	profit	float		No	Ninguna	

4. Creación de la aplicación:

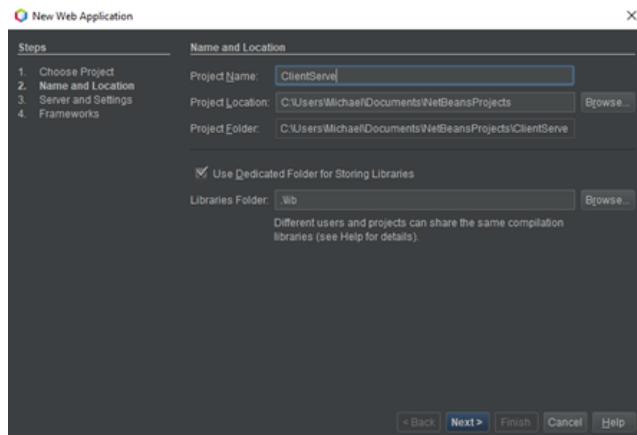
Para crear un nuevo proyecto se deberá dirigir a la aplicación **Apache NetBeans 12.5** que fue descargada e instalada anteriormente se hace clic en **File**, seguido de **New Project** como se muestra en la siguiente imagen.



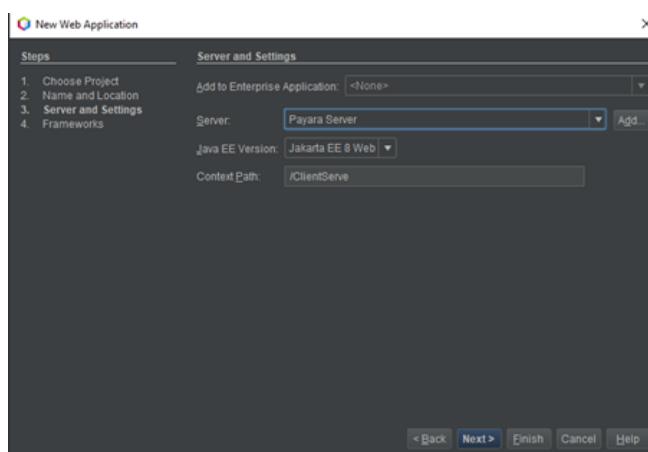
En la siguiente pantalla se tiene que desplegar las opciones de **Java with Ant** y elegir **Java Web** en **categories** seguido de eso se tiene que elegir **Web Application** en **Projects** y finalmente se hace clic en el botón **Next**.



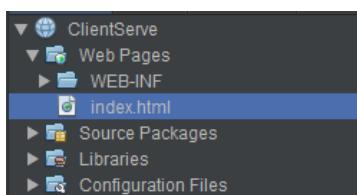
En la siguiente pantalla se da un nombre al proyecto, en este caso **ClientServer** para que tenga relación al tema que se está tratando que es una aplicación web **Cliente – Servidor**, se elige la ruta donde se guardará el proyecto, y finalmente se hace clic en el botón **Next**.



En la siguiente pantalla, se selecciona el servidor que se va a utilizar: **Payara Server**, el resto de las opciones no se deberán modificar y se hace clic en el botón **Finish**.

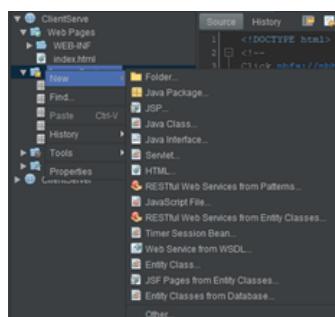


Una vez que se terminó de crear el proyecto se tiene la siguiente estructura.

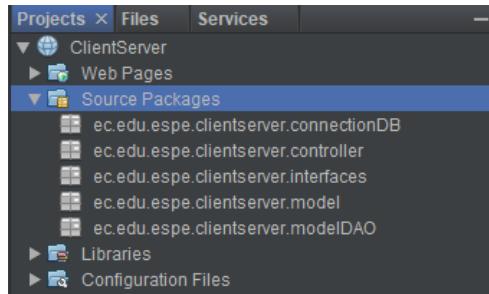


5. Reestructuración del proyecto.

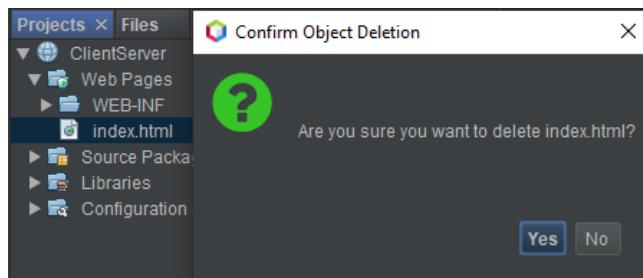
Se deberá dirigir a la carpeta **Sources Package** donde con clic derecho se va a crear los paquetes que servirán para reestructurar el proyecto y seguir con el patrón de diseño **MVC (Modelo Vista Controlador)**.



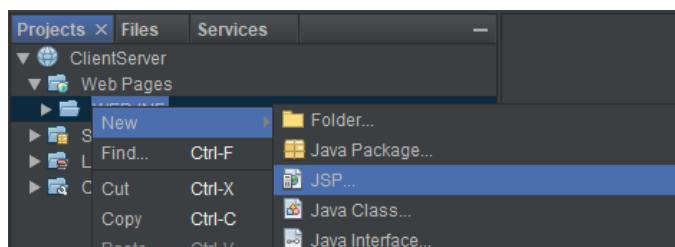
Se deberá crear los paquetes **ec.edu.espe.clientserver.connectionDB**, **ec.edu.espe.clientserver.controller**, **ec.edu.espe.clientserver.interfaces**, **ec.edu.espe.clientserver.model** y **ec.edu.espe.clientserver.modelDAO** lo cual deberá quedar como la siguiente imagen.



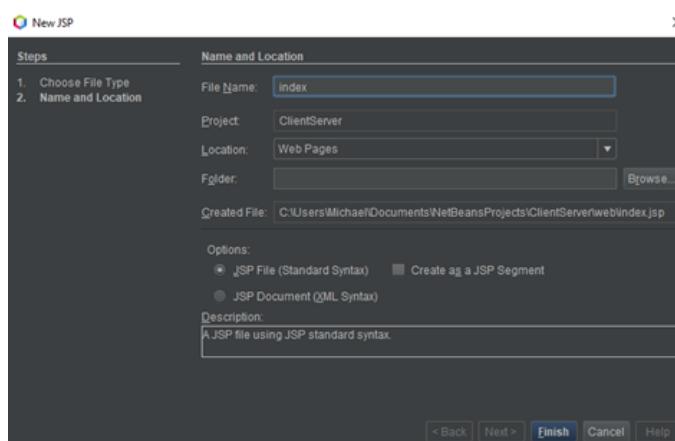
Se debe dirigir a la carpeta **WEB-INF** donde se tiene el archivo **index.html** y se deberá eliminar ya que dicho archivo se genera automáticamente al momento de crear el proyecto, se elimina porque se va a utilizar archivos de extensión **.jsp**.



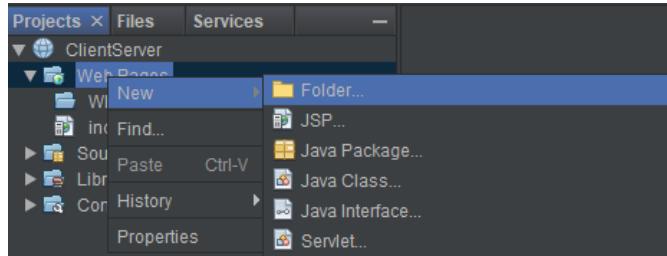
Se hace clic derecho sobre la carpeta **Web Pages**, se tiene que elegir la opción **New** y seguido de **JSP**.



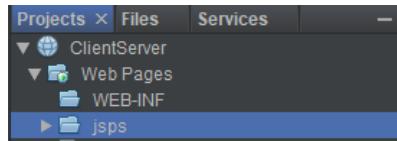
En la siguiente pantalla se debe dar un nombre al archivo en este caso será **index**, y el **resto de las opciones no se deberá modificar** y se hace clic en el botón **Finish**.



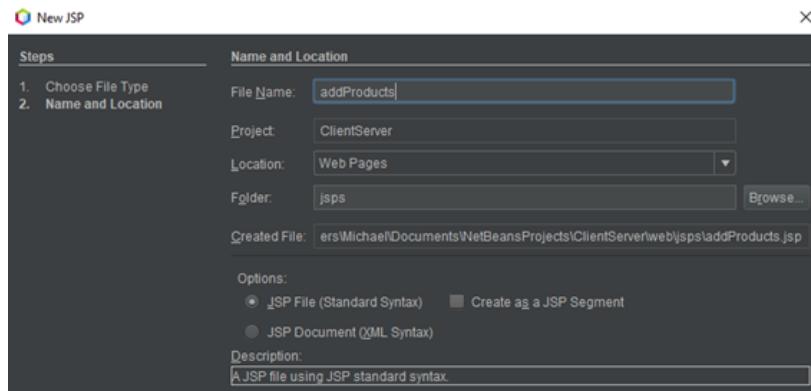
Se hace clic derecho sobre la carpeta **Web Pages**, seguido de **New** y **Folder** en esta carpeta es donde se van a encontrar las vistas, la parte que el usuario o el cliente va a visualizar.



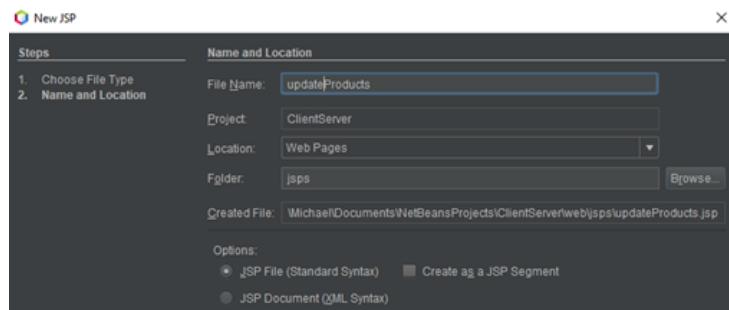
La carpeta creada anteriormente tiene el nombre de **jsp**s donde se va a tener dos archivos de extensión **jsp**: **addProduct** y **updateProduct**.



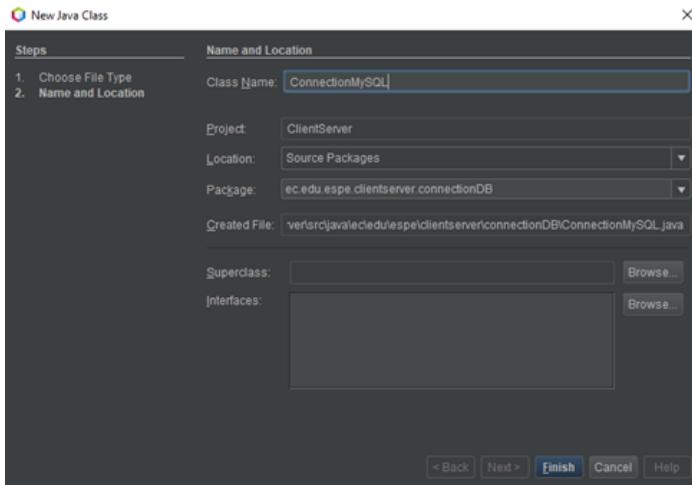
Dentro de la carpeta **jps** se debe crear un nuevo archivo de extensión **JSP** y se tiene que especificar el nombre del archivo en este caso **addProducts**.



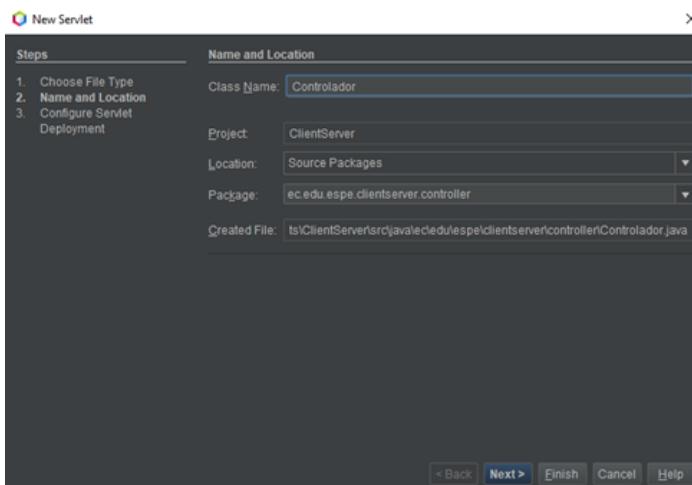
Dentro de la carpeta **jps** se debe crear un nuevo archivo de extensión **JSP** y se tiene que especificar el nombre del archivo en este caso **updateProducts**.



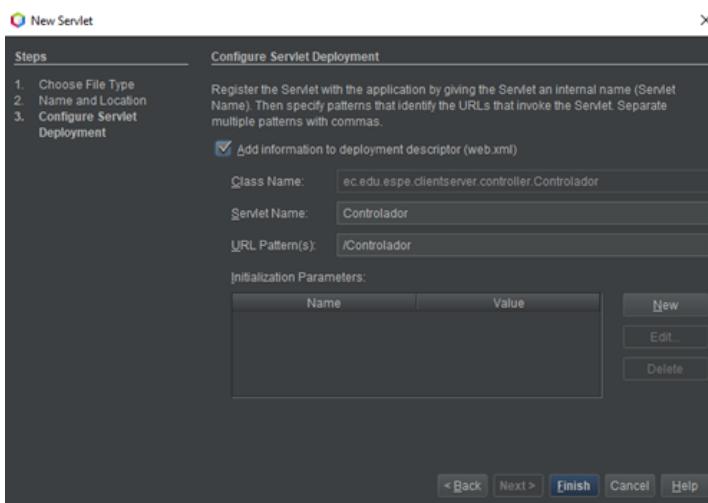
Dentro del paquete **ec.edu.espe.clientserver.controller** que se creó anteriormente se tiene que crear una nueva **Java Class** y se le asigna el nombre de **ConnectionMySQL** en este archivo se realizará la conexión a la base de datos que fue creada anteriormente, **el resto de las opciones no se deberá modificar** y finalmente se hace clic en el botón **Finish**.



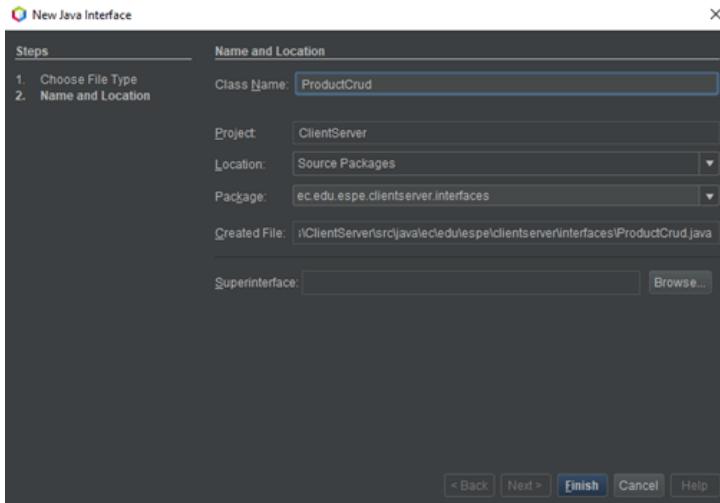
Dentro del paquete **ec.edu.espe.clientserver.controller**, se deberá crear un nuevo **Servlet** y asignarle un nombre en este caso es **Controlador** y finalmente se hace clic en el botón **Next**.



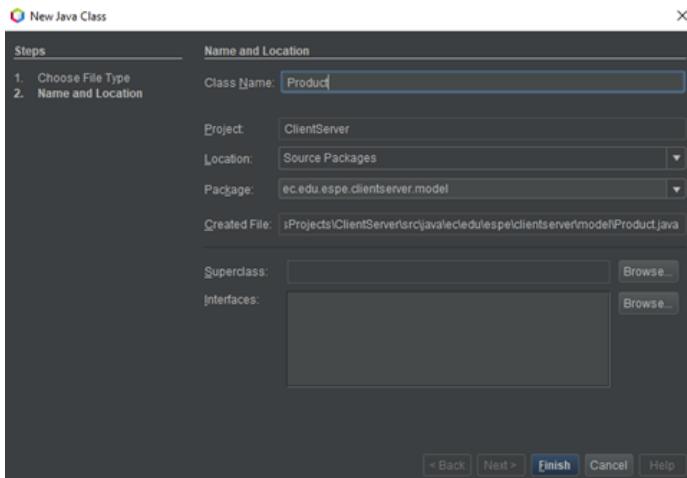
En la siguiente pantalla se deberá marcar la opción de **Add Information to deployment descriptor (web.xml)** y finalmente se hace clic en el botón **Finish**.



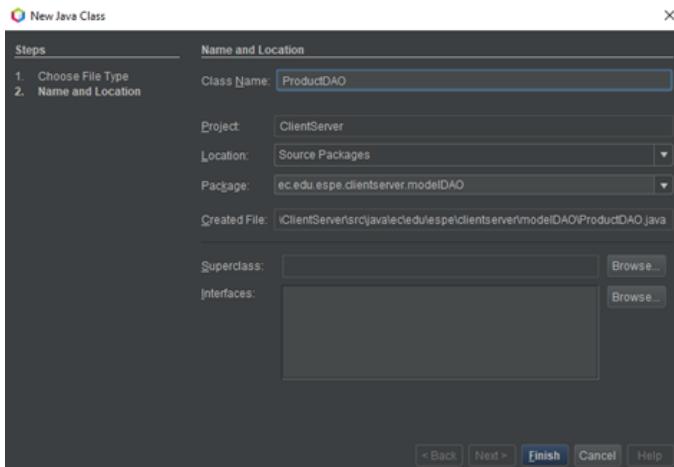
Dentro del paquete **ec.edu.espe.clientserver.interfaces** se deberá crear un nuevo archivo de tipo **Java Interface** con el nombre de **ProductCrud** en este archivo se encontrarán las declaraciones de los métodos CRUD y de la regla de negocio, finalmente se hace clic en el botón **Finish**.



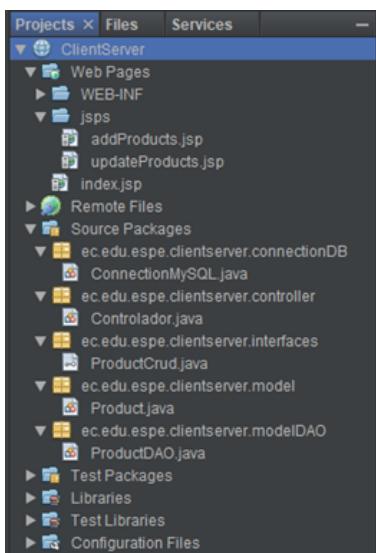
Dentro del paquete **ec.edu.espe.clientserver.model** se debe crear un nuevo archivo del tipo **Java Class** y asignarle el nombre de **Product** en este archivo se programará la persistencia a los datos.



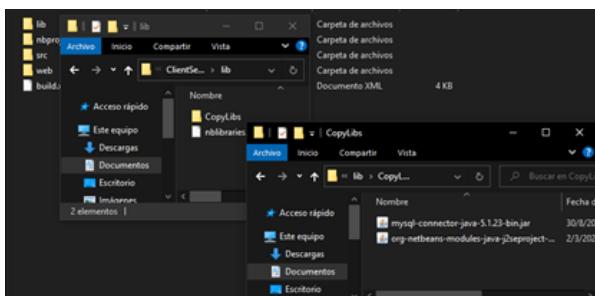
Dentro del paquete **ec.edu.espe.clientserver.modelDAO** se deberá crear un nuevo archivo del tipo **Java Class** y asignarle el nombre de **ProductDAO** este archivo implementará los métodos declarados en la interface.



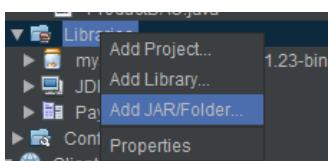
Una vez finalizado la creación de los archivos en cada uno de los paquetes se obtendrá la siguiente estructura.



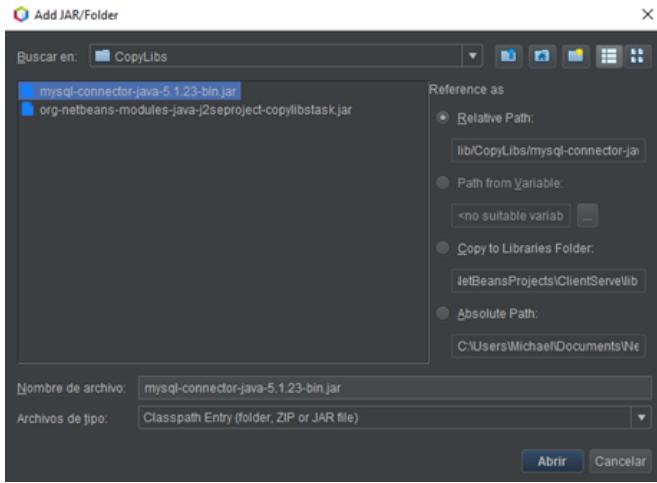
Para poder importar el driver **MySQL Driver JDBC** lo que se deberá realizar es dirigirse a la carpeta **CopyLibs** que se encuentra en el proyecto y pegar el driver.



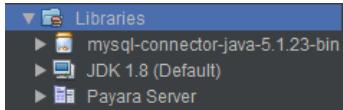
Para poder utilizar el **driver** y finalizar con la configuración del proyecto dirigirse a la opción de **Libraries** donde se tiene que hacer clic derecho y seleccionar la opción de **Add JAR/Folder**.



En la siguiente pantalla se debe seleccionar el driver antes copiado en la carpeta **CopyLibs** y finalmente se hace clic en el botón **Abrir**.



Una vez importado el driver al proyecto, se visualiza de la siguiente manera.



6. Desarrollo de la aplicación.

NOTA: En esta sección, la indentación se la hace al margen para que se pueda leer el código con más claridad.

index.jsp

Este archivo contiene una tabla con los datos de los productos existentes en la ferretería, además de un botón que permite el acceso al formulario para agregar un nuevo producto.

```
<%--  
Document : index  
Created on : 18/03/2022, 12:33:19  
Author : Michael  
--%>  
  
<%@page import="ec.edu.espe.clientserver.model.Product"%>  
<%@page import="ec.edu.espe.clientserver.modelDAO.ProductDAO"%>  
<%@page import="java.util.Iterator"%>  
<%@page import="java.util.List"%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
<head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"  
          rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFlvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"  
          crossorigin="anonymous">  
    <title>JSP Page</title>  
</head>  
<body>  
    <div class="container mt-4">
```

```

<h1 class="text-center mt-4">Lista de Productos</h1>
<table class="table table-dark mt-4">
    <thead>
        <tr>
            <th class="text-center">ID</th>
            <th class="text-center">Nombre</th>
            <th class="text-center">Precio</th>
            <th class="text-center">Cantidad</th>
            <th class="text-center">Ganancias</th>
            <th class="text-center">Editar</th>
            <th class="text-center">Eliminar</th>
        </tr>
    </thead>
    <%
        ProductDAO productDAO = new ProductDAO();
        List<Products> list = productDAO.listProducts();
        Iterator<Product> iter = list.iterator();
        Product product = null;
        while (iter.hasNext()) {
            product = iter.next();

    %>
    <tbody>
        <tr>
            <td class="text-center"><%= product.getId()%></td>
            <td class="text-center"><%= product.getName()%></td>
            <td class="text-center"><%= product.getPrice()%></td>
            <td class="text-center"><%= product.getQuantity()%></td>
            <td class="text-center"><%= product.getProfit()%></td>
            <td class="text-center">
                <a href="Controlador?accion=updateProduct&id=<%= product.getId()%>">
                    <button class="btn btn-primary">
                        Editar
                    </button>
                </a>
            </td>
            <td class="text-center">
                <a href="Controlador?accion=deleteProduct&id=<%= product.getId()%>">
                    <button class="btn btn-danger">
                        Eliminar
                    </button>
                </a>
            </td>
        </tr>
    <% }%>
    </tbody>
</table>
<a href="Controlador?accion=addProducts">
    <button class="btn btn-success">
        Agregar
    </button>
</a>
</div>
</body>
</html>

```

ConnectionMySQL.java

Este archivo contiene la conexión a la base de datos donde se debe proporcionar la información necesaria para conectarse a la base de datos: nombre de la base de datos, usuario y contraseña; en este caso no se tiene una contraseña por lo cual el dato que se le indica es vacío.

```
package ec.edu.espe.clientserver.connectionDB;
```

```

import java.sql.Connection;
import java.sql.DriverManager;

/**
 *
 * @author Michael
 */
public class ConnectionMySQL {

    Connection connection;

    public ConnectionMySQL() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/ferreteria", "root", "");
        } catch (Exception e) {
            System.out.println("Error " + e);
        }
    }

    public Connection getConnection() {
        return connection;
    }
}

```

ProductCrud.java

Este archivo contiene la declaración de los métodos CRUD de la aplicación y las reglas de negocio, en este caso una regla de negocio la cual sirve para calcular las ganancias a partir de la cantidad y el precio de los productos.

```

package ec.edu.espe.clientserver.interfaces;

import ec.edu.espe.clientserver.model.Product;
import java.util.List;

/**
 *
 * @author Michael
 */
public interface ProductCrud {

    public List listProducts();

    public Product listProduct(int id);

    public boolean addProduct(Product product);

    public boolean updateProduct(Product product);

    public boolean deleteProduct(int id);

    /*Regla de negocio*/
    public float calculateProfits(int quantity, float price);
}

```

Product.java

Este archivo contiene los datos de la aplicación, y se encarga de su persistencia

```

package ec.edu.espe.clientserver.model;

/*

```

```

*
* @author Michael
*/
public class Product {

    private int id;
    private String name;
    private float price;
    private int quantity;
    private float profit;

    public Product() {
    }

    public Product(String name, float price, int quantity, float profit) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
        this.profit = profit;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public float getPrice() {
        return price;
    }

    public void setPrice(float price) {
        this.price = price;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public float getProfit() {
        return profit;
    }

    public void setProfit(float profit) {
        this.profit = profit;
    }
}

```

ProductDAO.java

Este archivo contiene el cuerpo de los métodos CRUD además implementa la interfaz ProductCrud, en este archivo se manejan las sentencias SQL que permiten obtener, agregar, modificar y eliminar datos, además de la regla de negocio (calcular total).

```
package ec.edu.espe.clientserver.modelDAO;

import ec.edu.espe.clientserver.connectionDB.ConnectionMySQL;
import ec.edu.espe.clientserver.interfaces.ProductCrud;
import ec.edu.espe.clientserver.model.Product;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Michael
 */
public class ProductDAO implements ProductCrud {

    ConnectionMySQL connectionMySQL = new ConnectionMySQL();
    Connection connection;
    PreparedStatement preparedStatement;
    ResultSet resultSet;
    Product product = new Product();

    @Override
    public List listProducts() {
        ArrayList<Product> list = new ArrayList<>();
        String query = "SELECT * FROM `productos`";
        try {
            connection = connectionMySQL.getConnection();
            preparedStatement = connection.prepareStatement(query);
            resultSet = preparedStatement.executeQuery();
            while (resultSet.next()) {
                Product product = new Product();
                product.setId(resultSet.getInt("id"));
                product.setName(resultSet.getString("name"));
                product.setPrice(resultSet.getFloat("price"));
                product.setQuantity(resultSet.getInt("quantity"));
                product.setProfit(resultSet.getFloat("profit"));
                list.add(product);
            }
        } catch (Exception e) {
            System.out.println("Error" + e);
        }
        return list;
    }

    @Override
    public Product listProduct(int id) {
        String query = "SELECT * FROM `productos` WHERE id=" + id;
        try {
            connection = connectionMySQL.getConnection();
            preparedStatement = connection.prepareStatement(query);
            resultSet = preparedStatement.executeQuery();
            while (resultSet.next()) {
                product.setId(resultSet.getInt("id"));
                product.setName(resultSet.getString("name"));
                product.setPrice(resultSet.getFloat("price"));
                product.setQuantity(resultSet.getInt("quantity"));
                product.setProfit(resultSet.getFloat("profit"));
            }
        } catch (Exception e) {
            System.out.println("Error " + e);
        }
    }
}
```

```

        }
        return product;
    }

@Override
public boolean addProduct(Product product) {
    String query = "INSERT INTO `productos`(`name`, `price`, `quantity`, `profit`) "
        + "VALUES ("
        + product.getName()
        + ", "
        + product.getPrice()
        + ", "
        + product.getQuantity()
        + ", "
        + product.getProfit()
        + ")";
    try {
        connection = connectionMySQL.getConnection();
        preparedStatement = connection.prepareStatement(query);
        preparedStatement.executeUpdate();
    } catch (Exception e) {
        System.out.println("Error " + e);
    }
    return false;
}

@Override
public boolean updateProduct(Product product) {
    String query = "UPDATE productos SET name='"
        + product.getName() + "', price = "
        + product.getPrice() + ", quantity = "
        + product.getQuantity() + ", profit = "
        + product.getProfit() + " WHERE id = " + product.getId();
    try {
        connection = connectionMySQL.getConnection();
        preparedStatement = connection.prepareStatement(query);
        preparedStatement.executeUpdate();
    } catch (Exception e) {
        System.out.println("Error " + e);
    }
    return false;
}

@Override
public boolean deleteProduct(int id) {
    String query = "DELETE FROM productos WHERE id = " + id;
    try {
        connection = connectionMySQL.getConnection();
        preparedStatement = connection.prepareStatement(query);
        preparedStatement.executeUpdate();
    } catch (Exception e) {
        System.out.println("Error " + e);
    }
    return false;
}

/*Regla de negocio*/
@Override
public float calculateProfits(int quantity, float price) {
    float profit;
    float IVA = (float) 0.12;
    profit = (quantity * price) * IVA;
    return profit;
}

```

Controlador.java

En este archivo se encuentra la lógica del negocio, las rutas de acceso a los distintos métodos, además que ayuda a la ejecución del servidor y sirve las páginas web.

```
package ec.edu.espe.clientserver.controller;

import ec.edu.espe.clientserver.model.Product;
import ec.edu.espe.clientserver.modelDAO.ProductDAO;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Michael
 */
public class Controlador extends HttpServlet {

    String addProduct = "jps/addProducts.jsp";
    String updateProduct = "jps/updateProducts.jsp";
    Product product = new Product();
    ProductDAO productDAO = new ProductDAO();
    int id;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Controlador</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet Controlador at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String access = "";
        String action = request.getParameter("accion");
        switch (action) {
            case "addProducts": {
                access = addProduct;
            }
            break;
            case "Agregar": {
                String name = request.getParameter("name");
                float price = Float.parseFloat(request.getParameter("price"));
                int quantity = Integer.parseInt(request.getParameter("quantity"));
                float profit = productDAO.calculateProfits(quantity, price);
                product.setName(name);
                product.setPrice(price);
                product.setQuantity(quantity);
                product.setProfit(profit);
            }
        }
    }
}
```

```

        productDAO.addProduct(product);
    }
    break;
    case "updateProduct": {
        request.setAttribute("id", request.getParameter("id"));
        access = updateProduct;
    }
    break;
    case "Actualizar": {
        id = Integer.parseInt(request.getParameter("idProduct"));
        String name = request.getParameter("name");
        float price = Float.parseFloat(request.getParameter("price"));
        int quantity = Integer.parseInt(request.getParameter("quantity"));
        float profit = productDAO.calculateProfits(quantity, price);
        product.setId(id);
        product.setName(name);
        product.setPrice(price);
        product.setQuantity(quantity);
        product.setProfit(profit);
        productDAO.updateProduct(product);
    }
    break;
    case "deleteProduct": {
        id = Integer.parseInt(request.getParameter("id"));
        product.setId(id);
        productDAO.deleteProduct(id);
    }
    break;
}
RequestDispatcher view = request.getRequestDispatcher(access);
view.forward(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
}

@Override
public String getServletInfo() {
    return "Short description";
}// </editor-fold>

}

```

addProducts.jsp

Este archivo contiene un formulario con los datos del producto a ingresar.

```

<%-- Document : addProduct Created on : 18/03/2022, 12:37:14 Author : Michael
--%> <%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-1BmE4kWBq78iYhFlqvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
    crossorigin="anonymous"
/>
<title>JSP Page</title>
</head>
<body>

```

```

<div class="container mt-4">
  <h1 class="text-center">Agregar Producto</h1>
  <form action="Controlador">
    <div class="row mt-4">
      <div class="col">
        <label>Nombre: </label>
        <input type="text" name="name" class="form-control" />
      </div>
      <div class="col">
        <label>Precio: </label>
        <input type="text" name="price" class="form-control" />
      </div>
      <div class="col">
        <label>Cantidad: </label>
        <input type="number" name="quantity" class="form-control" />
      </div>
    </div>
    <div class="row mt-4">
      <div class="col">
        <p class="text-center">
          <button
            type="submit"
            value="Agregar"
            name="accion"
            class="btn btn-success">
            >
            Agregar
          </button>
        </p>
      </div>
    </div>
  </form>
</div>
</body>
</html>

```

updateProducts.jsp

Este archivo contiene un formulario con los datos que se van a actualizar.

```

<%-- Document : updateProduct Created on : 18/03/2022, 12:38:36 Author : Michael
--%> <%@page import="ec.edu.espe.clientserver.model.Product"%> <%@page
import="ec.edu.espe.clientserver.modelDAO.ProductDAO"%> <%@page
contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-1BmE4kWBq78iYhFlvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
    crossorigin="anonymous"
  />
  <title>JSP Page</title>
</head>
<body>
  <div class="container mt-4">
    <% ProductDAO productDAO = new ProductDAO(); int id =
    Integer.parseInt((String) request.getAttribute("id")); Product product =
    (Product) productDAO.listProduct(id);%>
    <h1 class="text-center mt-4">Modificar Producto</h1>
    <form action="Controlador">
      <div>
        <input type="hidden" name="idProduct" value="<%= product.getId()%>" />
        <input

```

```

        type="hidden"
        name="profit"
        value="<% product.getProfit()%>"
    />
</div>
<div class="row mt-4">
    <div class="col">
        <label>Nombre: </label>
        <input
            type="text"
            name="name"
            value="<% product.getName()%>"
            class="form-control"
        />
    </div>
    <div class="col">
        <label>Precio: </label>
        <input
            type="text"
            name="price"
            value="<% product.getPrice()%>"
            class="form-control"
        />
    </div>
    <div class="col">
        <label>Cantidad: </label>
        <input
            type="number"
            name="quantity"
            value="<% product.getQuantity()%>"
            class="form-control"
        />
    </div>
    <div class="col">
        <label>Ganancias: </label>
        <input
            type="text"
            name="profit"
            value="<% product.getProfit()%>"
            readonly
            class="form-control"
        />
    </div>
</div>
<div class="row mt-4">
    <div class="col">
        <p>
            <button
                type="submit"
                value="Actualizar"
                name="accion"
                class="btn btn-primary"
            >
                Confirmar
            </button>
        </p>
    </div>
</div>
</form>
</div>
</body>
</html>

```

7. Pruebas funcionales de la aplicación.

En la siguiente pantalla se debe seleccionar la opción de **Run Project (ClientServer)**



Una vez que se ejecuta el proyecto, se muestra la siguiente pantalla, que es la tabla con los datos: ID, Nombre, Precio, Cantidad, Ganancias y las opciones Editar, Eliminar y agregar.

ID	Nombre	Precio	Cantidad	Ganancias	Editar	Eliminar
Agregar						

En la siguiente pantalla se muestra el formulario para agregar un nuevo producto donde se deberá ingresar los datos que se solicita como es el Nombre, el Precio y la Cantidad.

Agregar Producto

Nombrer: <input type="text" value="Clavos"/>	Precio: <input type="text" value="0.35"/>	Cantidad: <input type="text" value="100"/>
Agregar		

En la siguiente pantalla se puede visualizar el registro del producto.

ID	Nombre	Precio	Cantidad	Ganancias	Editar	Eliminar
1	Clavos	0.35	100	4.2	Editar	Eliminar
Agregar						

En la siguiente pantalla se ejecuta un Script para poder seleccionar todos los productos existentes en la base de datos.

SELECT * FROM `productos`
<input type="checkbox"/> Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]
<input type="checkbox"/> Mostrar todo Número de filas: <input type="text" value="25"/> Filtrar filas: <input type="text" value="Buscar en esta tabla"/>
+ Opciones
← → <input type="text" value="id name price quantity profit"/> <input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar 1 Clavos 0.35 100 4.2

En la siguiente pantalla se puede modificar el producto que se agregó anteriormente, pero los únicos datos que se van a poder modificar son: Nombre, Precio y Cantidad; las Ganancias se calculan automáticamente por la regla de negocio y el ID no se puede modificar porque es una clave primaria en la base de datos.

Modificar Producto

Nombrer: <input type="text" value="Clavos Nuevos"/>	Precio: <input type="text" value="0.35"/>	Cantidad: <input type="text" value="200"/>	Ganancias: <input type="text" value="4.2"/>
Confirmar			

En la siguiente pantalla se puede visualizar la tabla con la modificación que se realizó al producto.

Lista de Productos						
ID	Nombre	Precio	Cantidad	Ganancias	Editar	Eliminar
1	Clavos Nuevos	0.35	200	8.4	Editar	Eliminar

[Agregar](#)

En la siguiente pantalla se ejecuta un Script para poder visualizar el registro de productos ya que se realizó una modificación.

MySQL Query Result:

```
✓ Mostrando filas 0 - 0 (total de 1, La consulta tardó 0,0006 segundos.)
```

```
SELECT * FROM `productos`
```

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 ▾ Filtrar filas: Buscar en esta tabla

+ Opciones

	← T →	id	name	price	quantity	profit		
<input type="checkbox"/>	Editar	Copiar	Borrar	1	Clavos Nuevos	0.35	200	8.4

En la siguiente pantalla se hace uso de la opción de Eliminar para borrar un producto.

Lista de Productos						
ID	Nombre	Precio	Cantidad	Ganancias	Editar	Eliminar
1	Clavos Nuevos	0.35	200	8.4	Editar	Eliminar

[Agregar](#)

En la siguiente pantalla se puede visualizar la tabla vacía ya que se eliminó el producto.

Lista de Productos						
ID	Nombre	Precio	Cantidad	Ganancias	Editar	Eliminar

[Agregar](#)

En la siguiente pantalla se ejecuta un Script para poder visualizar los productos existentes en la base de datos.

MySQL Query Result:

```
SELECT * FROM `productos`
```

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

id	name	price	quantity	profit
1	Clavos Nuevos	0.35	200	8.4

A continuación el enlace del hands on lab, en Youtube

https://www.youtube.com/playlist?list=PLLV74OII0_H1NdtWeGRjBLWrAUjSYzKNw

El código del taller se encuentra en el siguiente repositorio GitHub:

<https://github.com/elascano/ESPE-DESARROLLO-WEB-2022>

Recursos complementarios

Making servlets

Java Servlets & JSP [3] - Making Servlets



<https://www.youtube.com/watch?v=UiBtz7rZ6Ec>

Java Servlets & JSP [3] - Making Servlets



Bibliografía



9 Types of Web Applications to Simplify Your Business [+Examples]. (s. f.). Recuperado 16 de febrero de 2022, de <https://www.spaceotechnologies.com/types-of-web-applications/>

15th Annual State Of Agile Report | Digital.ai. (s. f.). Recuperado 15 de febrero de 2022, de <https://digital.ai/resource-center/analyst-reports/state-of-agile-report>

Lascano, J. E. (2009). EXtreme Programming (XP) May Be Embedded Inside Scrum. 21st Annual Systems and Software Technology Conference (STSC) 2009: Technology: Advancing Precision. <http://www.ieee-stc.org/proceedings/2009/pdfs/sps81.pdf>

Luchaninov, Y. (s. f.). Web Application Architecture: Choosing the Right Type in 2021. MobiDev. Recuperado 16 de febrero de 2022, de <https://mobidev.biz/blog/web-application-architecture-types>

Progressive Web Application Development by Example. (s. f.). Packt. Recuperado 17 de febrero de 2022, de <https://www.packtpub.com/product/progressive-web-application-development-by-example/9781787125421>

Sekgweleo, T. (2018). Understanding Traditional Systems Development Methodologies. IJAME, 0, Article 0. <https://www.managementjournal.info/index.php/IJAME/article/view/370>

Autoevaluación

1. Las grandes empresas por lo general..

- Tienen su propio equipo de desarrollo web interno.
- Tercerizan el desarrollo de sus aplicaciones web.
- Contratan a un experto en el área de desarrollo web.

Pregunta 1 de 10

[Atras](#)[Siguiente](#)[Enviar todo](#)

Autoevaluación

2. ¿Para el desarrollo de sus aplicaciones web, las grandes empresas usan?

- Metodologías Clásicas como la de cascada.
- Metodologías agiles como SCRUM.
- No usan ningun metodología.

Pregunta 2 de 10

[Atras](#)[Siguiente](#)[Enviar todo](#)

Autoevaluación

3. Seleccione el conjunto de Tecnologías para desarrollar Front-End Web.

- HTML, CSS, JavaScript, React, Vue.js, Bootstrap.
- HTML, CSS, JavaScript, PHP, ASP, JSP, node.js.
- Laravel, Java, Ruby, Python.

Pregunta 3 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

3. Seleccione el conjunto de Tecnologías para desarrollar Front-End Web.

- HTML, CSS, JavaScript, React, Vue.js, Bootstrap.
- HTML, CSS, JavaScript, PHP, ASP, JSP, node.js.
- Laravel, Java, Ruby, Python.

Pregunta 3 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

4. El formato de datos usado en general por los servicios REST es:

- CSV
- XML
- JSON.

Pregunta 4 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

5. Este tipo de aplicaciones web corren completamente en un navegador web y requieren que se maneje muy bien la lógica de presentación:

- Multiple Page Web Applications.
- Content Management Systems.
- Single Page Web Applications.

Pregunta 5 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

6. Responsable de los requerimientos de la aplicación Web en el proceso de desarrollo ágil

- Product Owner.
- Scrum Master.
- Scrum Team.

Pregunta 6 de 10

[Atras](#)[Siguiente](#)[Enviar todo](#)

Autoevaluación

7. En esta arquitectura, cada capa corre en su propia infraestructura, pueden ser 3 o más componentes.

- Client-Server.
- Tier-archivtecture.
- Cloud Architecture.

Pregunta 7 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

8. La aplicación web se presenta al usuario en una sola pantalla y no le obliga a abandonarla, la misma página se forma con diferentes componentes, que incluso pueden ser de diferentes tecnologías.

- Micro Front-End.
- Single page application.
- Server-side rendering.

Pregunta 8 de 10

Atras

Siguiente

Enviar todo



Desarrollo de Aplicaciones Web

Tema 2 Servicios Web



Carrera en Línea
Tecnologías de la Información



< 1/36 >



2.1 Lenguajes y Frameworks para desarrollo Web

En el tema anterior, se hizo un resumen del desarrollo web, los tipos de aplicaciones, el proceso de desarrollo Web y se implementó aplicaciones cliente-servidor, usando uno de los lenguajes más usados hasta la actualidad, Java. Pero el campo de desarrollo de aplicaciones Web es amplio, y casi todos los lenguajes de programación, en estos días, permiten el desarrollo de aplicaciones web en el lado del servidor. Por otro lado, si bien es cierto que solo existen tres lenguajes que se usan del lado del cliente: 1) CSS, es un lenguaje de hojas de estilo que aplicará estilo y apariencia a una página web. 2) HTML, es el lenguaje de marcado de hipertexto utilizado para organizar la estructura y desplegar el contenido de una página web a través del uso de etiquetas. Y, 3) JavaScript, que es un lenguaje de programación, originalmente creado para el lado del cliente, permite hacerle más dinámica a una página, por ejemplo, mostrar contenido actualizado, acceder a datos privados o públicos, entre otros.

Del lado del cliente, como se mencionó en el apartado anterior, es necesario tener lenguajes de programación que ofrezcan las facilidades necesarias para implementar algoritmos, código limpio y acceso a datos de diferentes fuentes. Entre otros lenguajes, tenemos: Java, Python, PHP, JavaScript, C#. A diferencia del lado del cliente, en donde los lenguajes se han estandarizado en la última década, en el lado del servidor, cada cierto tiempo aparecen nuevos lenguajes, pero cumplen con el mismo objetivo, ayudar al servidor web a procesar datos. Como es conocido, la única tarea de un servidor web es proveer de archivos al cliente web que así lo solicite, entonces no tiene capacidad de procesamiento alguna.

Actualmente, los desarrolladores escriben su código utilizando frameworks web, que no son más que colecciones de objetos, funciones, reglas y otras construcciones de programación que resuelven problemas de diseño comunes, aumenta el rendimiento de los programadores, simplificando las tareas que ellos enfrentan en determinado dominio. Por ejemplo, Python con Django o Node.js con express.js permiten que los programadores tengan un servidor HTTP al instante, y así den inicio a la programación requerida sin necesidad de instalación de un servidor Web como Apache, Payara, etc.

2.2 Lenguajes de programación y frameworks del lado del servidor

Para el lado del servidor, en general se puede decir que con todos los lenguajes de programación de propósito general es posible desarrollar programas del lado del servidor para el Web. Como se mencionó en la sección anterior: Java, Ruby, Python Go, Scala, C#, Php, JavaScript, TypeScript,

PHP, lanzado al mercado en 1995, ha dominado el desarrollo de aplicaciones web por algún tiempo, pero siempre ha estado en sitios altos en los rankings de lenguajes de programación web, según WWW Technology Surveys (*W3Techs - extensive and reliable web technology surveys*, s. f.), a febrero de 2022, el lenguaje más popular es PHP con un 77.6% de aceptación, aunque va perdiendo un poco de popularidad sigue a la cabeza como lenguaje de servidor, los cuatro lenguajes que le siguen son: ASP.NET, Ruby, Java, Scala, pero en con una popularidad mucho menor.

Los lenguajes de programación web para el lado del cliente se clasificarían en lenguajes script -PHP, JSP, ASP, JavaScript- y lenguajes de propósito general -C#, Ruby, Java, Python-. Otro tipo de herramientas, como por ejemplo node.js, es un ambiente de ejecución basado en JavaScript que corre en una V8 Engine. A continuación una pequeña lista de sitios web y la tecnología de servidor con la que han sido desarrollados, y ventajas de usar esa tecnología («Top 5 Languages to Server-Side Scripting in 2022», 2022). Se debe considerar que, cada uno de esos desarrolladores no ha usado únicamente esta tecnología:

Tabla 1.

Lenguajes de programación y frameworks del lado del servidor

[Rankings](#) | [Lenguajes](#) | [de Empresarios que lo Venteis](#)

Para el lado del servidor, en general se puede decir que con todos los lenguajes de programación de propósito general es posible desarrollar programas del lado del servidor para el Web. Como se mencionó en la sección anterior: Java, Ruby, Python Go, Scala, C#, Php, JavaScript, TypeScript,

PHP, lanzado al mercado en 1995, ha dominado el desarrollo de aplicaciones web por algún tiempo, pero siempre ha estado en sitios altos en los rankings de lenguajes de programación web, según WWW Technology Surveys (*W3Techs - extensive and reliable web technology surveys*, s. f.), a febrero de 2022, el lenguaje más popular es PHP con un 77.6% de aceptación, aunque va perdiendo un poco de popularidad sigue a la cabeza como lenguaje de servidor, los cuatro lenguajes que le siguen son: ASP.NET, Ruby, Java, Scala, pero en con una popularidad mucho menor.

Los lenguajes de programación web para el lado del cliente se clasificarían en lenguajes script -PHP, JSP, ASP, JavaScript- y lenguajes de propósito general -C#, Ruby, Java, Python-. Otro tipo de herramientas, como por ejemplo node.js, es un ambiente de ejecución basado en JavaScript que corre en una V8 Engine. A continuación una pequeña lista de sitios web y la tecnología de servidor con la que han sido desarrollados, y ventajas de usar esa tecnología («Top 5 Languages to Server-Side Scripting in 2022», 2022). Se debe considerar que, cada uno de esos desarrolladores no ha usado únicamente esta tecnología:

Tabla 1.

Lenguajes de programación y frameworks del lado del servidor

Ranking	Lenguaje de Programación	Empresas que lo usan	Ventajas
1	Java	Airbnb Uber Pinterest LinkedIn Groupon eBay	Escalable y simple Multihilo Open-source Seguridad avanzada

		Evernote Fitbit	
2	C#	Xbox Market invoice Stack Overflow	Gran soporte de Microsoft Syntactic sugar Fácil de aprender
3	PHP	Wikipedia Slack WordPress Tumblr Nvidia Pinterest Facebook	Open-source Versatilidad Actualización constante Facilidad de uso Automatización de funciones
4	Python	Spotify Pandora Netflix Mozilla Uber Pinterest Quora Asana	Open-source Gran cantidad de recursos disponibles Códigos de inserción
5	Node.js	LinkedIn Trello eBay Walmart PayPal	High performance Full-stack (JavaScript) Alta escalabilidad Open-source

A más de los lenguajes de programación, los desarrolladores tienen a su disponibilidad bibliotecas y frameworks de desarrollo. Un framework como tal provee un conjunto de herramientas y bibliotecas para simplificar varias tareas del desarrollador. Los frameworks facilitan la vida del desarrollador, no es obligatorio su uso, pero en general, aceleran el tiempo de desarrollo. No todos los frameworks ofrecen las mismas facilidades, por lo tanto, aprender a utilizar un framework no asegura que el desarrollador pueda aprender a usar otros frameworks basado en su conocimiento previo. Por ejemplo, si usamos Django o Flask, ambos frameworks de Python para mostrar un mensaje, el código es completamente diferente, por ejemplo, los siguientes pedazos de código muestran como Django y Flask retornan un mensaje:

DJANGO:

```
# Django view function
from django.http import HttpResponse

def index(request):
    return HttpResponse('Hello from Python, It is Edison responding')
```

FLASK:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return " Hello from Python, It is Edison responding "
if __name__ == "__main__":
    app.run()
```

Otros Frameworks son («Comparison of Server-Side Web Frameworks», 2022): Laravel, CakePHP, Symfony for PHP; Frogon, Poco, CppCMS for C++; Google Web Toolkit, Java Server Faces, Spring for Java; Express.js, Meteor for JavaScript.

2.2.1 Node.js

Node.js es un ambiente back-end de ejecución de JavaScript, que es multi-plataforma y open-source («Node.Js», 2022). Corre en la V8 engine y ejecuta JavaScript sin necesidad de un navegador Web, por lo tanto, el desarrollador puede programar software en línea de comandos o del lado del servidor para el Web. Al usar JavaScript, node.js adopta la capacidad multiplataforma del lenguaje de programación, adicionalmente, y como ventaja para el programador, podría permitirle que se convierta en un desarrollador Full Stack, en caso de que domine JavaScript para el lado del cliente.

Node.js maneja una arquitectura basada en eventos que es capaz de manejar Entradas y Salidas de manera asíncrona, de esta manera mejora su rendimiento y escalabilidad, y es muy útil para aplicaciones en tiempo real, por ejemplo, aplicaciones médicas, juegos, comunicaciones.

En el mundo actual del desarrollo, las aplicaciones que retornan información (APIs) utilizan formatos de datos que sean compatibles con los clientes que las consumen. La última década ha sido testigo del aparecimiento y uso de dos formatos de datos que han sido muy utilizados, XML y JSON, este último se ha convertido sin lugar a duda en el estándar de facto para compartir datos entre aplicaciones, no únicamente servicios Web, sino también otros tipos de aplicaciones.

JSON es un formato de datos que se basa en el par key:value, donde key representa un identificador relacionado al value (valor) de la derecha (Marrs, 2017). La sintaxis de JSON es simple, utiliza los siguientes caracteres: { , : , [, “, y .. Y a cada elemento lo denomina un Objeto. Un objeto se encuentra encerrado entre llaves { ... } , un arreglo se encierra entre corchetes [...], el nombre del key siempre se debe escribir entre comillas “ “, el value puede escribirse siempre entre comillas, o se puede omitir las comillas, en caso de ser un tipo de datos primitivo numérico, tal como un entero, un flotante, un booleano. Cabe anotar, que el “value” puede diseñarse de manera recursiva y podría ser por ejemplo otro par del tipo key : value. A continuación se muestra un ejemplo de

2.3 APIs públicas

Una API (Application Programming Interface) es un conjunto de funciones y procedimientos que permiten la creación de aplicaciones que acceden a funcionalidades o datos de otras aplicaciones de una manera fácil y transparente, una API permite que la comunicación entre dos o más programas de computadora sea fácil, rápida, y transparente. Una API provee a su programa cliente la información necesaria para que lo pueda utilizar, puede ser tan simple como la firma de una función que contiene los parámetros que recibe la función, su nombre y el tipo de dato que la función retorna, o tan compleja como un objeto que viaja por la red con sus respectivas métodos.

Este concepto de API puede proveer a los desarrolladores una característica adicional, la facilidad de ser multiplataforma. Para esto, las APIs han migrado de ser nativas al sistema operativo o al dispositivo donde se encuentran ejecutándose hacia el Web, donde se valen del protocolo http para prestar sus servicios a clientes de cualquier tipo, por ejemplo, aplicaciones de escritorio, aplicaciones de consola, IoT, Aplicaciones Web, etc.

En conclusión, una API, es una puerta de entrada de un servidor hacia otras aplicaciones cliente para ejecutar procesos o acceder a datos o recursos, el uso de APIs mejora la reusabilidad de código (funcionalidad).



2.3.1 Bibliotecas, Frameworks, las primeras APIs

Para el desarrollo de APIs los lenguajes de programación de propósito general proveen bibliotecas, que se importan al inicio del programa y permiten la implementación de estas interfaces de programación y que su comunicación a través de la red de computadoras pueda implementar. Por otro lado, los Frameworks poseen un conjunto de bibliotecas, pero más allá de proveer las facilidades para la programación, aceleran el desarrollo de APIs. Los frameworks proveen código pre establecido, así como estructuras, procedimientos, clases y funciones.

La diferencia entre API, Framework y Biblioteca (Library). Un framework contiene bibliotecas de código y puede obtener información a través de APIs. Una biblioteca es un grupo de líneas de código que ayudan a simplificar las tareas, por lo general ayudan a reducir las líneas de código que fueran muchas más si no se usaran las bibliotecas, en otras palabras, si no hubiera bibliotecas, el programar debería codificar todo. Las APIs son las interfaces de programación que permiten la interacción entre aplicaciones o sistema operativos, como tal, las primeras APIs fueron conceptualizadas para permitir la programación modular, ya allá en los años 1940, 1950 se tenía un catálogo de bibliotecas para los programas. Ya en las décadas de 1960 y 1970, se acuña el término Application Program Interface (sin inglés en el término Program) para computadoras remotas gráficas, en 1990, la API ya se define con un conjunto de servicios disponibles para el programador y da por inicio el concepto de Web API. Las primeras APIs fueron definidas para el Internet en ORBA (IDL- Interface Definition Language), COM y DCOM. Ya en los años 2000, con el trabajo de disertación de Roy Fielding, se define las interfaces de aplicación de programas basados en red, se populariza el concepto de Web API.

El concepto de “API First Design” implica que las empresas que desarrollan aplicaciones Web, por ende, APIs, deben empezar por el diseño de las APIs antes de analizar los canales de consumo, haciendo de esta manera un diseño ideal de las APIs, pero se debe consultar a los stakeholders acerca del diseño. Se deja de lado el estado de TI, y más adelante las TI se adaptan a las APIs, en lugar de que suceda al contrario.

2.3.2 Server-side

Del lado del servidor, las APIs producen datos o recursos, que son generados con lenguajes del lado del servidor, con información o recursos almacenados en servidores a los cuales tiene acceso el servidor web, los recursos pueden estar en bases de datos relacionales, bases de datos no relacionales, en sistemas de archivos locales, en discos duros virtuales en la Nube o en el mismo servidor. Por ahora diremos que para consumir una API, debemos conocer su URI (Universal Resource Identifier), el cual contiene el protocolo de comunicación (por lo general HTTP), el servidor, el puerto, y el path/camino al recurso, que se traduce como los parámetros necesarios para retornar los recursos solicitados, por ejemplo, el URI <http://api.plos.org/search?q=title:%22Drosophila%22%20and%20body:%22RNA%22&fl=id&start=1&rows=100> retorna la información que se observa en la Figura 5.

En el mismo servicio podemos usar otra URI para buscar artículos técnicos acerca de un tema en particular. Por ejemplo busquemos artículos relacionados a GitHub: <http://api.plos.org/search?q=title:github> (ver Figura 5)

Not Secure — api.plos.org

cloud3682OK Cloud 3730 USU CANVAS CCloud3703 googleInternships AdamsElementary www.affinity-sports.com News Popular >

{ "response":{ "numFound":7, "start":0, "maxScore":13.661479, "docs": [{ "id":"10.1371/journal.pcbi.1004668", "journal":"PLOS Computational Biology", "eissn":"1553-7358", "publication_date":"2016-01-19T00:00:00Z", "article_type":"Education", "author_display":["John D. Blischak", "Emily R. Davenport", "Greg Wilson"], "abstract":[], "title_display":"A Quick Introduction to Version Control with Git and GitHub", "score":13.661479}, { "id":"10.1371/journal.pone.0205898", "journal":"PLOS ONE", "eissn":"1932-6203", "publication_date":"2018-10-31T00:00:00Z", "article_type":"Research Article", "author_display":["Pamela H. Russell", "Rachel L. Johnson", "Shreyas Ananthan", "Benjamin Harnke", "Nichole E. Carlson"], "abstract":["\nIn recent years, the explosion of genomic data and bioinformatic tools has been accompanied by a growing conversation around reproducibility of results and usability of software. However, the actual state of the body of bioinformatics software remains largely unknown. The purpose of this paper is to investigate the state of source code in the bioinformatics community, specifically looking at relationships between code properties, development activity, developer communities, and software impact. To investigate these issues, we curated a list of 1,720 bioinformatics repositories on GitHub through their mention in peer-reviewed bioinformatics articles. Additionally, we included 23 high-profile repositories identified by their popularity in an online bioinformatics forum. We analyzed repository metadata, source code, development activity, and team dynamics using data made available publicly through the GitHub API, as well as article metadata. We found key relationships within our dataset, including: certain scientific topics are associated with more active code development and higher community interest in the repository; most of the code in the main dataset is written in dynamically typed languages, while most of the code in the high-profile set is statically typed; developer team size is associated with community engagement and high-profile repositories have larger teams; the proportion of female contributors decreases for high-profile repositories and with seniority level in author lists; and, multiple measures of project impact are associated with the simple variable of whether the code was modified at all after paper publication. In addition to providing the first large-scale analysis of bioinformatics code to our knowledge, our work will enable future analysis through publicly available data, code, and methods. Code to generate the dataset and reproduce the analysis is provided under the MIT license at <https://github.com/pamelarussell/github-bioinformatics>. Data are available at [\n"\], "title_display":"A large-scale analysis of bioinformatics code on GitHub", "score":13.661479}, { "id":"10.1371/journal.pcbi.1004947", "journal":"PLOS Computational Biology", "eissn":"1553-7358", "publication_date":"2016-07-14T00:00:00Z", "article_type":"Editorial", "author_display":\["Yasset Perez-Riverol", "Laurent Gatto", "Rui Wang", "Timo Sachsenberg", "Julian Uszkoreit", "Felipe da Veiga Leprevost"\], "abstract":\[\], "title_display":"Editorial: Bioinformatics on GitHub", "score":13.661479} \] } \] }](https://doi.org/10.17605/OSF.IO/UWHX8)

2.3.3 Client-side



Del lado del cliente, para consumir una API pública se puede usar simple HTML y JavaScript, principalmente se debe conocer el método http del servicio: (GET, PUT, POST, DELETE), la URI y el formato de datos usado. El cliente debe recibir los datos y mostrarlos al usuario final como el programador lo deseé, en un formulario, en una tabla, o en una página web con información legible. Algunas herramientas para desarrollo de clientes son: AngularJS (casi en desuso), React.js, vie.js, etc. En la siguiente unidad, se estudiará y utilizará una de estas herramientas a profundidad. Por ahora nos concentraremos en el lado del servidor.

2.3.4 APIs Web públicas

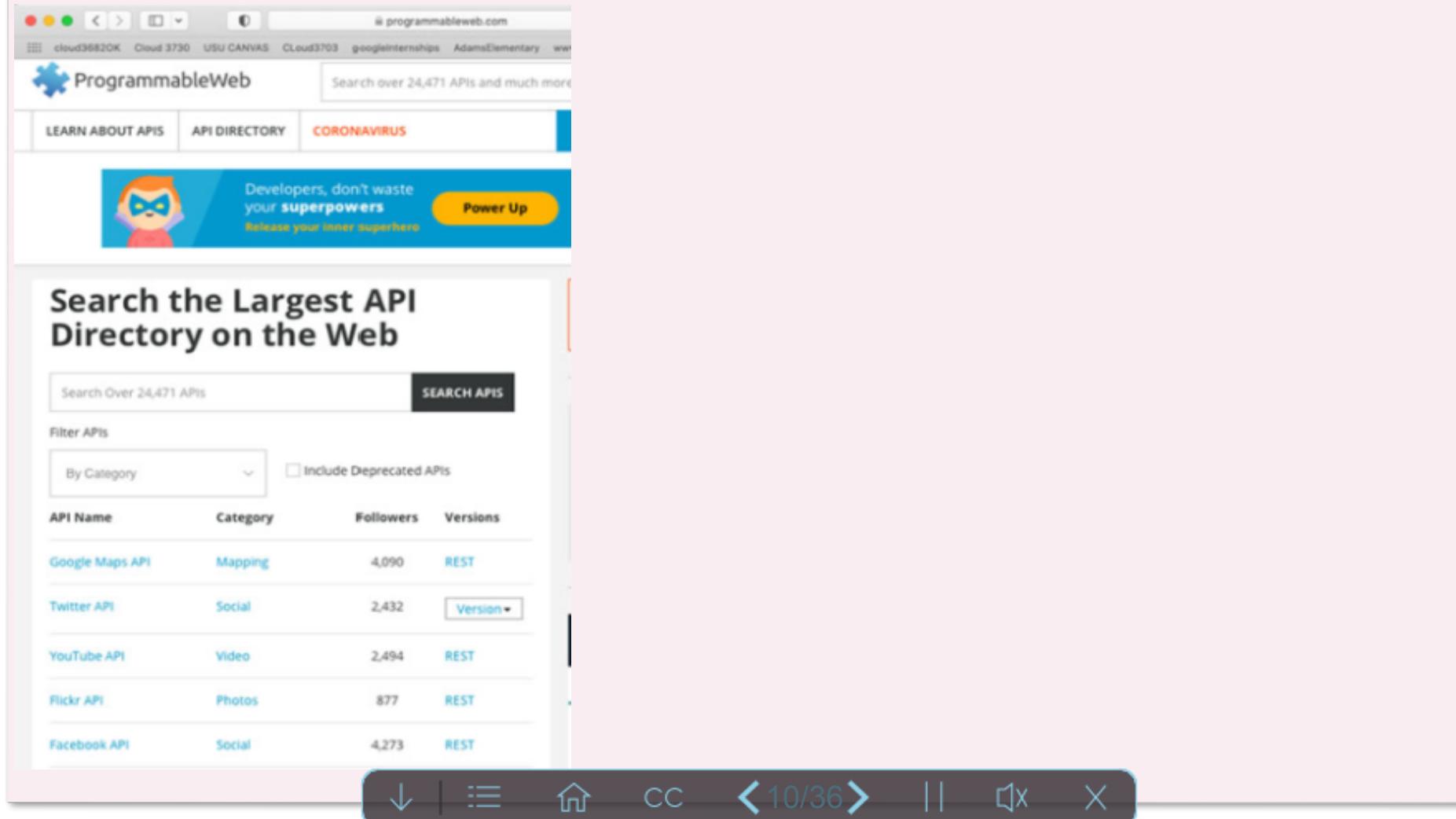
Las APIs públicas como tal, se las conoce así porque se puede acceder a ellas desde cualquier parte del Internet, existen pocas que son free, pero casi en su totalidad tiene un costo el consumo de los servicios que proveen. Los servicios por lo general retornan un recurso, este recurso puede ser representado como un tipo MIME (Multipurpose Internet Mail Extension), el cual es un estándar de Internet para enviar información adjunta a los correos electrónicos, por ejemplo, texto (ASCII), audio, video, imágenes, varios programas de aplicaciones, documentos de Word, etc. Afortunadamente para los programas cliente y para los programas servidor, esta información puede viajar en modo texto en formato JSON, y el browser la puede interpretar de acuerdo al tipo MIME seleccionado.

En muchos casos para las aplicaciones web, es necesario acceder a esta información “pública” para su beneficio y así no tener que programar lo que otros servicios ofrecen, por ejemplo, existen APIs que retornan datos del clima, datos de Geolocalización, validación de usuarios. El portal <https://www.programmableweb.com>, muestra una lista de alrededor de 24471 APIs públicas en distintas categorías: Mapping, Social, Video, Photos, eCommerce, Telephony, Music, Messaging, etc.



2.3.4 APIs Web públicas

Figura 7
Ejemplos de APIs



The screenshot shows the ProgrammableWeb website, which is a directory of over 24,471 APIs. The interface includes a search bar at the top, a sidebar for filtering APIs by category, and a main table displaying various APIs with columns for Name, Category, Followers, and Versions. Popular APIs listed include Google Maps API, Twitter API, YouTube API, Flickr API, and Facebook API.

API Name	Category	Followers	Versions
Google Maps API	Mapping	4,090	REST
Twitter API	Social	2,432	Version ▾
YouTube API	Video	2,494	REST
Flickr API	Photos	877	REST
Facebook API	Social	4,273	REST

2.3.4 APIs Web públicas

Como se puede observar, en su mayoría son APIs tipo REST (más adelante en esta guía se cubre el tema REST). Programmableweb.com a más de ser mostrar la colección de APIs públicas, es una ayuda para el desarrollador, porque además muestra información acerca de los SDKs disponibles, las bibliotecas que se puede usar, y un enlace al sitio para el desarrollador. El uso de aplicaciones de terceros y la combinación con APIs propias del desarrollador en una nueva aplicación se conoce como un MASHUP. La ventaja como se puede concluir es la reutilización de funcionalidad, pero una de sus mayores desventajas es, que si uno de esos servicios es dado de baja por su dueño, esa funcionalidad dejará de existir en el Mashup. Esto obligará al desarrollador a cubrir esa falencia desarrollando esa funcionalidad o accediendo a otro servicio que se encuentre activo. Las APIs públicas funcionan bajo varios modelos de cobranza, por ejemplo, subscripción, pay-per-use, licencia, etc.

Aparte de REST, otros protocolos usados por las APIs son, JSON, específicos, XML, http, SOAP, FTP. En su mayoría, las APIs implementan REST, SOAP, JavaScript y XML-RPC. De acuerdo a (Which Are the Most Popular APIs in the World?, s. f.), las APIs más populares durante el 2021 fueron: Facebook API, Google Maps API, Twitter API, YOUTube API, AccuWeather API, LinkedIn API, Family Watchdog Sex Offender API, Yahoo Weather API, Amazon Product Advertising API.



2.3.4 APIs Web públicas

Las APIs públicas como tal, se las conoce así porque se puede acceder a ellas desde cualquier parte del Internet, existen pocas que son free, pero casi en su totalidad tiene un costo el consumo de los servicios que proveen. Los servicios por lo general retornan un recurso, este recurso puede ser representado como un tipo MIME (Multipurpose Internet Mail Extension), el cual es un estándar de Internet para enviar información adjunta a los correos electrónicos, por ejemplo, texto (ASCII), audio, video, imágenes, varios programas de aplicaciones, documentos de Word, etc. Afortunadamente para los programas cliente y para los programas servidor, esta información puede viajar en modo texto en formato JSON, y el browser la puede interpretar de acuerdo al tipo MIME seleccionado.

En muchos casos para las aplicaciones web, es necesario acceder a esta información “pública” para su beneficio y así no tener que programar lo que otros servicios ofrecen, por ejemplo, existen APIs que retornan datos del clima, datos de Geolocalización, validación de usuarios. El portal <https://www.programmableweb.com>, muestra una lista de alrededor de 24471 APIs públicas en distintas categorías: Mapping, Social, Video, Photos, eCommerce, Telephony, Music, Messaging, etc.



2.4 Servicios REST

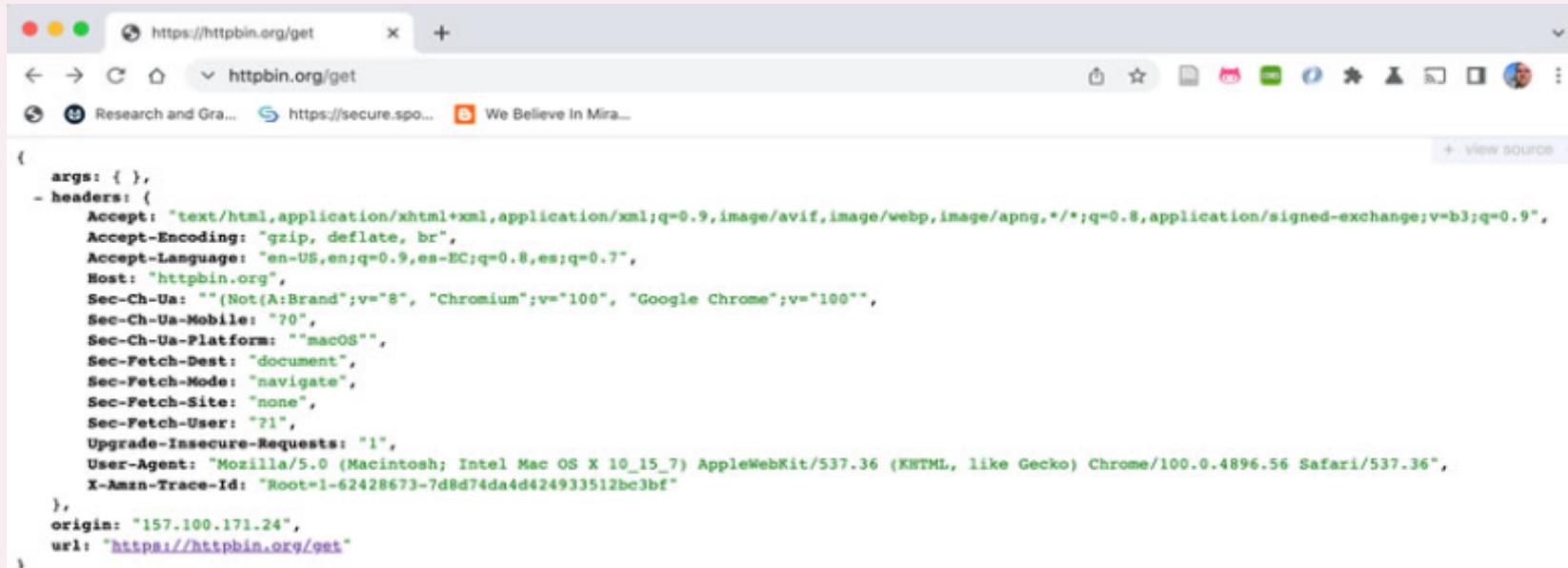
Los servicios web, componentes principales del Web 2.0 han evolucionado junto con sus plataformas y lenguajes de programación tales como: Java, /Net, Python, PHP, Ruby, TypeScript, Node.js, y muchos más. Al mismo tiempo, han ido de la mano y han sobre pasado a los objetos distribuidos: JEE-EJBs, .Net remoting. Si hablamos de Java por ejemplo, se tiene los JPA (APIs de persistencia de java): Linq, Hibernate, Eclipse Link, todos ellos manejan el concepto de ORM (Object Relational Mapping), donde los objetos de programación son mapeados a datos de una base de datos, i.e., objetos con tablas, atributos con columnas, tipos de datos primitivos con tipos de datos de BDD.

Pensando en la arquitectura MVC (Model-View-Controller), los servicios REST representan el controller, donde se realizan las reglas de negocio, y existe comunicación directa con la vista (cliente Web).

Básicamente un servicio REST es un programa que retorna una respuesta en base a una petición enviada a través de una URI, por ejemplo <http://www.myCompany.com/employee/1> retornaría los datos del empleado 1 de mi empresa. REST (Representational State Transfer) es un estilo arquitectónico orientado a recursos. A partir de la disertación de grado PhD de Roy Fielding (Architectural Styles and the Design of Network-based Software Architectures, s. f.), inicia la propagación en los años 2000, un servicio REST ofrece sus servicios por lo general a través del protocolo HTTP, del cual Fielding es uno de sus autores. Cabe notar que podría usarse otro protocolo. A continuación un cliente web (browser) solicita a un servicio web (REST) información de la máquina, a través de <https://httpbin.org/get>. El servicio le responde con la información solicitada (response) a manera de recurso en formato JSON:

2.4 Servicios REST

Figura 8
Json response



A screenshot of a web browser window displaying a JSON response. The URL in the address bar is `https://httpbin.org/get`. The browser interface includes standard controls like back, forward, and search, along with a toolbar and a tab bar showing other open pages.

```
{
  "args": {},
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
    "Accept-Encoding": "gzip, deflate, br",
    "Accept-Language": "en-US,en;q=0.9,es-EC;q=0.8,es;q=0.7",
    "Host": "httpbin.org",
    "Sec-Ch-UA": "(Not(A:Brand";v="8", "Chromium";v="100", "Google Chrome";v="100",
    "Sec-Ch-UA-Mobile": "70",
    "Sec-Ch-UA-Platform": "macOS",
    "Sec-Fetch-Dest": "document",
    "Sec-Fetch-Mode": "navigate",
    "Sec-Fetch-Site": "none",
    "Sec-Fetch-User": "?1",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.56 Safari/537.36",
    "X-Amzn-Trace-Id": "Root=1-62428673-7d8d74da4d424933512bc3bf"
  },
  "origin": "157.100.171.24",
  "url": "https://httpbin.org/get"
}
```

2.4 Servicios REST

Los servicios web, componentes principales del Web 2.0 han evolucionado junto con sus plataformas y lenguajes de programación tales como: Java, /Net, Python, PHP, Ruby, TypeScript, Node.js, y muchos más. Al mismo tiempo, han ido de la mano y han sobre pasado a los objetos distribuidos: JEE-EJBs, .Net remoting. Si hablamos de Java por ejemplo, se tiene los JPA (APIs de persistencia de java): Linq, Hibernate, Eclipse Link, todos ellos manejan el concepto de ORM (Object Relational Mapping), donde los objetos de programación son mapeados a datos de una base de datos, i.e., objetos con tablas, atributos con columnas, tipos de datos primitivos con tipos de datos de BDD.

Pensando en la arquitectura MVC (Model-View-Controller), los servicios REST representan el controller, donde se realizan las reglas de negocio, y existe comunicación directa con la vista (cliente Web).

Básicamente un servicio REST es un programa que retorna una respuesta en base a una petición enviada a través de una URI, por ejemplo <http://www.myCompany.com/employee/1> retornaría los datos del empleado 1 de mi empresa. REST (Representational State Transfer) es un estilo arquitectónico orientado a recursos. A partir de la disertación de grado PhD de Roy Fielding (Architectural Styles and the Design of Network-based Software Architectures, s. f.), inicia la propagación en los años 2000, un servicio REST ofrece sus servicios por lo general a través del protocolo HTTP, del cual Fielding es uno de sus autores. Cabe notar que podría usarse otro protocolo. A continuación un cliente web (browser) solicita a un servicio web (REST) información de la máquina, a través de <https://httpbin.org/get>. El servicio le responde con la información solicitada (response) a manera de recurso en formato JSON:

2.4.1 Web Resources

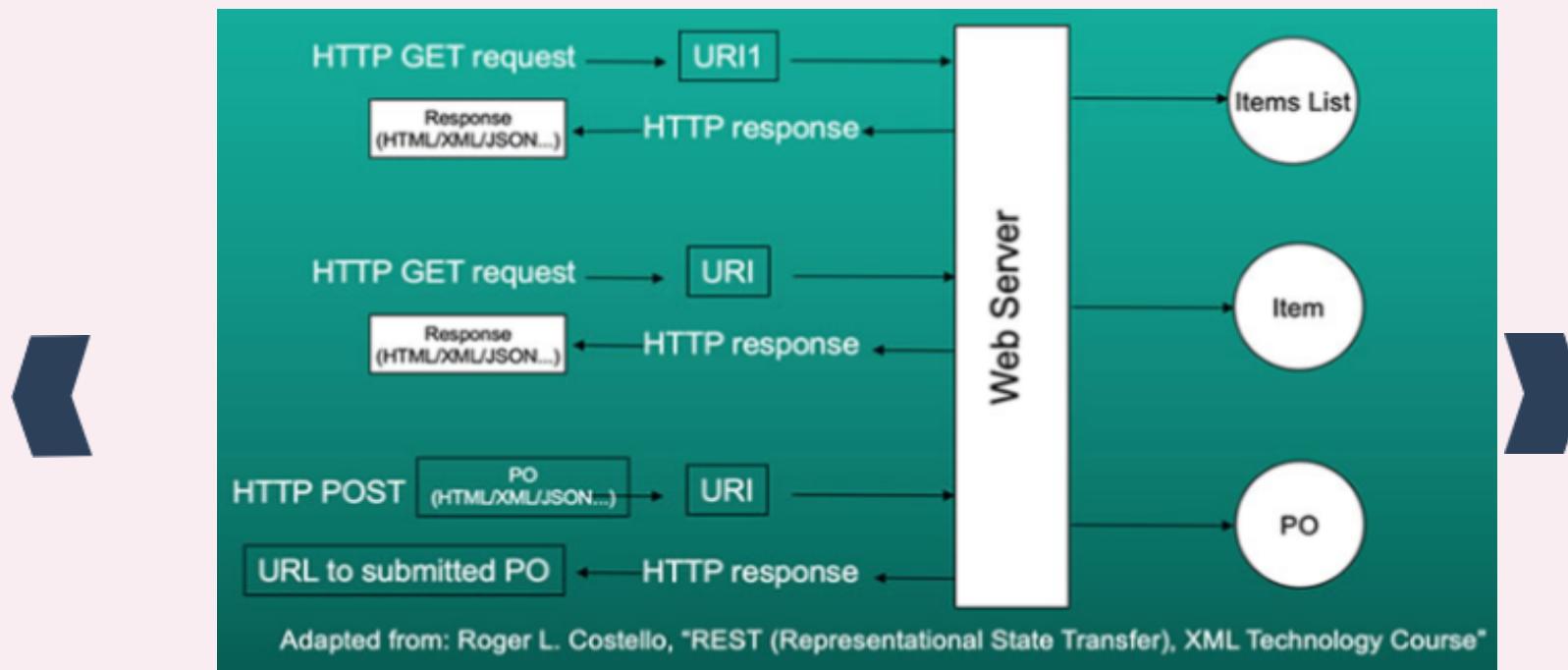
Un recurso Web es una representación del mismo recurso que es retornado. Lo hace a petición del cliente y es retornado en uno de los múltiples tipos MIME, los más comunes de estos tipos durante mucho tiempo fueron text/plain y text/html, el recurso retornado en formato JSON debe ser interpretado en el cliente web de acuerdo al tipo MIME seleccionado, por ejemplo, una imagen image/png, un documento application/pdf, una presentación application/vnd.ms-powerpoint. Una lista completa de Common MIME Types se la puede revisar en (Common MIME Types - HTTP | MDN, s. f.). Recuerde que a pesar de ser JSON el formato de datos más usado, no es el único existente.



2.4.2 El estilo arquitectónico REST

Se podría decir que REST describe al web de tal forma que mapea lo que ya existía y aclara varios puntos (en total 6, que se mencionan más adelante en la sección RESTful Web Services). En otras palabras, REST vino a poner orden en el desarrollo Web, a describir algo que ya existía. REST está siendo usado en la industria para delinear la creación de APIs web confiables y stateless. Esencialmente, un cliente solicita a un servidor web un recurso o un procesamiento, el servicio web retorna un recurso en formato JSON u otro, ninguno de los dos, tanto el cliente como el servicio web necesitan conocer como esta implementado el otro, tampoco es necesario conocer como están implementados los otros servicios. Un servicio web, podría convertirse en cliente de otro servicio Web a la vez.

Figura 9
Arquitectura REST



Las arquitecturas anteriormente mostradas, muestran dos tecnologías para desarrollo de aplicaciones Web, estas son Java y .Net, estas herramientas pueden ser reemplazadas por cualquiera de las tecnologías del lado del servidor estudiadas en el apartado anterior (tema 1).

2.4.3 Métodos HTTP

No todos los servicios retornan un recurso. Algunos servicios recibirán recursos para actualizarlos en una base de datos por ejemplo. Otras peticiones pueden requerir que un recurso sea actualizado o eliminado de la base de datos del sistema. Roy Fielding recomienda usar cuatro métodos HTTP para mapear con las operaciones CRUD (Create, Read, Update, Delete) de una base datos:

Tabla 2

Mapeo métodos HTTP vs operación CRUD

Método HTTP	Operación CRUD (SQL)
GET	SELECT * FROM table ... WHERE ...
PUT	UPDATE table SET ... WHERE ...
POST	INSERT INTO table ... VALUES ...
DELETE	DELETE FROM ... WHERE

Se considera que con estos cuatro métodos y la URI apropiada se puede resolver todos los problemas de un sistema transaccional.



2.4.4 URI vs. URL

A pesar de que una URI y una URL se miran de la misma manera, y puede haber confusión al verlas, la diferencia básica radica en lo que retorna cada una de ellas luego de ser ejecutada. En el caso de una URI, se asume que fue invocada desde un cliente y un servicio web retorna un recurso o una respuesta a la recepción de un recurso. En el caso de una URL, al realizar la petición (request) el cliente, el servidor web retorna un archivo o un conjunto de archivos al cliente que lo solicito, este archivo será una página HTML y sus correspondientes archivos de ser necesario. En otras palabras, la respuesta de una URL es renderizada en el cliente web utilizado, una URI, por otro lado, retorna un recurso en el formato solicitado, por ejemplo, JSON, XML, IMG, etc. A continuación, se presenta una lista de URIs que retornan información en formato JSON, por favor siéntase libre de probarlas, pero recuerde, que como toda API pública, existe la posibilidad de que algún momento desaparezcan y se deberá buscar otras opciones para probarlas. Como todas las URIs son de tipo GET, se las puede ejecutar con un browser y nada más. En caso de ser PUT, POST o DELETE, se deberá instalar un plugin para el navegador capaz de ejecutar estos métodos, o se podría también instalar una aplicación, por ejemplo, POSTMAN, otra posible opción sería utilizar el comando curl (Command URL), este comando viene disponible en Linux, pero se lo puede instalar para DOS o para MacOS:

- <https://httpbin.org/get>
- <https://httpbin.org/ip>
- <https://httpbin.org/anything>
- <https://httpbin.org/post>
 - curl -iX POST <https://bin.org/post>
 - postman app
- <https://httpbin.org/response-headers?key=val>
- <https://httpbin.org/user-agent>

A pesar de que una URI y una URL se miran de la misma manera, y puede haber confusión al verlas, la diferencia básica radica en lo que retorna cada una de ellas luego de ser ejecutada. En el caso de una URI, se asume que fue invocada desde un cliente y un servicio web retorna un recurso o una respuesta a la recepción de un recurso. En el caso de una URL, al realizar la petición (request) el cliente, el servidor web retorna un archivo o un conjunto de archivos al cliente que lo solicita, este archivo será una página HTML y sus correspondientes archivos de ser necesario. En otras palabras, la respuesta de una URL es renderizada en el cliente web utilizado, una URI, por otro lado, retorna un recurso en el formato solicitado, por ejemplo, JSON, XML, IMG, etc. A continuación, se presenta una lista de URIs que retornan información en formato JSON, por favor siéntase libre de probarlas, pero recuerde, que como toda API pública, existe la posibilidad de que algún momento desaparezcan y se deberá buscar otras opciones para probarlas. Como todas las URIs son de tipo GET, se las puede ejecutar con un browser y nada más. En caso de ser PUT, POST o DELETE, se deberá instalar un plugin para el navegador capaz de ejecutar estos métodos, o se podría también instalar una aplicación, por ejemplo, POSTMAN, otra posible opción sería utilizar el comando curl (Command URL), este comando viene disponible en Linux, pero se lo puede instalar para DOS o para MacOS:

- <https://httpbin.org/get>
- <https://httpbin.org/ip>
- <https://httpbin.org/anything>
- <https://httpbin.org/post>
 - curl -iX POST <https://bin.org/post>
 - postman app
- <https://httpbin.org/response-headers?key=val>
- <https://httpbin.org/user-agent>
- <https://httpbin.org/put>

La ejecución de una URI se mira así:

<http://universities.hipolabs.com/search?country=Uzbekistan>

Figura 11

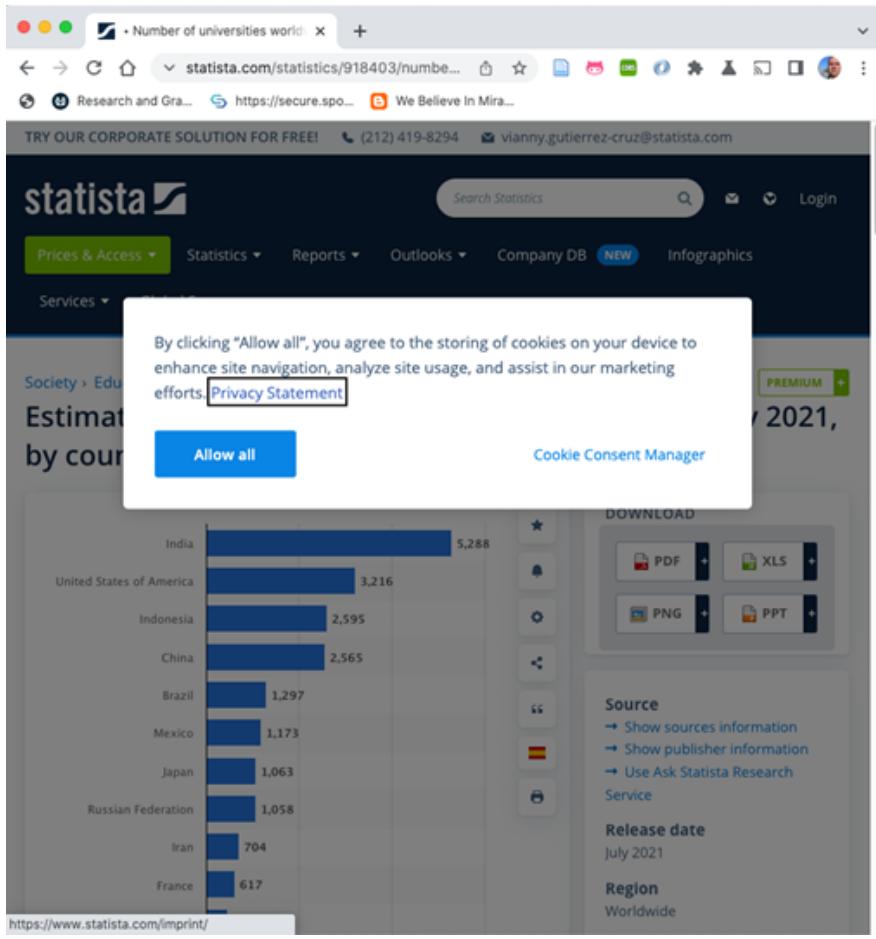
JSON Response

La ejecución de una URL se mira así:

<https://www.statista.com/statistics/918403/number-of-universities-worldwide-by-country/>

Figura 12

Ejemplo página web (URL)



Cuando se programa una API, esta se compone de una serie de URIs, las cuales deben ser documentadas apropiadamente para que el desarrollador del cliente Web que las consuma pueda usarlas sin inconvenientes. Entre otros elementos la documentación de cada URI debe poseer para poder ser usada son: La URI + el o los parámetros { }, el formato de datos usado por retornar o recibir el recurso (JSON, XML, CSV, otros), el tipo de método (GET, PUT, POST, DELETE), la respuesta -response, el envío de datos -request (en caso de ser POST, PUT), opcional el código http retornado (202, 404, ... ver Tabla 3 para algunos ejemplos de estos códigos), a continuación dos ejemplos de documentación de URIs (más ejemplos se encuentran en el contenido de la unidad 1, tema 2):

Figura 13

Documentación URI: Consulta días faltantes para fecha de vuelo

9. Consultar días faltantes hasta fecha del vuelo

Método	GET
URI	http://(ip):8080/SalesCompany/sales/service/calculateDate/{idTicket}
Parámetros	idTicket: id del ticket
Formato	JSON
Devuelve	<pre>{ "FechaDeVuelo": "2019-06-19", "DiasFaltantes": 28, "FechaActual": "2019-5-21" }</pre>

Figura 14

Documentación URI: Ingreso de vendedor (seller)

7. Ingreso Seller

Método	POST
URI	http://(ip):8080/SalesCompany/sales/service/seller
Parámetros	<pre>{ "address": "87949 Bunker Hill Circle", "email": "mryall4@technorati.com", "idSeller": 140, "lastName": "Ryall", "name": "Micheline", "pendingValues": 0, "telephone": "0975864757", "userName": "mryall", "password": "12345" }</pre>
Formato	JSON
Devuelve	{"http-code":201}

Figura 15

Documentación URI: ver preguntas de un foro, añadir preguntas de un foro

URI: <http://www.example.com:8080/forum/questions/{tag}/{name}/{order}>

- Data format: JSON
- GET : Response?, Request?
 - Request: {tag}/{name}/{order}
 - Response:

```
{  
    "tag": "Networking", "name": "edi", "order": "newest"  
}
```
- 404
{"error": "404", "errorMessage": "No tags found"}
- POST : Request? Response?
 - Request : JSON data

```
{  
    "tag": "Networking", "name": "edi", "order": "newest"  
}
```
 - Response
 - http code: 202

Figura 16

Documentación URI: buscar cakes por id

URI GET cakes by id

Method -> Get

Format -> JSON

Request -> id = this is the id of the cake in the database, that return in json format

Response -> {"id":2,"name" : "chocolate cake","type" : "cake",price": 22,50,"quantity" : 12 }

URI -> <https://LaReina.com/cakes/{id}>

Figura 17

Documentación de URI: Conseguir información de un granjero por su id

Method → GET

URI → <https://www.cowbook.com/farmer/{farmerID}/{json}> -> Uri to get farmer information by id.

Response → JSON to return all the information of the farmer by id. Example: {"farId":1, "farIdentification": "1725043887", "farName": "Edison Chinlle", "farBirthDate": "2000-09-05", "farAddress": "Sangolqui", "farPhone": "0987654321", "farEmail": "eschinlle@espe.edu.ec"}

Request → param (farmerID) where farmerID is the unique identification that has the farmer in the system, param (JSON) that is the format of the response data, could be another format like XML, CSV, TXT, etc.

Figura 18

Documentación de URI: Obtener residentes

URI get residents

Method -> GET

Format -> JSON

Request -> residentId=this is the id of the resident in database Mongodb atlas
form = this is the format the response data, its JSON
{residentId}

Response -> resident has = id, name, ci, address, born date, number house, telephone

```
{"id":123,"name":"Erick","ci":"1725156218","address":"Calixto Muzo","bornDate":"19-03-2000","numberHouse":"10","telephone":"0984151434"}
```

<https://www.Manatial.com:8080/resident/{residentId}>

<https://www.Manatial.com:8080/resident/?id=123#top>

```
{"error": "404", "errorMessage": "Report not found"}
```

2.4.5 RESTful Web Services



Un servicio REST es aquel que retorna un recurso a petición de un cliente a través de una URI, “un servicio Web RESTful, es aquel que cumple con las 6 restricciones REST” Roy Fielding.

Tres restricciones del cliente:

1. Se debe usar la arquitectura Cliente/Servidor para la separación de concerns, cada componente del sistema se implementa y se ofrece por separado.
2. Debe ser stateless, el cliente maneja el estado de la sesión en una comunicación C/S
3. Es cacheable, la página se puede almacenar temporalmente en memoria, para mejorar la escalabilidad y el rendimiento. Pero se debe tener cuidado con mostrar/ver información desactualizada
4. Tres restricciones del código:
5. Sistema en capas, cada capa no sabe nada acerca de la implementación de la otra, el URI es suficiente.
6. Código bajo demanda, un poco de JavaScript o ningún JavaScript (este constrain es opcional).
7. Interface Uniforme, para todos los servicios web.

2.4.6 URIs nominales

Se debe una Interface Uniforme RESTful:

- Usar los códigos (familias) HTTP Request/Response
 - 1xx Información:
 - Proceso recibido y continúa
 - 100, 101
 - 2xx OK:
 - La ejecución fue recibida, entendida y aceptada
 - 200, 201, 202, 203, 204, 205, 206
 - 3xx Redirect:
 - Se necesita una acción adicional para completar el request.
 - 300, 301, 302, 303, 304, 305, 306, 307
 - 4xx Client Error, e.g., 404 Not found
 - El request contiene alguna sintaxis incorrecta
 - 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417
 - 5xx server error, e.g., syntax error
 - El servidor fallo en responder a un request válido
 - 500, 501, 502, 503, 504, 505
- Las URIs deben ser nominales, por ejemplo:
 - <http://www.example.com/resources>
 - retorna todos los recursos del sistema

Se debe una Interface Uniforme RESTful:

- Usar los códigos (familias) HTTP Request/Response
 - 1xx Información:
 - Proceso recibido y continúa
 - 100, 101
 - 2xx OK:
 - La ejecución fue recibida, entendida y aceptada
 - 200, 201, 202, 203, 204, 205, 206
 - 3xx Redirect:
 - Se necesita una acción adicional para completar el request.
 - 300, 301, 302, 303, 304, 305, 306, 307
 - 4xx Client Error, e.g., 404 Not found
 - El request contiene alguna sintaxis incorrecta
 - 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417
 - 5xx server error, e.g., syntax error
 - El servidor fallo en responder a un request válido
 - 500, 501, 502, 503, 504, 505
- Las URIs deben ser nominales, por ejemplo:
 - <http://www.example.com/resources>
 - retorna todos los recursos del sistema
 - <http://www.example.com/resources/item1>
 - retorna el recurso cuyo id es item1
 - <http://www.example.com/resources/item1/json>
 - retorna el recurso item2 en formato JSON
 - <http://www.example.com/resources/item1?format=json>
 - retorna el recurso item1 en formato JSON, pero en la URI utiliza un QUERY_STRING
- Se debe usar modelar las URIS utilizando 4 métodos HTTP, se puede pasar argumentos en el URI, pero es recomendable que los datos viajen como parte del cuerpo e la página web que hace la petición, el único método que no soporta este tipo de envío de información es el método GET, si no se usa un QUERY_STRING (?key=value&key2=value2), se puede usar la forma nominal por medio de sustantivos y el argumento/parámetro e escribe en paréntesis en el diseño y en varios de los lenguajes de programación de la misma forma . Se puede utilizar la misma URI para las cuatro operaciones básicas, por ejemplo:

- GET /ítems/{id} lee (Read) un ítem en base a id
- POST /ítems/{id} crea (Create) un ítem en base a id
- PUT /ítems/{id} actualiza (Update) un ítem en base a id
- DELETE /ítems/{id} Elimina (Delete) un ítem en base a id

En los casos de POST y PUT, la información a ser insertada y/o actualizada viaja en el cuerpo de la página Web, en muchos casos por cuestión de seguridad, inclusive estos datos deberían enviarse encriptados, lo cual no se podría hacer o no sería recomendable como parte de la URI.

A continuación una serie de consejos acerca de REST

- Utilice los métodos HTTP estándar
- Mejore su diseño REST: provea una URI para cada recurso, minimice el uso de QUERY_STRING, prefiera *items/1* a *items?id=1*, use URLs lógica antes que físicas, prefiera *items/1* a *items/1.html*, no use verbos en las URLs, recuerde las URIs representan recursos, no acciones.
- Utilice hipervínculos en la respuesta (Response) cuando necesite comunicar una acción de asentimiento (ack/nack)
- Utilice / para representar relaciones padre/hijo, e.g., *stock/item1*, *doctor1/patients/1*.
- Utilice GET para permitir que el cliente recupere recursos, no use POST en estos casos
- Utilice POST/PUT para actualizar un objeto grande (con bastantes datos), así la URI no será muy larga, y sus datos no serán vistos en al navegador web a simple vista.
- Utilice GET, PUT, POST, DELETE apropiadamente

La Tabla 3 muestra algunos códigos de estatus http, para ver una lista completa de los códigos, refiérase a (*HTTP - Status Codes*, s. f.)

Tabla 3

Ejemplos de HTTP status codes

Mensaje	Descripción
---------	-------------

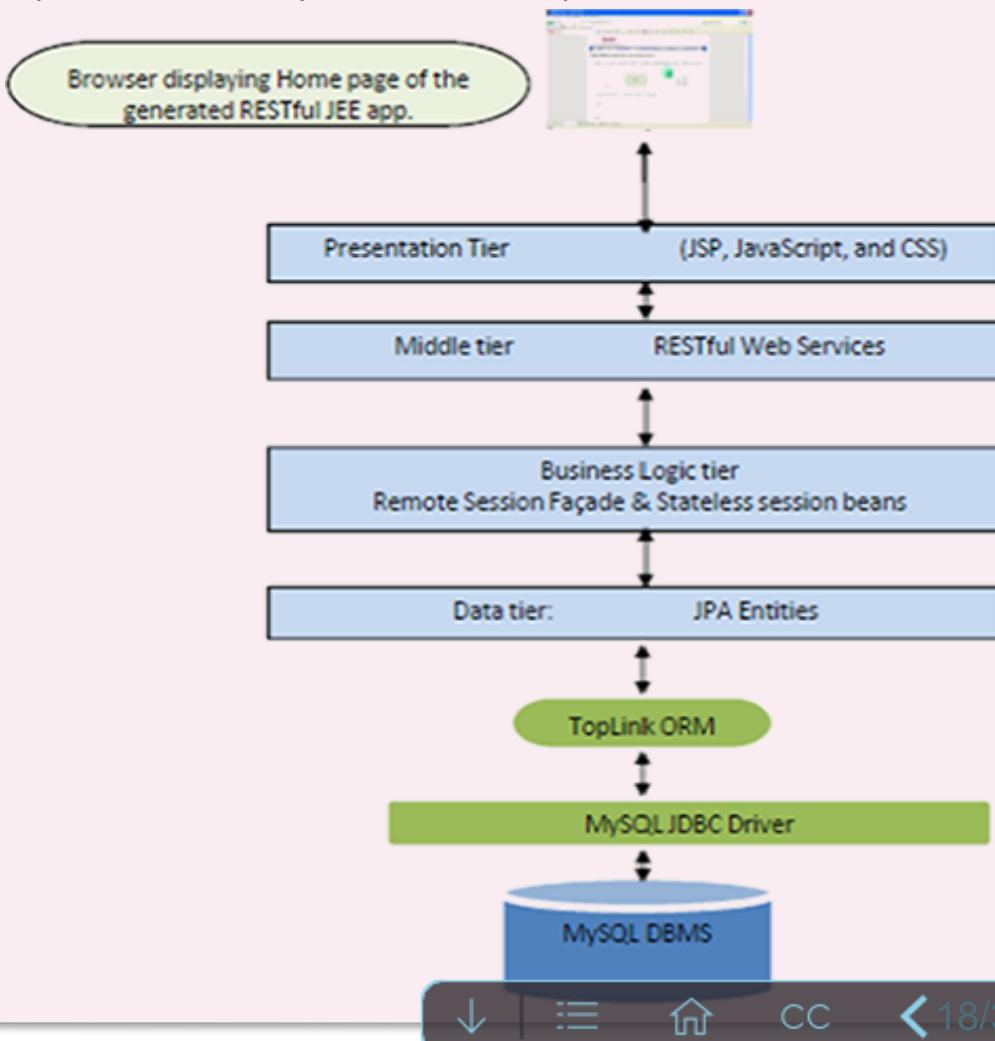
Mensaje	Descripción
100 Continue	Solo una parte del request ha sido recibida por el servidor, pero siempre y cuando no haya sido rechazada, el cliente debe continuar con el proceso
101 Switching Protocols	El servidor cambia entre protocolos
200 OK	El request esta OK
201 Created	El request esta completo, y se ha creado un nuevo recurso
202 Accepted	El request es aceptado para procesamiento, pero el procesamiento no esta completo
203 Non Authoritative Information	La información en la cabecera de la entidades de un acopia de terceros o es local, no desde el servidor original
300 Multiple Choices	Una lista de enlaces, el usuario puede seleccionar un enlace para ir a esa localización, máximo cinco direcciones
301 Moved Permanently	La página requerida (requested) ha sido movida a un nuevo URL
302 Found	La página requerida (requested) ha sido movida temporalmente a un nuevo URL
303 See Other	La página requerida (requested) puede ser encontrada en un diferente URL
400 Bad Requests	El servidor no entendió el request
401 Unauthorized	La página requerida (requested) necesita username y password
402 Payment Required	No puede usar este código todavía
403 Forbidden	El acceso es prohibido a la página requerida (requested)
404 Not Found	El servidor no puede encontrar la página requerida (requested resource)

Mensaje	Descripción
500 Internal Server Error	El request no fue completado. El servidor encontró una condición inesperada
501 Not Implemented	El request no fue completado. El servidor no soportó la funcionalidad requerida
502 Bad Gateway	El request no fue completado. El servidor recibió una response inválida del servidor previo
503 Service Unavailable	El request no fue completado. El servidor esta temporalmente sobrecargado o esta abajo.
504 Gateway Timeout	Se acabó el tiempo del Gateway
505 HTTP Version Not Supported	El servidor no soporta la versión del “protocolo http”

2.4.7 Tecnologías para desarrollo REST

Figura 19

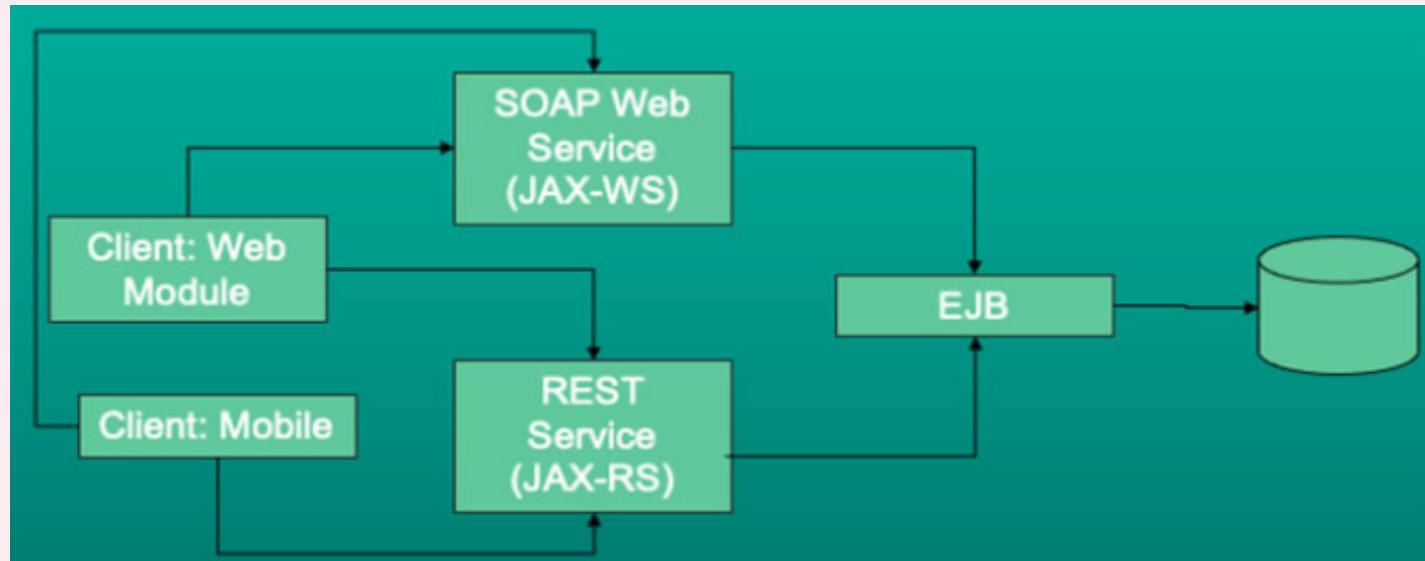
Arquitectura de una aplicación web n-capas



Los servicios REST se pueden programar con tecnologías el lado del servidor, por ejemplo: Java, PHP, node.js, ASP.Net, Ruby, etc. El siguiente ejemplo asume el uso de Java para el desarrollo de los servicios REST. El lado del cliente mantiene las 3 tecnologías usuales, a continuación de esta capa, se tiene la capa de Servicios REST que es lo único que ve el cliente Web a través de las URLs proveídas, como el servicio REST es programado en Java, la parte que el cliente no ve esta programada usando Session Beans y JPA Entities, los session beans son usados en el mundo de Java para programar la lógica del negocio, mientras que las JPA entities realizan el mapeo objeto relacional entre los objetos del sistema y las tablas de la base de datos a ser usada. Para conectarse a la base de datos, en esta arquitectura, se propone el uso de un framework llamado TopLink ORM, el cual se comunica con la base de datos MySQL a través del respectivo Driver. En esta arquitectura se utiliza tecnología Java con servicios REST.

2.4.7 Tecnologías para desarrollo REST

Otra posible arquitectura usando Java, pero dos tipos diferentes de servicios web (SOAP y REST) se presenta a continuación. Vale la pena mencionar, que a pesar de que los servicios web tipo SOAP no son muy usados en la actualidad, pero son útiles en casos de aplicaciones web hacia el interior de la empresa por ejemplo.



2.4.7 Tecnologías para desarrollo REST

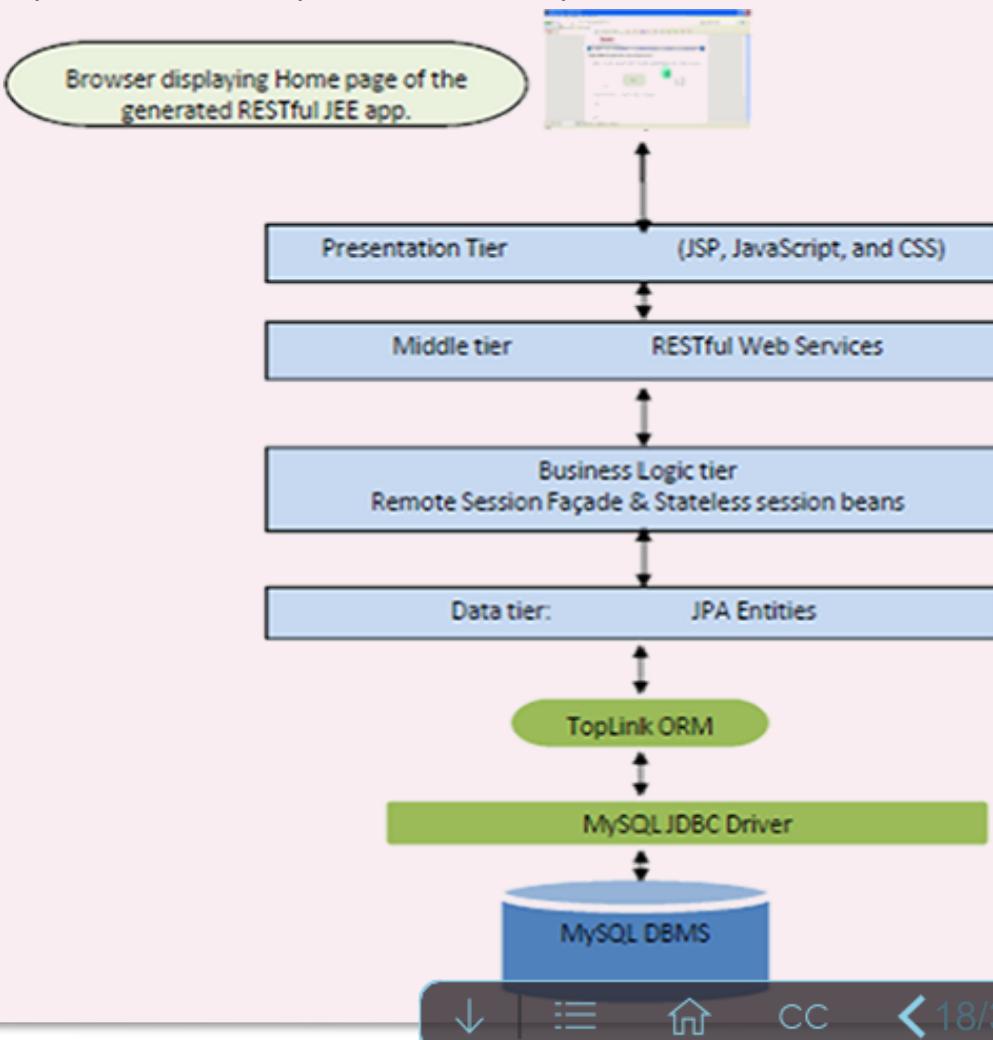
En la actualidad (año 2022), y a pesar de que Java sigue dominando un buen porcentaje de desarrollo web, ya no es la tecnología preferida de muchas empresas, por lo tanto, es importante entender los conceptos aquí vertidos, en lugar de concentrarse al 100% en una tecnología en particular. A la fecha, el lenguaje más popular para desarrollo de aplicaciones y servicios es node.js. pero existen otras posibilidades. Una de las diferencias entre desarrollar en Java y desarrollar en node.js, es que para programar en Java, se necesita haber instalado un servidor web que soporte Java, como por ejemplo, Apache Tomcat, Payara. Mientras que para desarrollar en node.js sólo hace falta usar el framework Express.js, el cual provee de un servidor web en el código del servicio a desarrollar. Lo cual hace que no sea necesario instalar un servidor Web para iniciar el desarrollo. En la sección final de este tema, se presenta el desarrollo de un servicio web REST desarrollado con node.js



2.4.7 Tecnologías para desarrollo REST

Figura 19

Arquitectura de una aplicación web n-capas

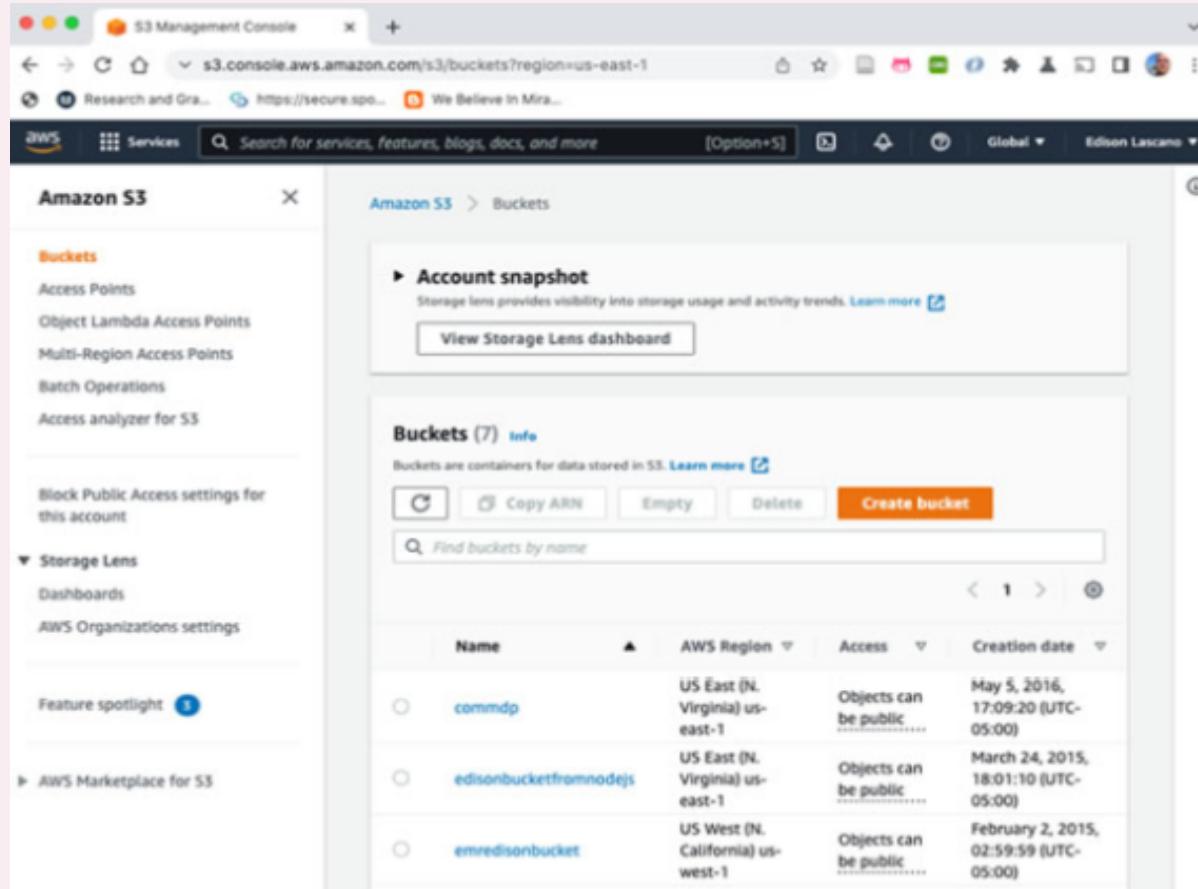


Los servicios REST se pueden programar con tecnologías el lado del servidor, por ejemplo: Java, PHP, node.js, ASP.Net, Ruby, etc. El siguiente ejemplo asume el uso de Java para el desarrollo de los servicios REST. El lado del cliente mantiene las 3 tecnologías usuales, a continuación de esta capa, se tiene la capa de Servicios REST que es lo único que ve el cliente Web a través de las URIs proveídas, como el servicio REST es programado en Java, la parte que el cliente no ve esta programada usando Session Beans y JPA Entities, los session beans son usados en el mundo de Java para programar la lógica del negocio, mientras que las JPA entities realizan el mapeo objeto relacional entre los objetos del sistema y las tablas de la base de datos a ser usada. Para conectarse a la base de datos, en esta arquitectura, se propone el uso de un framework llamado TopLink ORM, el cual se comunica con la base de datos MySQL a través del respectivo Driver. En esta arquitectura se utiliza tecnología Java con servicios REST.

2.5 Almacenamiento de Datos en la Nube

Figura 21

Servicio S3 de AWS



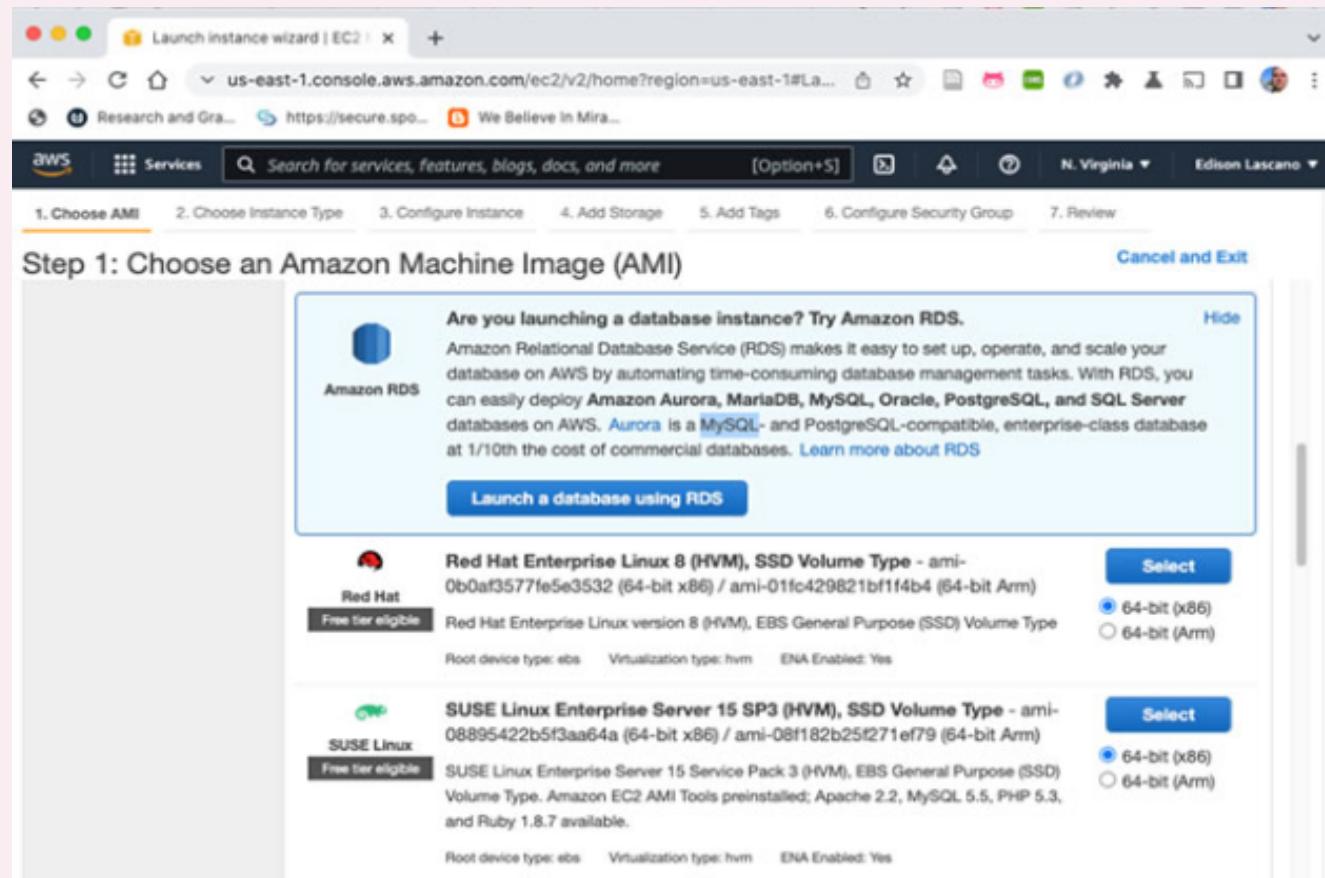
The screenshot shows the AWS S3 Management Console interface. The left sidebar has a 'Buckets' section with options like Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and Access analyzer for S3. Below that are sections for Block Public Access settings, Storage Lens, Dashboards, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3. The main content area shows an 'Account snapshot' with a link to the Storage Lens dashboard. Below it is a 'Buckets (7) info' section with a table listing seven buckets. The table columns are Name, AWS Region, Access, and Creation date. The buckets listed are:

Name	AWS Region	Access	Creation date
commdp	US East (N. Virginia) us-east-1	Objects can be public	May 5, 2016, 17:09:20 (UTC-05:00)
edisonbucketfromnodejs	US East (N. Virginia) us-east-1	Objects can be public	March 24, 2015, 18:01:10 (UTC-05:00)
emredisonbucket	US West (N. California) us-west-1	Objects can be public	February 2, 2015, 02:59:59 (UTC-05:00)
(3)			
(2)			
(1)			

Los datos en la nube se pueden almacenar como archivos o como bases de datos. (1) Si se los almacena como archivos, se tendría un disco duro virtual atado a mi servidor web (unidad de disco duro virtual, por ejemplo un S3) Figura 21, y se debería programar el manejo de datos con un feature de administración de archivos, por ejemplo archivos planos, archivos JSON, etc. (2) La segunda opción sería obtener una máquina virtual en la nube y en esta máquina instalar un motor de bases de datos, por ejemplo una máquina con Linux e instalar MySQL o una máquina virtual con el sistema de gestión de base de datos pre instalado, Figura 22. (3) La tercera opción sería adquirir los servicios de un base de datos en la Nube, por ejemplo MongoDB Atlas y acceder a sus datos usando un programa cliente como Mongo Compass, ver Figura 23.

2.5 Almacenamiento de Datos en la Nube

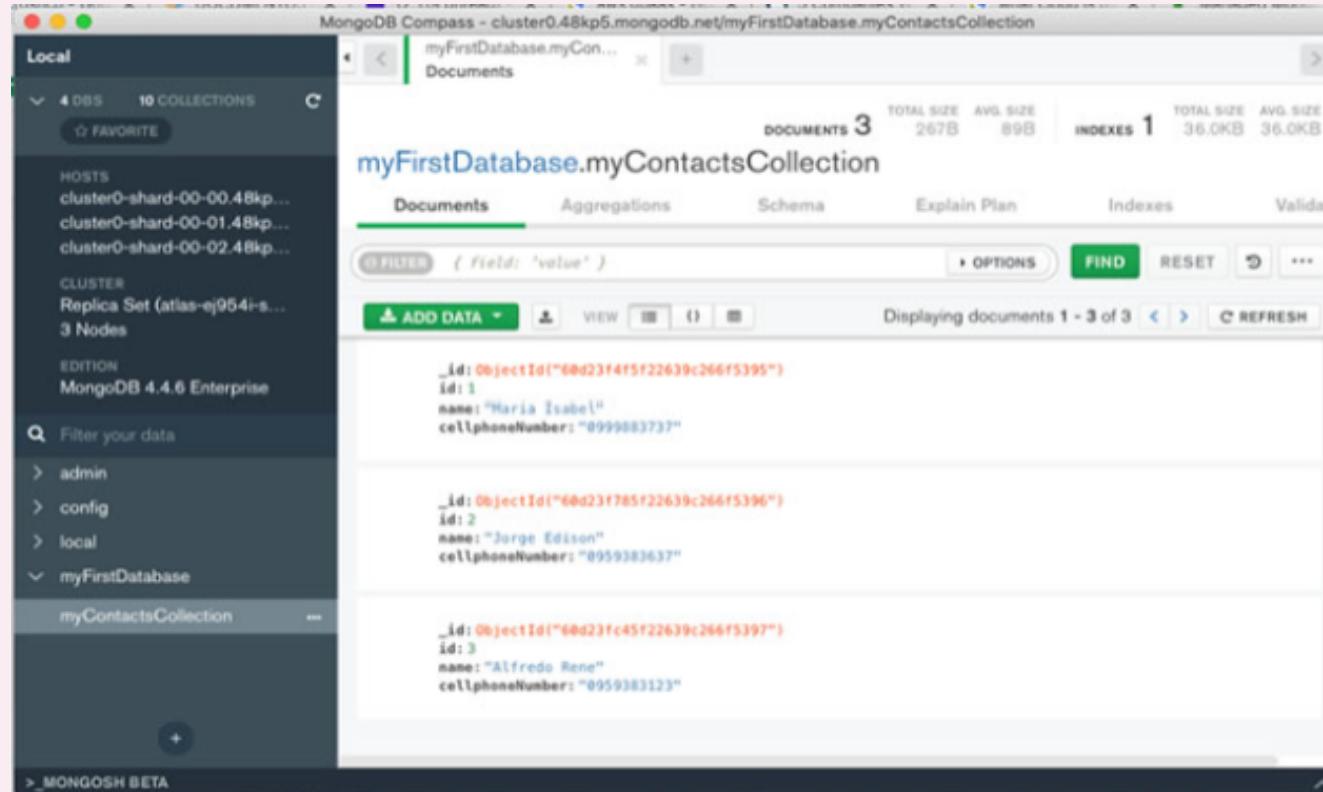
Figura 22
PaaS de AWS



2.5 Almacenamiento de Datos en la Nube

Figura 23

Mongo Compass



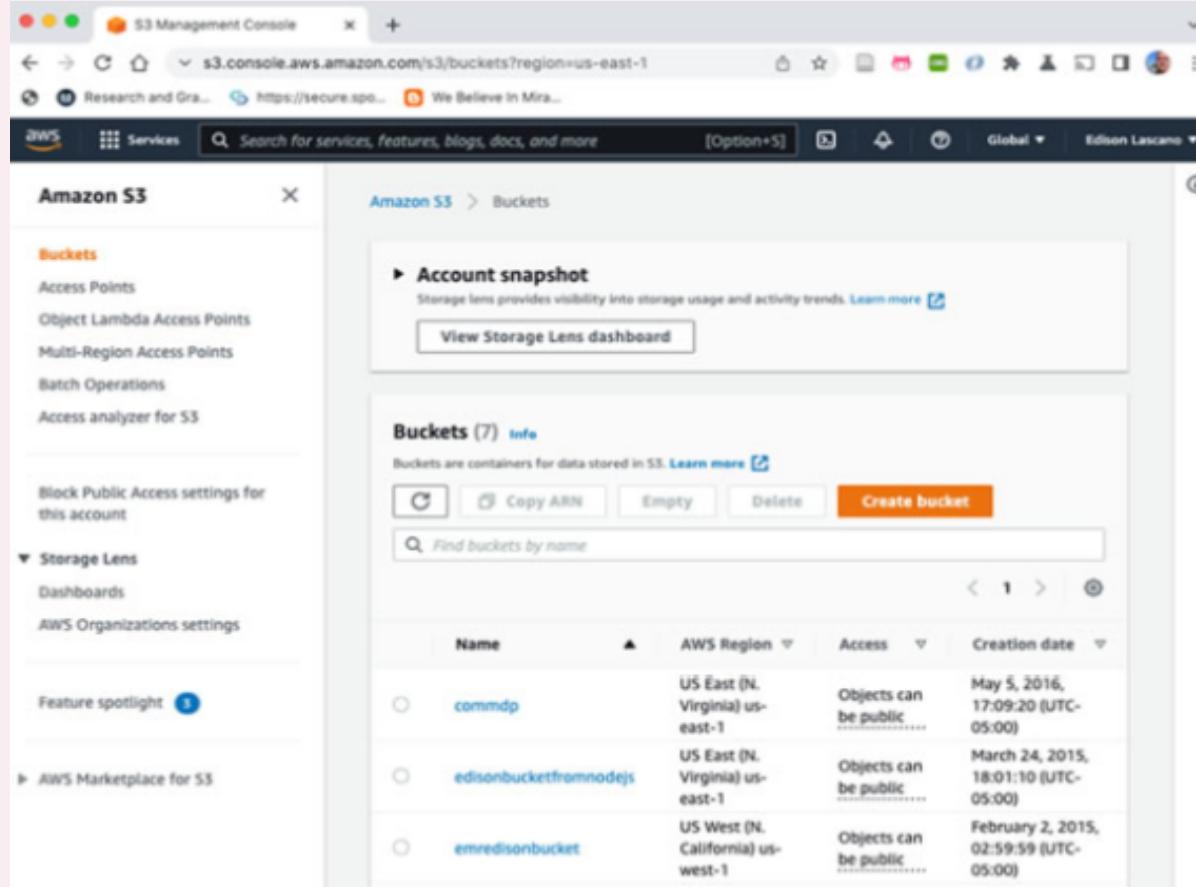
The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays the Local connection with 4 DBs and 10 collections. The 'myFirstDatabase' section is expanded, showing the 'myContactsCollection' collection. The main pane shows the 'Documents' tab for this collection, which contains three documents:

- Document 1:
_id: ObjectId("60d23f4f5f22639c266f5395")
id: 1
name: "Maria Isabel"
cellphoneNumber: "0999883732"
- Document 2:
_id: ObjectId("60d23f785f22639c266f5396")
id: 2
name: "Jorge Edison"
cellphoneNumber: "0959383637"
- Document 3:
_id: ObjectId("60d23fc45f22639c266f5397")
id: 3
name: "Alfredo Rene"
cellphoneNumber: "0959383123"

2.5 Almacenamiento de Datos en la Nube

Figura 21

Servicio S3 de AWS



The screenshot shows the AWS S3 Management Console interface. The left sidebar has a 'Buckets' section with options like Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and Access analyzer for S3. Below that are sections for Block Public Access settings, Storage Lens, Dashboards, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3. The main content area shows an 'Account snapshot' with a link to the Storage Lens dashboard. Below it is a 'Buckets (7) info' section with a table listing seven buckets. The table columns are Name, AWS Region, Access, and Creation date. The buckets listed are:

Name	AWS Region	Access	Creation date
commdp	US East (N. Virginia) us-east-1	Objects can be public	May 5, 2016, 17:09:20 (UTC-05:00)
edisonbucketfromnodejs	US East (N. Virginia) us-east-1	Objects can be public	March 24, 2015, 18:01:10 (UTC-05:00)
emredisonbucket	US West (N. California) us-west-1	Objects can be public	February 2, 2015, 02:59:59 (UTC-05:00)
(3)			
(2)			
(1)			

Los datos en la nube se pueden almacenar como archivos o como bases de datos. (1) Si se los almacena como archivos, se tendría un disco duro virtual atado a mi servidor web (unidad de disco duro virtual, por ejemplo un S3) Figura 21, y se debería programar el manejo de datos con un feature de administración de archivos, por ejemplo archivos planos, archivos JSON, etc. (2) La segunda opción sería obtener una máquina virtual en la nube y en esta máquina instalar un motor de bases de datos, por ejemplo una máquina con Linux e instalar MySQL o una máquina virtual con el sistema de gestión de base de datos pre instalado, Figura 22. (3) La tercera opción sería adquirir los servicios de un base de datos en la Nube, por ejemplo MongoDB Atlas y acceder a sus datos usando un programa cliente como Mongo Compass, ver Figura 23.

2.5.1 IaaS vs. SaaS

Para entender un poco más las propuestas anteriormente mencionadas, en esta sección hablaremos brevemente de servicios en la nube. El modelo de negocios de los proveedores de servicios en la nube se basa en pay-as-you-go, es como los servicios básicos que se paga en el hogar, tanto consumos de luz, tanto pagas, tanto consumos de agua, tanto pagas. En el caso de la nube, utilizas una computadora por 3 horas, pagas por esas tres horas, accedes a tus datos en tu disco duro virtual en la nube n- veces, pues pagas n-centavos. Este modelo es muy útil cuando se va a realizar proyectos cortos, y la adquisición de máquinas físicas sería un costo en lugar de una inversión, no es tan simple como aquí se lo menciona, pero para nuestro caso, asumiremos que alquilar los recursos de la nube durante una semana o un mes resuelve nuestros problemas de publicación en el Web. Los servicios en la nube presentan tres modelos básicos: Infraestructura como Servicio (IaaS), Plataforma como Servicio (PaaS) y Software como Servicio (SaaS). Existen más modelos, pero en este apartado no se los topará.

IaaS

PaaS

SaaS



IaaS

Es un servicio que ofrece al cliente infraestructura virtual que esta disponible en la nube para que el usuario la tenga al alcance de la mano, lo único que necesita el cliente es una conexión a Internet, ejemplos de este servicio en AWS (Amazon Web Services) es EC2. Cabe anotar que los otros dos grandes proveedores de servicios en la nube Microsoft Azure y Google Cloud ofrecen servicios similares, y se da por descontado que el estudiante esta en capacidad de analizar y comprender que la solución en los otros proveedores es factible. El usuario adquiere el servicio con disco duro, procesador y memoria, es responsabilidad del usuario instalar el software necesario para el desarrollo o para deployar o ejecutar las aplicaciones a desarrollar o desarrolladas. El usuario controla casi toda la infraestructura virtual, y debe instalar su software.



CC

< 20/36 >



2.5.2 Proveedores de datos SQL y NO-SQL en la nube

Los proveedores de servicios en la nube, en general proveen IaaS y PaaS, pero también pueden proveer SaaS, por ejemplo, AWS ofrece AWS RDS, que es un servicio administrado de bases de datos relacionales para MySQL, PostgreSQL, MariaDB, Oracle BYOL, o SQL Server.

Respecto a proveedores de bases de datos en la nube, se tiene diferentes opciones afuera de los tres grandes proveedores, una opción NoSQL es MongoDB Atlas, donde se ofrece una base de datos con baja capacidad, pero completamente gratuita, se puede almacenar hasta 512 MB. Se puede contratar el servicio y pagar por uso.



2.5.3 Drivers y cadenas de conexión



Al igual que para el desarrollo de aplicaciones de escritorio, móviles, y web, se necesita de drivers para conectarse a las bases de datos, sean relacionales o no, adicionalmente se debe conoce la cadena de conexión.

2.5.4 REST + BDD

A continuación se presenta un taller hands-on lab (no evaluado), que el alumno debe realizar antes de implementar la actividad 2 del curso, para aprender tecnologías y lenguajes de programación orientados al desarrollo de servicios REST. El presente proyecto fue desarrollado por Adrián Burgos bajo la supervisión del docente autor Jorge Edison Lascano.

i. Tema

Desarrollo de servicios REST

ii. Objetivo

Desarrollar un API REST que permita la obtención de información de una base de datos No SQL de MongoDB publicada en MongoDB Atlas.

iii. Introducción

¿Qué es un servicio REST?

Es un servicio basado en una interfaz que permite la conexión a varios sistemas mediante el protocolo HTTP (Hypertext Transfer Protocol), podría hacerlo usando otro protocolo.

¿Qué funcionalidades brinda?

Al ser un servicio apoyado por HTTP nos permite que las funciones básicas de un CRUD (Create/ POST, Read/GET, Update/PUT, Delete/DELETE), se puedan ejecutar desde cualquier cliente Web.

A continuación se presenta un taller hands-on lab (no evaluado), que el alumno debe realizar antes de implementar la actividad 2 del curso, para aprender tecnologías y lenguajes de programación orientados al desarrollo de servicios REST. El presente proyecto fue desarrollado por Adrián Burgos bajo la supervisión del docente autor Jorge Edison Lascano.

i. Tema

Desarrollo de servicios REST

ii. Objetivo

Desarrollar un API REST que permita la obtención de información de una base de datos No SQL de MongoDB publicada en MongoDB Atlas.

iii. Introducción

¿Qué es un servicio REST?

Es un servicio basado en una interfaz que permite la conexión a varios sistemas mediante el protocolo HTTP (Hypertext Transfer Protocol), podría hacerlo usando otro protocolo.

¿Qué funcionalidades brinda?

Al ser un servicio apoyado por HTTP nos permite que las funciones básicas de un CRUD (Create/ POST, Read/GET, Update/PUT, Delete/DELETE), se puedan ejecutar desde cualquier cliente Web.

¿Por qué debemos utilizar REST?

Uno de los motivos más importantes por el que deberíamos usarlo es que contiene toda la información necesaria tomada de un servidor gracias a la publicación de URIs y una respuesta (response). Además, se apoya de un protocolo que está siendo altamente utilizado, y no solamente por páginas web modernas, sino por cualquier tipo de aplicaciones, por ejemplo: móviles, IoT.

iv. Descripción de la actividad

Se plantea el desarrollo de un API REST mediante node.js, express.js y Mongoose, además, se probará la funcionalidad del servicio REST mediante la herramienta Postman.

v. Análisis

Requisitos Funcionales

- El servicio REST deberá permitir la gestión y manejo de clientes nuevos y existentes en una base de datos, mediante una URI específica para las operaciones básicas CRUD.
- El servicio deberá permitir la comunicación e interacción con una base de datos manejada en MongoDB Atlas.

Requisitos No Funcionales

- El servicio REST deberá estar desarrollado en Node.JS mediante la implementación de módulos para un mejor y fácil desarrollo.
- El servicio deberá ser probado mediante la herramienta PostMan y también de una revisión continua de la base de datos.

- El servicio deberá ser capaz de ser consumido desde un cliente mediante un módulo que se lo implementará más adelante (no en este tema de la asignatura).
- El diseño de las URIs se lo hará de la forma verbal para mayor comprensión del ejercicio, pero recuerdo que en la práctica en el día a día del desarrollo, y en especial en su proyecto, los URIs deben ser nominales, es decir usar sustantivos, por ejemplo <http://mycompany.com/customer/{001}>

vi. Diseño

- Diagrama de clases

Para el ejemplo que se ha decidido desarrollar, únicamente se necesita la clase “Cliente” ya que nuestro servicio REST estará orientado a gestión de clientes

Figura 1

Diagrama de Clases

Object-Oriented Model	
Model: Diagrama de clases REST Clientes	
Package:	
Diagram: Diagrama de clases REST Clientes	
Author: Adrian Burgos	Date: 18/3/2022
Version:	

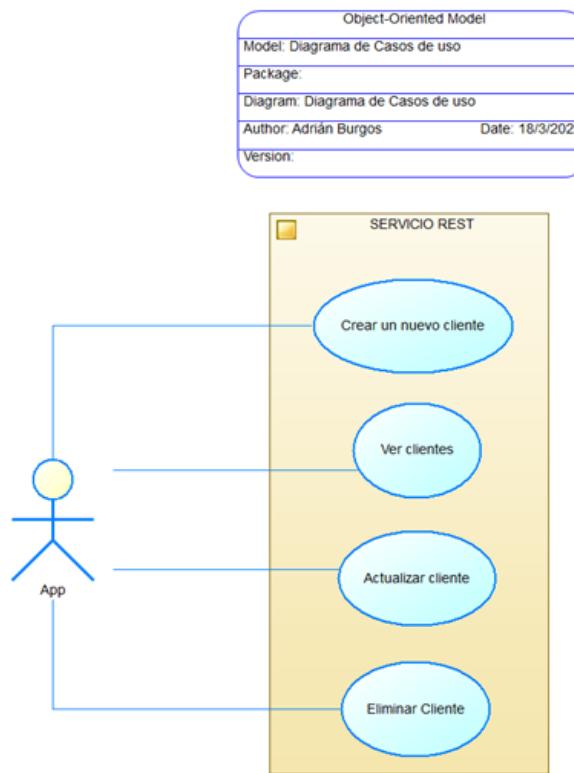
CLIENTE
- nombre : String
- apellido : String
- edad : int
- subTotal : float

- Diagrama de casos de uso

Para el diagrama de casos de uso, se ha planteado que el cliente sea una aplicación móvil ya que en caso de que este servicio REST sea aplicado, debería ser consumida desde un cliente, esto para manejar de manera correcta los recursos de la base de datos a través de los URIs aquí desarrollados. Cabe mencionar, que la implementación del cliente no es parte de este taller.

Figura 2

Diagrama de casos de uso

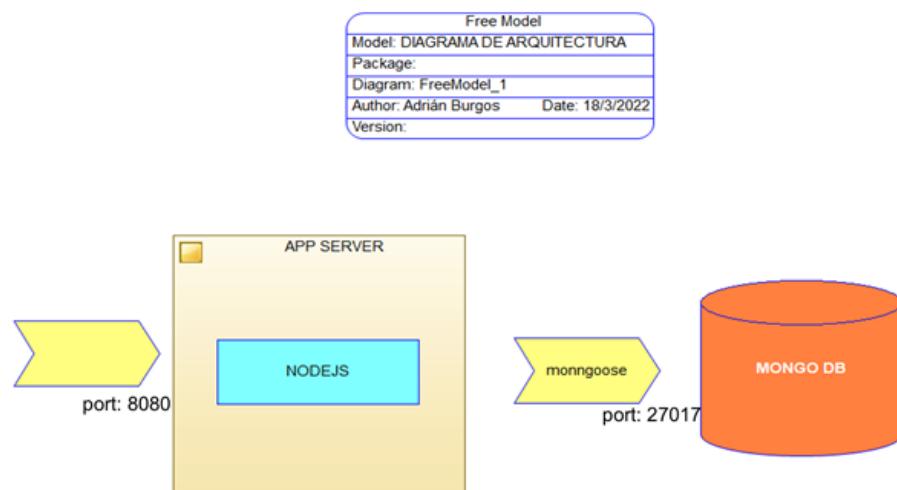


- **Diagrama de la arquitectura**

La arquitectura del ejemplo práctico está basada únicamente en la aplicación del servidor y en la base de datos, esto debido a que se va a desarrollar las APIs REST, mismas que se conectan a una base de datos NoSQL MongoDB en la nube.

Figura 3

Diagrama de arquitectura del servicio REST.



- **URIS**

- **POST / CREAR CLIENTE**

- **URI:** `http://localhost:8080/add`
 - **Body:**

```

    {
        name: String,
        lastName: String,
        age: Number,
        subTotal: Number
    }

    - Response: Cliente registrado en formato JSON
    - Method: POST

    ○ GET / VER CLIENTES
        - URI: http://localhost:8080/clients
        - Body: No recibe ningún dato en el body.
        - Response: Arreglo de los clientes existentes
        - Method: GET

    ○ PUT / ACTUALIZAR SUBTOTAL
        - URI: http://localhost:8080/updateClient
        - Body:
            {
                name: String,
                subTotal: Number
            }

        - Response: Información del cliente actualizado en formato JSON.
        - Method: PUT

    ○ DELETE / ELIMINAR CLIENTE
        - URI: http://localhost:8080/deleteClient
        - Body:
            {
                name: String
            }

        - Response: Información del cliente eliminado en formato JSON.
        - Method: DELETE

```

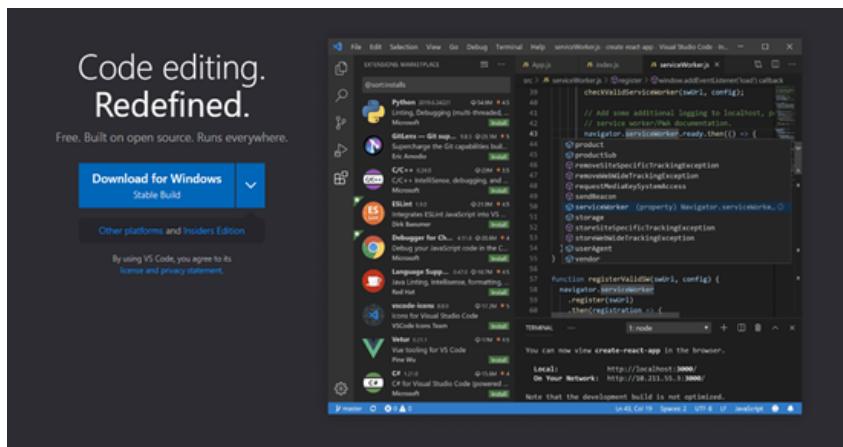
vii. Desarrollo

1. Descarga e instalación del IDE a usar, Visual Studio Code

- Se procede a ingresar a la página oficial de descarga del IDE (<https://code.visualstudio.com/>) y se hace clic en el botón de descarga.

Figura 4

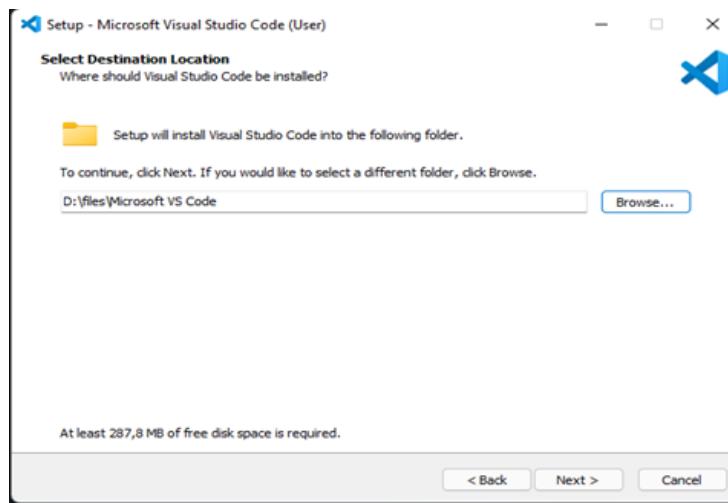
Página de descargar del ide Visual Studio Code



- Una vez descargado el IDE, se procede a instalarlo en el computador.

Figura 5

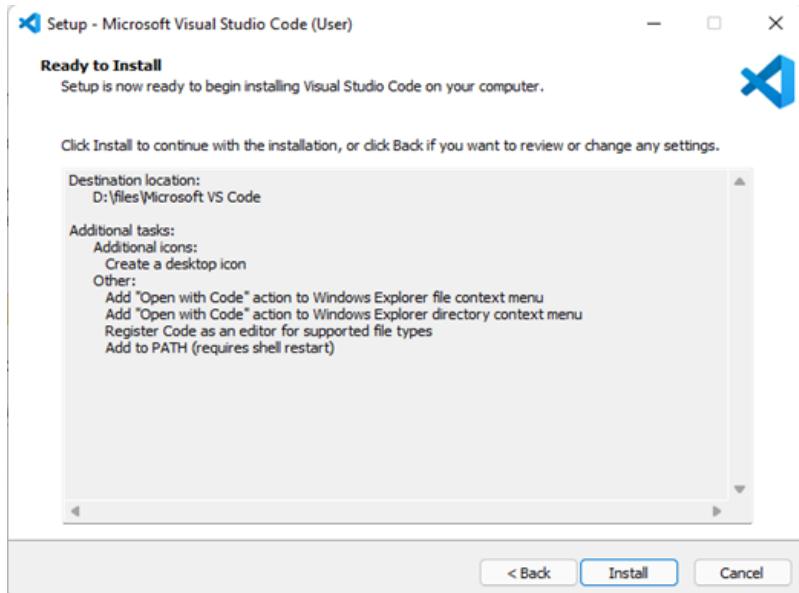
Selección de ubicación de la instalación del IDE



- Se hace clic en el botón **Install** y se espera a que finalice la instalación.

Figura 6

Comienzo de la instalación



d. Listo, ahora se puede utilizar el IDE para programar el servicio REST

Figura 7

Finalización de la instalación de VSCode

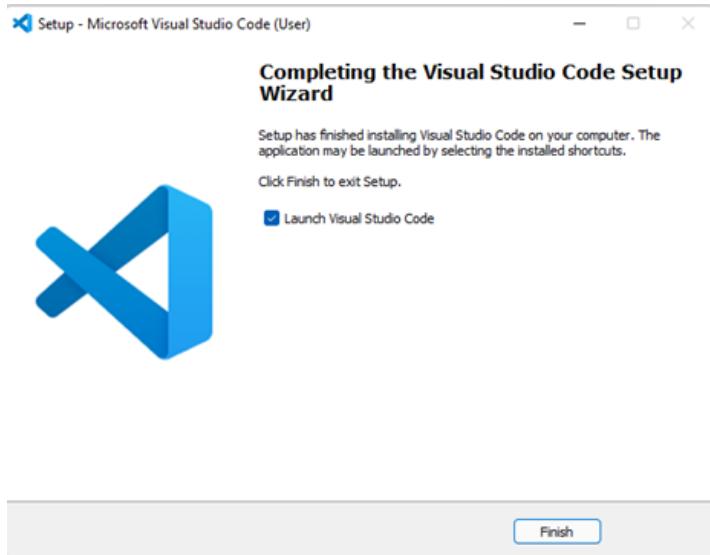
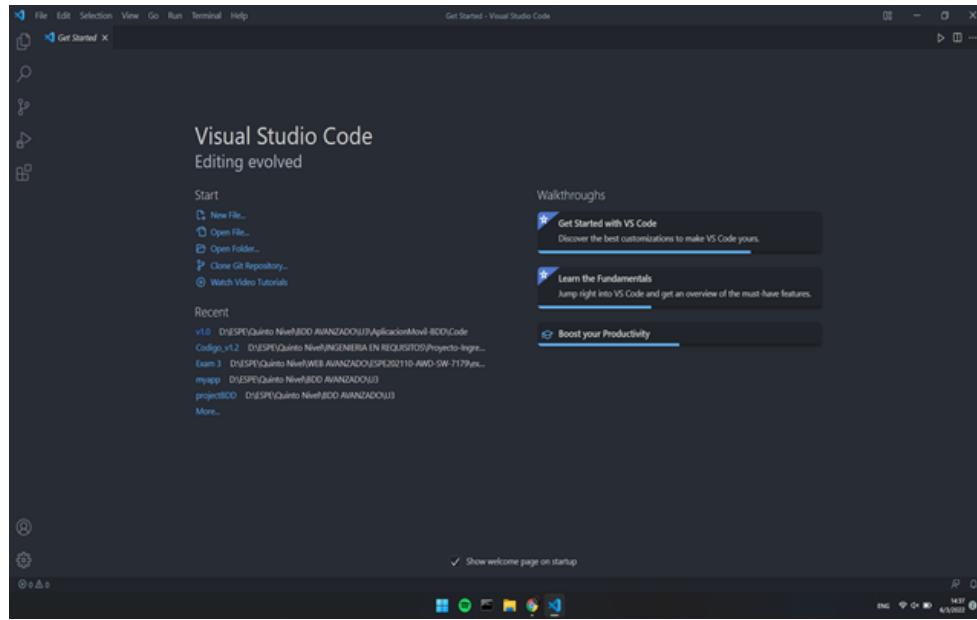


Figura 8

Imagen de bienvenida al entrar al ide



2. Descarga e instalación de NodeJs junto sus módulos ExpressJs y Mongoose

- En el sitio oficial de descarga de node.js (<https://nodejs.org/es/>) se descarga e instala la versión estable.

Figura 9

Página de descarga de node js



- Al terminar la descarga, se procede a instalar nodejs siguiendo las instrucciones que se muestran en cada paso de la instalación.

Figura 10

Comienzo de la instalación de nodejs

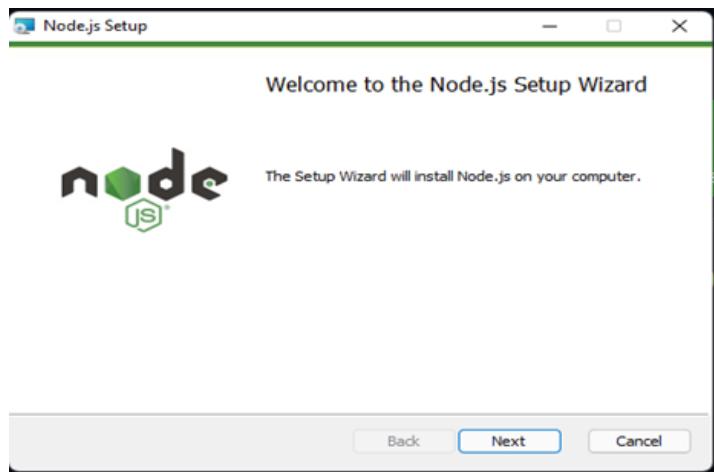


Figura 11

Comienzo de instalación.

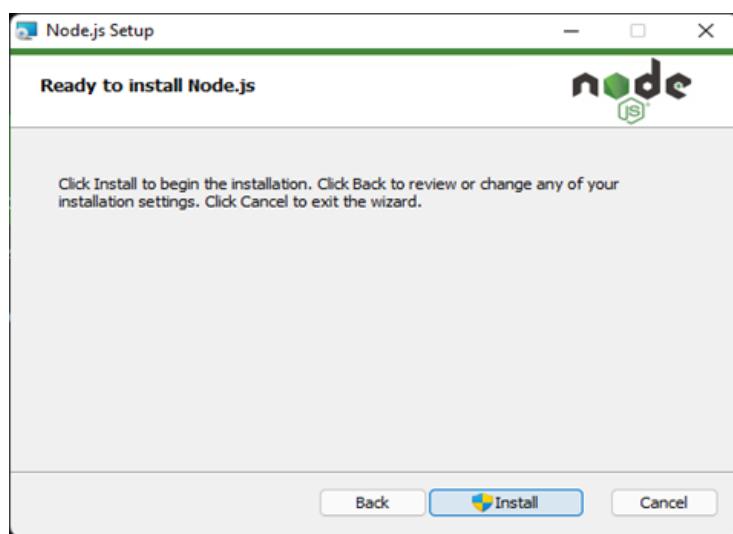


Figura 12

Selección de ubicación de instalación de nodejs

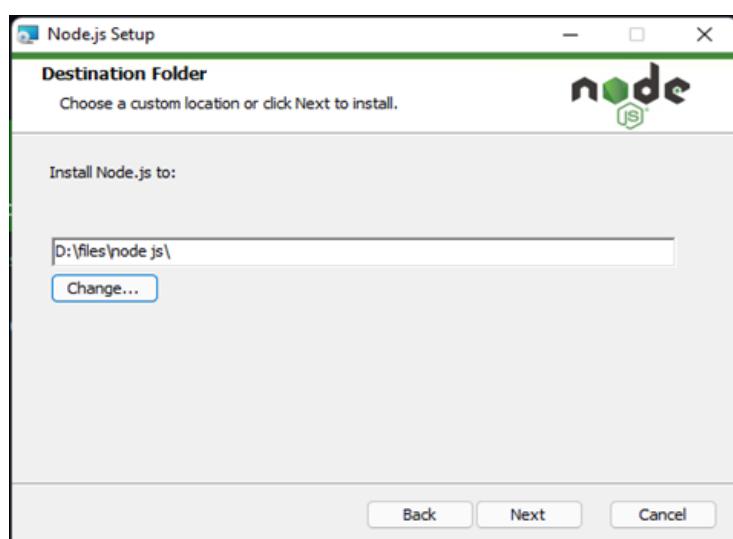
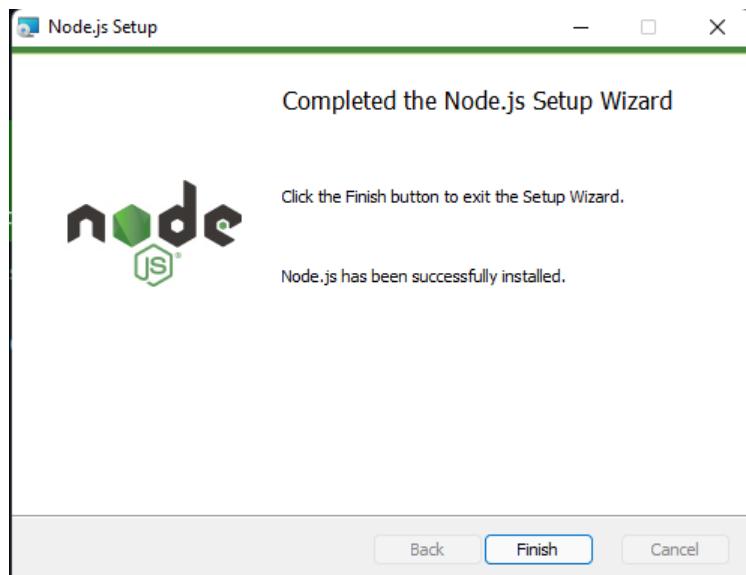


Figura 13

Finalización de la instalación de nodejs



Una vez instalado Node js, se procede a instalar sus módulos Express y Mongoose, para ello en la consola de comandos del sistema operativo se escribe el siguiente comando para instalar Express.

Figura 14

Instalación del primer módulo “express”

A screenshot of a Command Prompt window. The title bar says "Command Prompt". The command line shows "C:\Users\aburg>npm install express". The rest of the window is black, indicating the command is still running or has completed.

Figura 15

Finalización de la instalación de express

```
C:\Users\aburg>npm install express
npm WARN old lockfile
npm WARN old lockfile The package-lock.json file was created with an old version of npm,
npm WARN old lockfile so supplemental metadata must be fetched from the registry.
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, please be patient...
npm WARN old lockfile

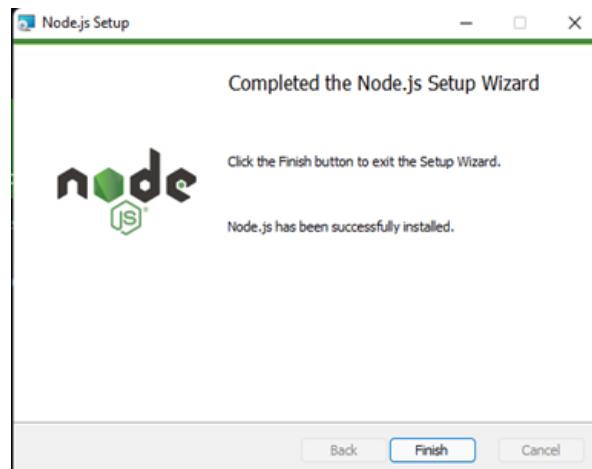
removed 149 packages, changed 8 packages, and audited 51 packages in 20s

2 packages are looking for funding
  run `npm fund` for details

Found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.3.1 -> 8.5.3
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.5.3
npm notice Run npm install -g npm@8.5.3 to update!
npm notice

C:\Users\aburg>
```

Cuando se haya instalado el módulo se mostrará el siguiente mensaje.



- c. Ahora de la misma manera, se escribe el siguiente comando para la instalación de mongoose

Figura 16

Instalación del módulo “mongoose”

```
C:\ Command Prompt
C:\Users\aburg>npm install mongoose
```

Cuando se haya instalado el módulo se mostrará el siguiente mensaje.

Figura 17

Finalización de la instalación de mongoose

```
C:\Users\aburg>npm install mongoose
added 28 packages, and audited 79 packages in 6s
6 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
C:\Users\aburg>_
```

3. Diagrama de Clase

Para la creación del servicio REST primero se debe tener en cuenta el diagrama de clase que se tiene planteado, teniendo en cuenta que será un servicio para un CRUD de clientes.

A continuación, se muestra el diagrama de la clase Cliente.

Figura 18

Diagrama de la clase “Cliente”

CLIENTE
- nombre : String - apellido : String - edad : int - subtotal : float

4. Estructura del código.

El servicio REST estará estructurado de la siguiente manera, 3 carpetas que son:

- *api*: En esta carpeta se tiene el archivo donde están las URLs de nuestras APIs.
- *controller*: En esta carpeta se tiene el archivo con los métodos que permitirán interactuar con la información de la base de datos.
- *model*: En esta carpeta se tiene el archivo donde estará la estructura del objeto u objetos, similar al esquema de los datos de la base de datos.

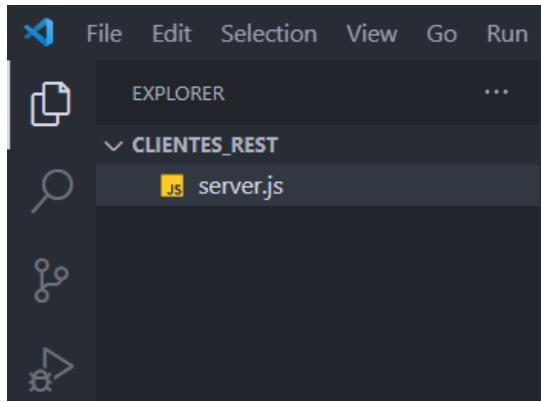
Y finalmente se tiene el archivo *server.js* donde se ejecutará todo el servicio.

5. Código

- a. Se empieza primero creando la carpeta donde se guardará el servicio REST.
- b. Se abre la carpeta con visual studio code, y se crea el archivo *server.js*

Figura 19

Creación del archivo del servidor



c. Ahora se debe llenar el archivo del servidor de la siguiente manera.

Figura 20

Código del archivo del servidor

```
server.js X

server.js > [o] server
1 var express = require('express') //Llamamos al modulo express para el Levantamiento del servidor
2 var app = express()
3 const portParameter = 8081 //Asignamos un puerto para que se levante el sevidor
4
5 var server = app.listen(portParameter, function(){ //funcion que se encarga del levantamiento del server
6   var host = server.address().address
7   var port = server.address().port
8   console.log(`Server is Running on port ${portParameter}...`)
9 })
```

The code editor window shows the 'server.js' file. The code defines an Express application, sets it to listen on port 8081, and logs a message to the console indicating it is running on that port.

Además, en una terminal se puede probar el servidor ejecutando el comando C:\>node server.js y se mostrará el siguiente mensaje que indicará que el servidor está funcionando de manera correcta.

Figura 21

Levantamiento del servidor

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Users\aburg\OneDrive\Escritorio\clientes_REST> node .\server.js
Server is Running on port 8081...
```

A terminal window titled 'TERMINAL' shows the command 'node .\server.js' being run. The output is 'Server is Running on port 8081...', confirming the server is up and listening on the correct port.

Figura 22

Creación de la base de datos en mongoDBAtlas

ADRIAN'S ORG - 2020-08-22 > PROJECTS

Create a Project

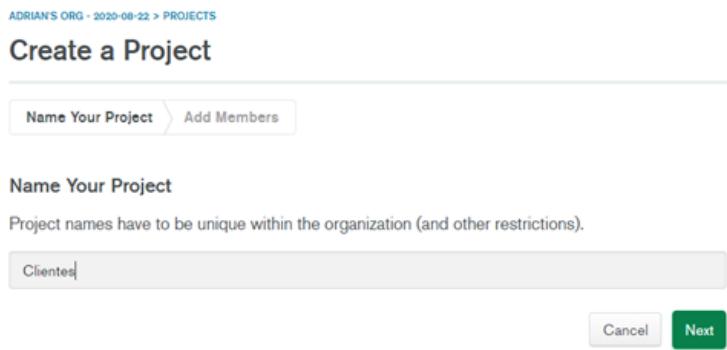
Name Your Project Add Members

Name Your Project

Project names have to be unique within the organization (and other restrictions).

Cientes

Cancel Next



- d. Ahora se tiene que crear la base de datos en MongoDB Atlas donde se guardarán los datos con los que interactúa nuestro servicio REST.

Figura 23

Creación de la base de datos

ADRIAN'S ORG - 2020-08-22 > CLIENTES

Database Deployments

Create a database

Choose your cloud provider, region, and specs.

Build a Database

Once your database is up and running, live migrate an existing MongoDB database into Atlas with our Live Migration Service.

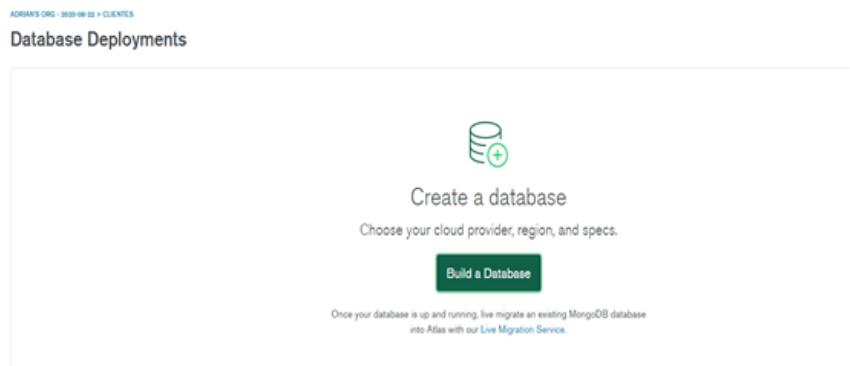


Figura 24

Finalización de la creación de la base de datos.

ADRIAN'S ORG - 2020-08-22 > CLIENTES

Database Deployments

Find a database deployment.

+ Create

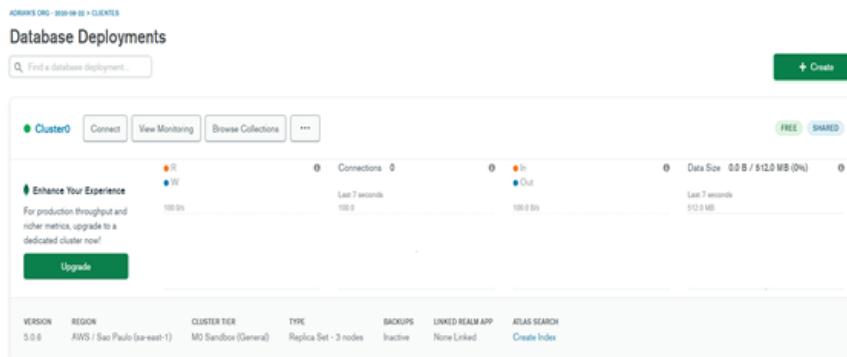
Cluster0 Connect View Monitoring Browse Collections ...

FREE SHARED

Enhance Your Experience
For production throughput and other metrics, upgrade to a dedicated cluster now!

Upgrade

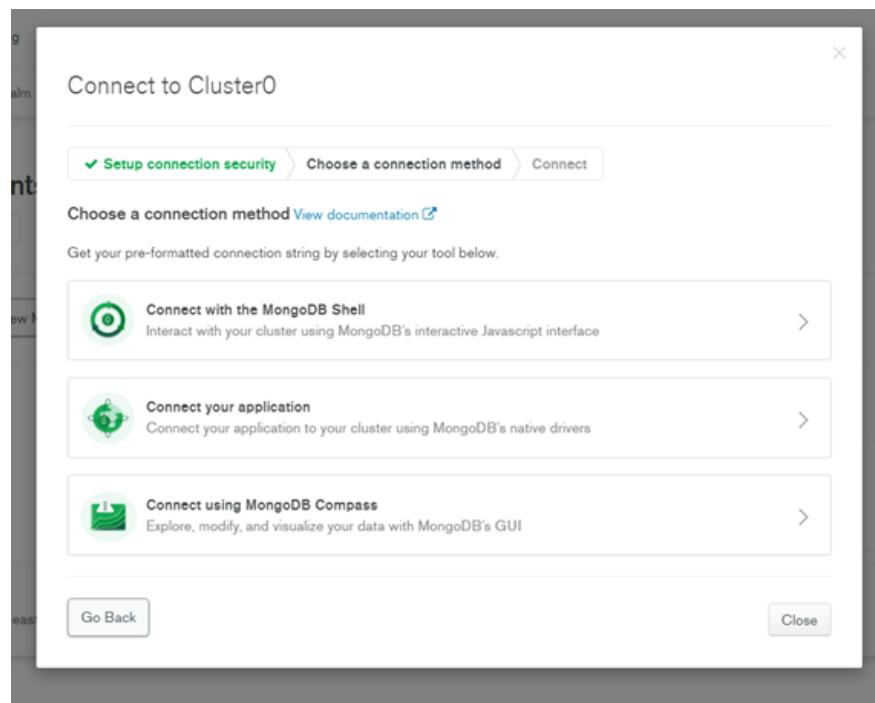
VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED REALM APP	ATLAS SEARCH
5.0.6	AWS / Sao Paulo (sa-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index



Ahora, para poder conectar con esta base de datos creada, se dirige a Connect

Figura 25

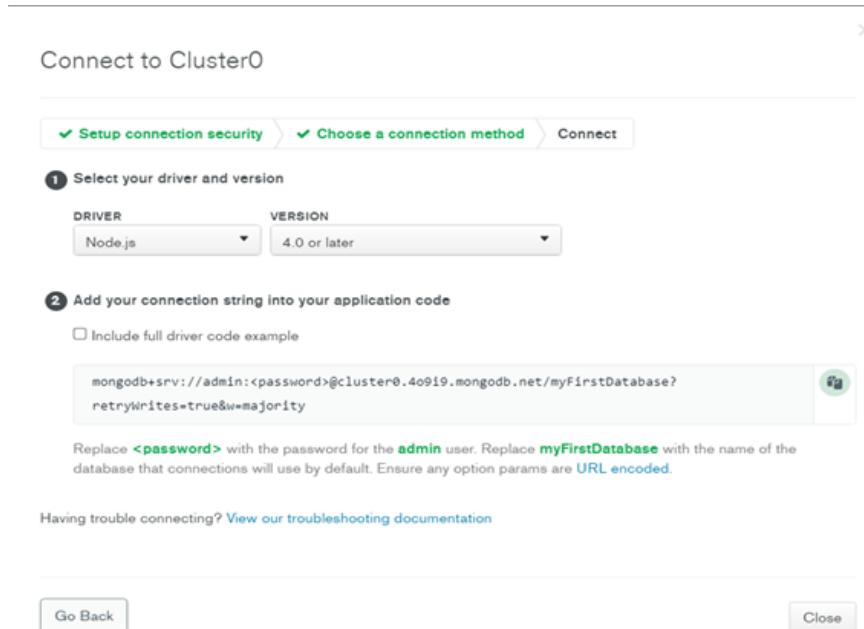
Comienzo de la conexión de la base de datos con la aplicación



Se selecciona la opción “Connect your application” y se copia el link de conexión.

Figura 26

Selección de los parámetros correspondientes a nuestro servicio REST (node.js)



- e. Ahora hay que dirigirse a server.js y se modifica el código de tal manera que se incluya la conexión con la base de datos creada.

Figura 27

Adición de la conexión a la base de datos en el código

```
server.js > ...
1 var express = require('express') //Llamamos al modulo express para el levantamiento del servidor
2 var mongoose = require('mongoose') //Incluimos el modulo mongoose para conectar con mongoDB Atlas
3 var app = express()
4 const portParameter = 8081 //Asignamos un puerto para que se levante el sevidor
5
6 mongoose.connect(`//Establecemos conexion
7 "mongodb+srv://admin:admin@cluster0.4o9i9.mongodb.net/CLIENTES?retryWrites=true&w=majority", /
8 {useNewUrlParser: true},
9 (err,res) => {
10   err && console.log("Error conectado a la base de datos")
11   app.listen(portParameter, () => {
12     console.log(`Server is running at http://localhost:${portParameter}`)
13   })
14 }
15 `)
```

Se prueba el servidor con el comando C:\>node server.js y si todo está bien, se mostrará el mensaje indicando el puerto de conexión, de la siguiente manera:

Figura 28

Inicialización del servidor.

```
PS C:\Users\aburg\OneDrive\Escritorio\clientes_REST> node .\server.js
Server is running at http://localhost:8081
```

f. Es momento de crear el modelo, para eso se crea la carpeta model y el archivo Cliente.js

Figura 29

Creación del modelo “Cliente”



Ahora debemos crear el modelo en base al diagrama de clase creado, teniendo en cuenta los atributos de la clase junto su tipo de dato, de la siguiente manera:

Figura 30

Código del archivo del modelo “Cliente”

```
Client.js  X
model > Client.js > <unknown>
1  const mongoose = ...
2  const Schema = mongoose.Schema
3
4  const ClientSchema = new Schema({
5    name: {type: String},
6    lastName: {type: String},
7    age: {type: Number},
8    subTotal: {type: Number}
9  })
10
11 module.exports = Client = mongoose.model('Client', ClientSchema)
```

g. Ahora se crea la carpeta controller y el archivo Clients.js, obteniendo el siguiente árbol de archivos.

Figura 31

Creación del controlador “Clientes”



Dentro del archivo Clients.js se tiene que importar el modelo creado y se crea las funciones CRUD, se empieza creando el método POST/Create para insertar el primer registro en la base de datos y el método GET/Read para obtener los registros de la base de datos, el código quedaría de la siguiente manera:

Figura 32

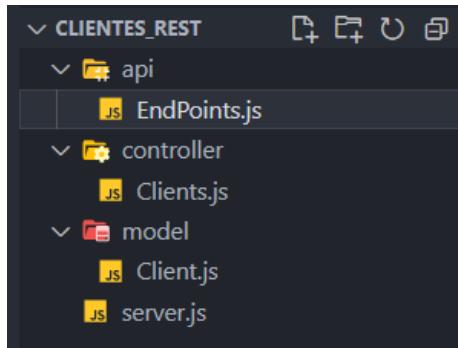
Código del controlador “Cliente”

```
js Clients.js  X
clientes_REST > controller > js Clients.js > ...
1  const Client = require('../model/Client')
2
3  const createClient = (req, res) => {
4      let client = new Client({
5          name: req.body.name,
6          lastName: req.body.lastName,
7          age: req.body.age,
8          subTotal: req.body.subTotal
9      })
10
11  client.save((err, cli) => {
12      err && res.status(500).json(err.message)
13      res.status(200).json(cli)
14  })
15
16
17  const getClients = (req, res) => {
18      Client.find((err, clients) => {
19          err && res.status(500).send(err.message)
20          res.status(200).json(clients)
21      })
22}
```

- Finalmente completando la estructura establecida, se crea la carpeta api junto al archivo EndPoints.js donde se asignan las URLs para consumir el servicio REST.

Figura 33

Creación del archivo de las uris



Ahora deberemos importar el controlador creado, y mediante un router, crear las URIs para cada método del servicio, el código estaría de la siguiente manera:

Figura 34

Código del archivo de las uris “EndPoints”

```
js EndPoints.js X
api > EndPoints.js > ...
1 const ClientController = require('../controller/Clients') //Importamos el controlador creado
2 const express = require('express') //Importamos el modulo express para la creacion de las uris
3 const router = express.Router() //Creamos el router para enrutar las uris con los metodos creados
4
5 /* ----- CREACION DE LAS URIS ----- */
6 //router.metodo_CRUD('uri', controlador.metodo)
7 router.post("/add", ClientController.createClient)
8 router.get("/all", ClientController.getclients)
9
10 module.exports = router //Exportamos el router con las uris creadas
11
```

Ahora lo que se debe hacer es agregar estas URIs en el archivo del servidor, teniendo finalmente:

Figura 35

Adición del archivo de las uris en el archivo del servidor

```

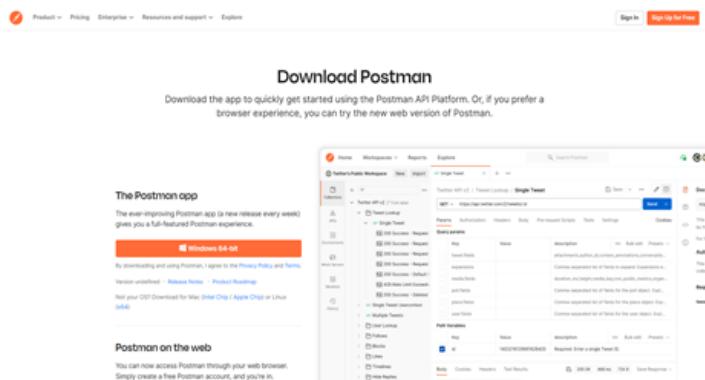
js server.js X
clientes_REST > server.js ...
1 const express = require('express')
2 const mongoose = require('mongoose')
3 const bodyParser = require('body-parser')
4 const portParameter = 8081
5 const EndPoints = require('../api/EndPoints')
6
7 var app = express()
8
9 app.use(bodyParser.urlencoded({extended: false}))
10 app.use(bodyParser.json())
11 //http://localhost:8081/clientes/uri
12 app.use("/clientes",EndPoints)
13
14 mongoose.connect(
15   "mongodb+srv://admin:admin@cluster0.4o9i9.mongodb.net/CLIENTES?retryWrites=true&w=majority",
16   {useNewUrlParser: true},
17   (err, res) => {
18     err && console.log("Error al conectarse a la base de datos")
19     app.listen(portParameter,() => {
20       console.log(`Server is running on port ${portParameter}`)
21     })
22   }
23 )

```

- Finalmente, se levanta el servidor con `C:\>node server.js` y se prueba el API mediante el programa postman (link de descarga: https://www.postman.com/downloads/?utm_source=postman-home).

Figura 36

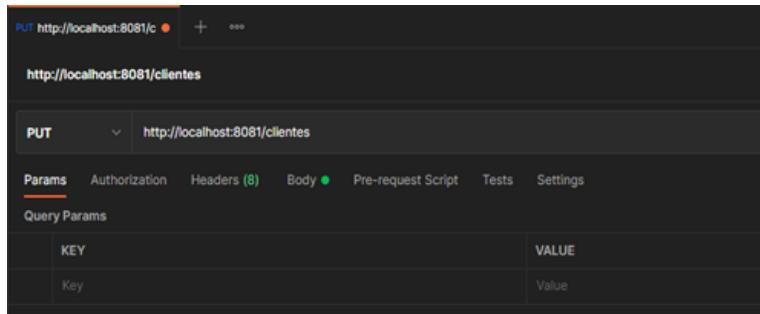
Página de descarga de Post Man



Una vez en la herramienta, se ingresa el URI de nuestro servicio <http://localhost:8081/clientes> de la siguiente manera.

Figura 37

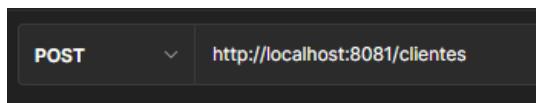
Prueba de la función para actualizar un cliente



En las opciones de la derecha se tiene los métodos http (GET, POST, PUT, DELETE) donde se selecciona POST para insertar nuestro primer registro

Figura 38

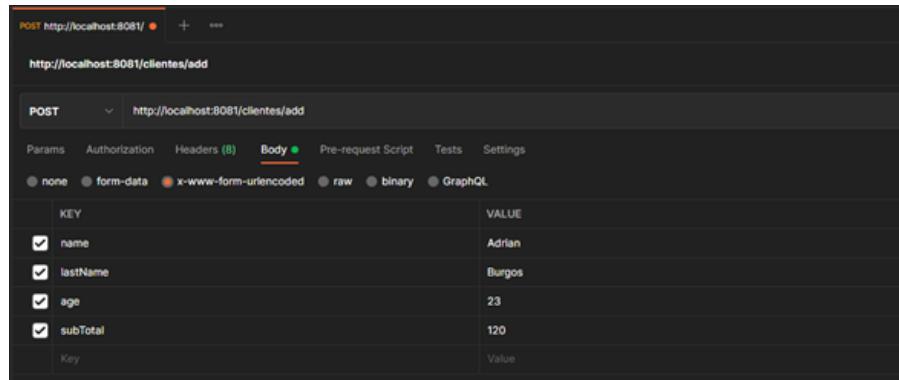
URI para actualizar cliente



Para poder probar la funcionalidad “insertar un registro”, se debe ir a la sección Body y en x-www-form-urlencoded y se llena los campos de la siguiente manera atributo : valor (key:value)

Figura 39

Prueba de la función para actualizar un cliente



Se hace clic en el botón Send

Figura 40

Ejecución de la prueba del URI

The screenshot shows a Postman interface with a POST request to `http://localhost:8081/clientes/add`. The 'Body' tab is selected, showing form-data parameters:

KEY	VALUE
<input checked="" type="checkbox"/> name	Adrian
<input checked="" type="checkbox"/> lastName	Burgos
<input checked="" type="checkbox"/> age	23
<input checked="" type="checkbox"/> subTotal	120

The 'Body' tab also displays the raw JSON response:

```

1   {
2     "name": "Adrian",
3     "lastName": "Burgos",
4     "subTotal": 120,
5     "_id": "622949c6130bed256dcf0925",
6     "__v": 0
7   }

```

Figura 41

Comprobación de los datos en mongoDB Atlas

The screenshot shows a MongoDB Atlas interface with the title 'QUERY RESULTS 1-2 OF 2'. It displays two documents:

```

_id: ObjectId("622949c6130bed256dcf0925")
name: "Adrian"
lastName: "Burgos"
subTotal: 120
__v: 0

_id: ObjectId("62294a6b130bed256dcf0927")
name: "Nombre_1"
lastName: "Apellido_1"
subTotal: 200
__v: 0

```

Y como se puede observar, se ha registrado con éxito, ahora se puede probar el URI para obtener los registros ingresados en la base de datos.

Figura 42

URI para visualizar los clientes registrados en la base de datos

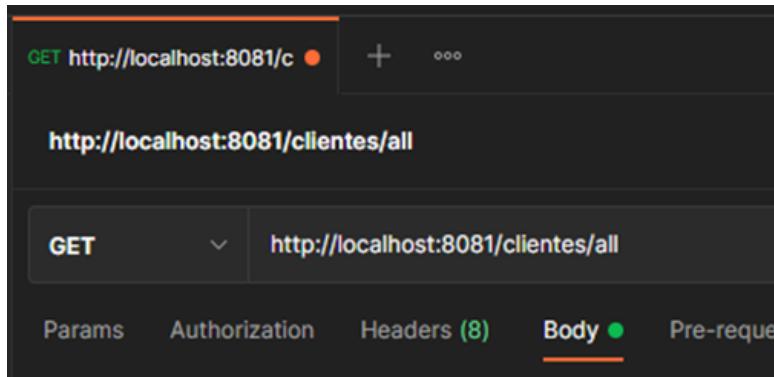


Figura 43

Resultados obtenidos al ejecutar la uri

```
1 [  
2   {  
3     "_id": "622949c6130bed256dcf0925",  
4     "name": "Adrian",  
5     "lastName": "Burgos",  
6     "subTotal": 120,  
7     "__v": 0  
8   },  
9   {  
10    "_id": "62294a6b130bed256dcf0927",  
11    "name": "Nombre_1",  
12    "lastName": "Apellido_1",  
13    "subTotal": 200,  
14    "__v": 0  
15  }  
16 ]
```

A screenshot of a Postman 'Body' tab showing a JSON array of two objects. Each object represents a client with fields: _id, name, lastName, subTotal, and __v.

Y como se puede observar, los datos registrados han sido mostrados por pantalla.

Enlaces:

- Aquí les comparto el enlace (link/URL) del repositorio del código del servicio REST para que lo puedan probar en sus computadoras. Cabe mencionar, que se debe seguir los pasos descritos en este taller para poder probar el código. En este caso un simple copy and paste pueda que no funcione.

<https://github.com/elascano/ESPE-DESARROLLO-WEB-2022>

El estudiante Adrián Burgos adicionalmente, tiene su propio repositorio, de donde pueden bajar el código.

https://github.com/AdrianBurgosA/Clientes_REST

- Este es el enlace de la base de datos en mongoDB Atlas de Adrián Burgos para que lo sigas paso a paso:

<mongodb://admin:admin@cluster0.4o9i9.mongodb.net/CLIENTES?retryWrites=true&w=majority>

- A continuación, en enlace del hands-on lab, en YouTube donde se crea servicios REST CRUD completos desde 0 usando node.js:

<https://youtu.be/CQN1ZQ5jVyE>



Recursos complementarios

Build A Simple JAX RS RESTful web service using NetBeans IDE

Build A Simple JAX RS RESTful web service using ...



<https://www.youtube.com/watch?v=enZoyyfbBg8>

Build A Simple JAX RS RESTful web service using NetBeans IDE



Bibliografía



Architectural Styles and the Design of Network-based Software Architectures. (s. f.). Recuperado 28 de marzo de 2022, de <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

Common MIME types—HTTP | MDN. (s. f.). Recuperado 28 de marzo de 2022, de https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types

Comparison of server-side web frameworks. (2022). En Wikipedia. https://en.wikipedia.org/w/index.php?title=Comparison_of_server-side_web_frameworks&oldid=1079789622

Express—Node.js web application framework. (s. f.). Recuperado 28 de marzo de 2022, de <https://expressjs.com/>

HTTP - Status Codes. (s. f.). Recuperado 1 de abril de 2022, de https://www.tutorialspoint.com/http/http_status_codes.htm

Marrs, T. (2017). JSON at Work: Practical Data Integration for the Web (1st edition). O'Reilly Media.

Node.js. (2022). En Wikipedia. <https://en.wikipedia.org/w/index.php?title=Node.js&oldid=1078841990>

Top 5 Languages to Server-Side Scripting in 2022. (2022, enero 8). Qulix Systems. <https://www.qulix.com/about/blog/the-best-server-side-language/>

W3Techs—Extensive and reliable web technology surveys. (s. f.). Recuperado 24 de marzo de 2022, de <https://w3techs.com/>

Which are the most popular APIs in the world? (s. f.). Call Me Fred. Recuperado 28 de marzo de 2022, de <https://www.callmefred.com/which-are-the-most-popular-apis-in-the-world/>

Autoevaluación

1. Provee un conjunto de herramientas y bibliotecas, que simplifican el proceso de programación

- Frameworks.
- Bibliotecas.
- Lenguajes de Programación.

Pregunta 1 de 10

[Atras](#)[Siguiente](#)[Enviar todo](#)

Autoevaluación

2. ¿Node.js es?

- Lenguaje de Programación.
- Open source Back-end environment.
- Biblioteca.

Pregunta 2 de 10

[Atras](#)[Siguiente](#)[Enviar todo](#)

Autoevaluación

3. ¿Laravel es?

- Framework PHP para desarrollo de aplicaciones Web.
- Framework PHP para desarrollo de servicios Web.
- Framework PHP para desarrollo de páginas Web.

Pregunta 3 de 10

[Atras](#)[Siguiente](#)[Enviar todo](#)

Autoevaluación

4. El concepto API First design se basa en:

- Las APIs deben adaptarse al estado de las Tecnologías de la Información
- El diseño de las APIs depende del cliente web que las consuma
- Se diseña las APIs antes de programar sus consumidores.

Pregunta 4 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

5. Lo importante del diseño y programación por capas es que

- Los programas (servicios y clientes) deben estar obligatoriamente en diferentes computadoras
- Los programas (servicios y clientes) deben conocer la estructura de los otros programas
- Los programas (servicios y clientes) no sepan nada de los otros programas, únicamente los acceden a través de sus interfaces de programación

Pregunta 5 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

6. En el Web actualmente se ofrecen muchas APIs para que los desarrolladores de aplicaciones Web las puedan reusar, la mayoría de las APIs son de tipo

- REST
- SOAP
- RPC

Pregunta 6 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

7. Un recurso web que es retornado por una API, para ser interpretado en el lado del cliente debe ser retornado al navegador como

- JSON
- MIME type
- XML

Pregunta 7 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

8. El método http que se debe usar para insertar datos en una aplicación Web es

- POST
- PUT
- GET

Pregunta 8 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

9. Un servicio web que cumple las seis restricciones del cliente y del código de acuerdo a Roy Fielding, se consideran

- SAOP Web Services
- REST Web Services
- RESTful Web Services

Pregunta 9 de 10

[Atras](#)[Siguiente](#)[Enviar todo](#)



Desarrollo de Aplicaciones Web

Tema 3 Desarrollo de Front-End Web



Carrera en Línea
Tecnologías de la Información



3.1 Arquitectura (patrón) Modelo-Vista-Controlador (MVC)



El desarrollo de aplicaciones software debe seguir ciertos estándares para que su código sea de calidad y sobre todo legible para que los programadores le puedan dar mantenimiento al código de una manera rápida y efectiva. Uno de los paradigmas de programación que más se aproxima a ofrecer una buena solución de modularidad que permite un mantenimiento por componentes es la Programación Orientada a Objetos.

La Programación Orientada a Objetos permite diseñar programas modulares y dividir las responsabilidades en diferentes unidades de programación llamadas Clases, la agrupación de clases depende del lenguaje de programación, por ejemplo: en Java se usa paquetes, en C# se usa namespaces, PHP usa namespaces, Python utiliza namespace packages, y así cada uno de los diferentes lenguajes de programación manejan de una u otra manera la organización de sus clases. Un estándar que la mayoría de herramientas de desarrollo modernas siguen es el patrón MVC (Model-View-Controller).

3.1.1 Introducción

El patrón Modelo/Vista/Controlador (Model-View-Controller) es una estrategia para dividir las responsabilidades de un componente GUI (Graphical User Interface) (Eck, 2016). Propuesto en 1988 por (Krasner & Pope, 1988), MVC establece que existen tres tipos de objetos: 1) *Model* representa el estado actual de los datos del componente, 2) *View* es la presentación visual del componente en la pantalla, 3) *Controller* es el responsable de reaccionar a los eventos que ocurren en la vista para cambiar el modelo o responde en respuesta a eventos generados por otras fuentes e.g., sensores. En resumen, cada una de estas responsabilidades es manejada por tres objetos diferentes. Cada objeto tiene un rol definido, una responsabilidad única.

El modelo MVC desacopla las vistas y los modelos estableciendo un protocolo subscribe/notify entre ellos, la vista debe asegurarse que si los datos cambian, el usuario vea esos datos, el modelo notifica a la vista de los cambios, de esta manera, se puede crear nuevas vistas sin modificar los modelos (Gamma et al., 1994). Por ejemplo, una aplicación de escritorio, una página Web, una app móvil.

Como regla para el código, y siguiendo los estándares de Java, los nombres de las variables y de los métodos empiezan con minúscula y si se componen de más de dos palabras las siguientes palabras deben empezar con mayúscula, en Java no se usa el _ (underscore), esto se conoce como lowerCamelCase. Para dar nombres a las clases se empieza con Mayúscula y lo mismo con las palabras que le sigan a la primera, esto se conoce como UpperCamelCase. Los nombres de los paquetes se escriben con minúsculas todo el nombre. Para este curso se debe usar inglés para nombrar a los identificadores en el código: variables, methods, parameters, Classes, interfaces, packages, libraries. Por ejemplo ver Tabla 1:

Tabla 1

Uso de Identificadores en Java

El patrón Modelo/Vista/Controlador (Model-View-Controller) es una estrategia para dividir las responsabilidades de un componente GUI (Graphical User Interface) (Eck, 2016). Propuesto en 1988 por (Krasner & Pope, 1988), MVC establece que existen tres tipos de objetos: 1) *Model* representa el estado actual de los datos del componente, 2) *View* es la presentación visual del componente en la pantalla, 3) *Controller* es el responsable de reaccionar a los eventos que ocurren en la vista para cambiar el modelo o responde en respuesta a eventos generados por otras fuentes e.g., sensores. En resumen, cada una de estas responsabilidades es manejada por tres objetos diferentes. Cada objeto tiene un rol definido, una responsabilidad única.

El modelo MVC desacopla las vistas y los modelos estableciendo un protocolo subscribe/notify entre ellos, la vista debe asegurarse que si los datos cambian, el usuario vea esos datos, el modelo notifica a la vista de los cambios, de esta manera, se puede crear nuevas vistas sin modificar los modelos (Gamma et al., 1994). Por ejemplo, una aplicación de escritorio, una página Web, una app móvil.

Como regla para el código, y siguiendo los estándares de Java, los nombres de las variables y de los métodos empiezan con minúscula y si se componen de más de dos palabras las siguientes palabras deben empezar con mayúscula, en Java no se usa el _ (underscore), esto se conoce como lowerCamelCase. Para dar nombres a las clases se empieza con Mayúscula y lo mismo con las palabras que le sigan a la primera, esto se conoce como UpperCamelCase. Los nombres de los paquetes se escriben con minúsculas todo el nombre. Para este curso se debe usar inglés para nombrar a los identificadores en el código: variables, methods, parameters, Classes, interfaces, packages, libraries. Por ejemplo ver Tabla 1:

Tabla 1

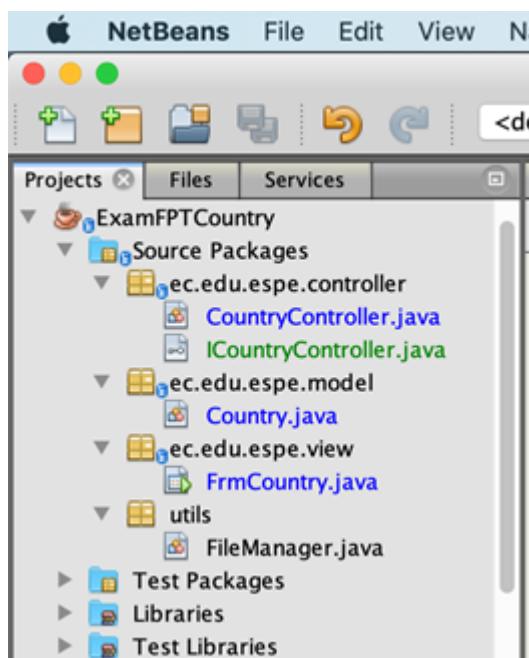
Uso de Identificadores en Java

Identificador	Tipo
<code>int numberOfWheels;</code>	variable
<code>public void getNumberOfWheels();</code>	method
<code>public class Vehicle{ ... }</code>	Class
<code>public interface IVehicleController{ ... }</code>	Interface
<code>float computeTotalPrice(float value, float ivaPercentage);</code>	parameters
<code>ec.edu.espe.dealer.model</code>	packages

En caso de una aplicación de escritorio, la organización del código (Java) se puede hacer en tres paquetes *model*, *view* y *controller* (Figura 1). Adicionalmente se tiene un paquete *utils* que es donde se programa el acceso a los datos del sistema. La persistencia de los datos puede ser manejada en archivos planos (txt, csv, xml, json) o en bases de datos estructuradas como MySQL, SQL Server, Oracle, o en bases de datos no estructuradas (No SQL) como MongoDB.

Figura 1

Paquetes de una Aplicación de Escritorio Siguiendo el Patrón MVC



El ejemplo aquí presentado fue desarrollado usando el IDE NetBeans. La vista (view) es una aplicación gráfica de escritorio (Figura 2), el modelo está representado por una clase POJO -Plain Old Java Object (Figura 3), el controlador es un elemento de programación que maneja la lógica de negocios y el acceso a base de datos luego de responder a un evento de la IU (Figura 4), que por lo general lo puede hacer a través de una clase utilitaria que se encuentra en el paquete utils (Figura 5) que maneja el acceso a un archivo (operaciones CRUD). En el presente ejemplo, se usa un archivo de texto con formato csv para manejar la persistencia de datos. Por lo general la persistencia está en una base de datos.

Figura 2

View -> Java Swing Form: FrmCountry



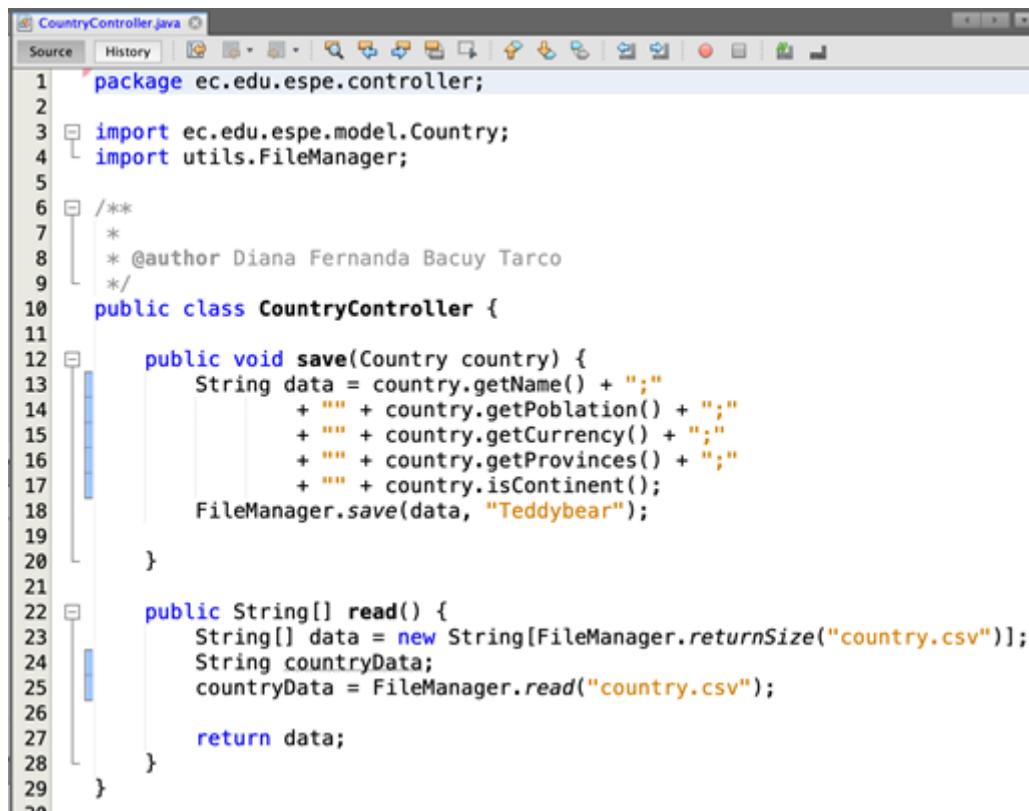
Figura 3

Model -> Clase POJO Country

```
Country.java
Source History ...
1 package ec.edu.espe.model;
2 /**
3  * 
4  * @author Diana Fernanda Bacuy Tarco
5 */
6 public class Country {
7     private String name;
8     private int poblation;
9     private String currency;
10    private String provinces;
11    private boolean continent;
12
13    public Country(String name, int poblation, String currency, String prov
14        this.name = name;
15        this.poblation = poblation;
16        this.currency = currency;
17        this.provinces = provinces;
18        this.continent = continent;
19    }
20
21    /**
22     * @return the name
23     */
24    public String getName() {
25        return name;
26    }
27
28    /**
29     * @param name the name to set
30     */
31    public void setName(String name) {
```

Figura 4

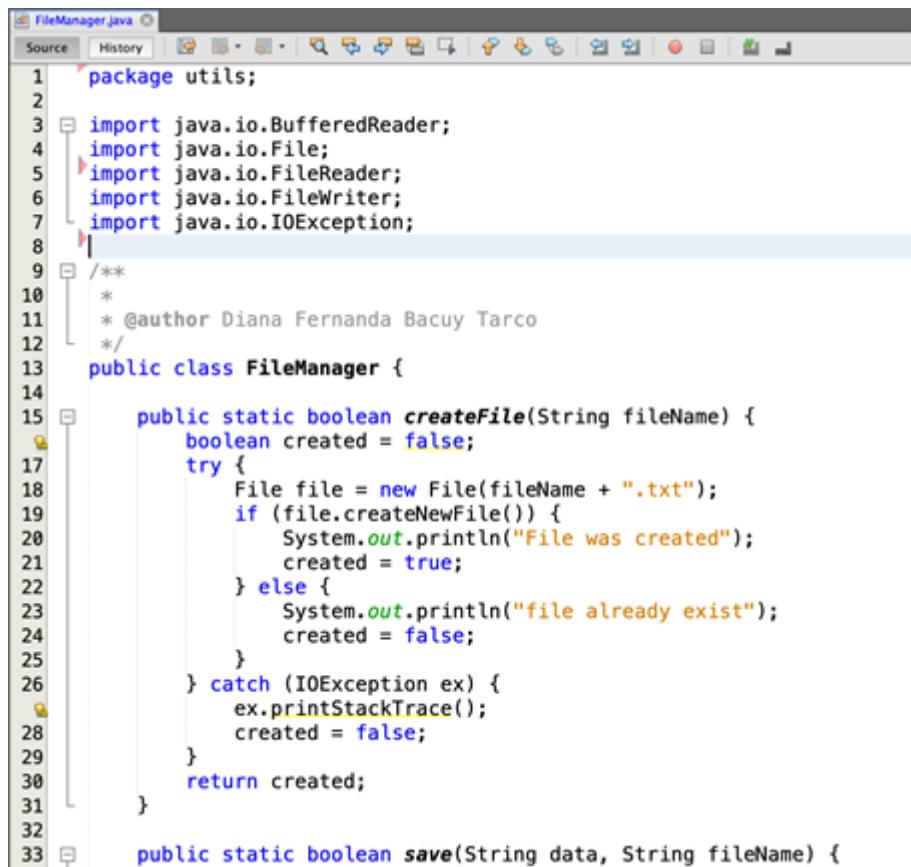
Controller -> Clase CountryController



```
1 package ec.edu.espe.controller;
2
3 import ec.edu.espe.model.Country;
4 import utils.FileManager;
5
6 /**
7  * 
8  * @author Diana Fernanda Bacuy Tarco
9  */
10 public class CountryController {
11
12     public void save(Country country) {
13         String data = country.getName() + ";"
14             + "" + country.getPopulation() + ";"
15             + "" + country.getCurrency() + ";"
16             + "" + country.getProvinces() + ";"
17             + "" + country.isContinent();
18         FileManager.save(data, "Teddybear");
19     }
20
21     public String[] read() {
22         String[] data = new String[FileManager.returnSize("country.csv")];
23         String countryData;
24         countryData = FileManager.read("country.csv");
25
26         return data;
27     }
28 }
29
30
```

Figura 5

Utils -> Clase FileManager que maneja la persistencia



```
1 package utils;
2
3 import java.io.BufferedReader;
4 import java.io.File;
5 import java.io.FileReader;
6 import java.io.FileWriter;
7 import java.io.IOException;
8
9 /**
10  * 
11  * @author Diana Fernanda Bacuy Tarco
12  */
13 public class FileManager {
14
15     public static boolean createFile(String fileName) {
16         boolean created = false;
17         try {
18             File file = new File(fileName + ".txt");
19             if (file.createNewFile()) {
20                 System.out.println("File was created");
21                 created = true;
22             } else {
23                 System.out.println("file already exist");
24                 created = false;
25             }
26         } catch (IOException ex) {
27             ex.printStackTrace();
28             created = false;
29         }
30         return created;
31     }
32
33     public static boolean save(String data, String fileName) {
34
35     }
```

3.1.2 Diseño

MVC es considerado un patrón arquitectónico, y un Framework (marco de trabajo), como tal, muchas herramientas actuales de desarrollo lo han implementado, o al menos alguna variación de este patrón: MVVM (Windows Presentation Foundation, AngularJS, ReactJS, Vue.js), MVP -Model View Presenter (JavaFX, ASP.NET, Swing, PHP), MVVC. La comunicación se puede dar entre los componentes del sistema, como se observa en la Figura 6. La pantalla solicita al controlador un servicio, en este caso save, el controlador utiliza el modelo y el utilitario de acceso a los datos para actualizar el repositorio de datos, de ser necesario, el controlador emitiría una orden de actualización de datos en la pantalla, por ejemplo, podría refrescar una tabla de países de ser necesario. En la mayoría de casos, el controlador retorna una respuesta de satisfactorio. En este ejemplo aquí presentado, no se actualiza la pantalla si existe un cambio en la base de datos.



view

FrmCountry

COUNTRY

Name: Ecuador

Population: 17640000

Currency: dollar

Provinces: 24

Continent: America

SAVE

← FrmCountry.java

controller

```
CountryController.java
```

```
1 package ec.edu.espe.controller;
2
3 import ec.edu.espe.model.Country;
4 import utils.FileManager;
5
6 /**
7 * @author Diana Fernanda Bacuy Tarco
8 */
9
10 public class CountryController {
11
12     public void save(Country country) {
13         String data = country.getName() + ";" +
14             + country.getPopulation() + ";" +
15             + country.getCurrency() + ";" +
16             + country.getProvinces() + ";" +
17             + country.isContinent();
18         FileManager.save(data, "Teddybear");
19     }
20
21     public String[] read() {
22         String[] data = new String[FileManager.returnSize("country.csv")];
23         String countryData;
24         countryData = FileManager.read("country.csv");
25
26         return data;
27     }
28
29 }
```

← CountryController.java

```
FileManager.java
```

```
1 package utils;
2
3 import java.io.BufferedReader;
4 import java.io.File;
5 import java.io.FileReader;
6 import java.io.FileWriter;
7 import java.io.IOException;
8
9 /**
10 * @author Diana Fernanda Bacuy Tarco
11 */
12
13 public class FileManager {
14
15     public static boolean createFile(String fileName) {
16         boolean created = false;
17         try {
18             File file = new File(fileName + ".txt");
19             if (!file.createNewFile()) {
20                 System.out.println("File was created");
21                 created = true;
22             } else {
23                 System.out.println("file already exist");
24                 created = false;
25             }
26         } catch (IOException ex) {
27             ex.printStackTrace();
28             created = false;
29         }
30         return created;
31     }
32
33     public static boolean save(String data, String fileName) {
34
35 }
```

← FileManager.java

model

```
Country.java
```

```
1 package ec.edu.espe.model;
2
3 /**
4 * @author Diana Fernanda Bacuy Tarco
5 */
6
7 public class Country {
8     private String name;
9     private int population;
10    private String currency;
11    private String provinces;
12    private boolean continent;
13
14    public Country(String name, int population, String currency, String prov
15        this.name = name;
16        this.population = population;
17        this.currency = currency;
18        this.provinces = provinces;
19        this.continent = continent;
20
21    /**
22     * @return the name
23     */
24    public String getName() {
25        return name;
26    }
27
28    /**
29     * @param name the name to set
30     */
31    public void setName(String name) {
32
33 }
```

← Country.java

persistencia

countries.csv

1	Ecuador;17640000;dollar;24;America;
2	Peru;32970000;nuevo Sol;196;America;
3	Belgium;11560000;Euro;10;Europe

↑
Countries.csv

3.1.3 Orientación a Objetos

La orientación a objetos ayuda a organizar el código de un proyecto web. La interfaz de usuarios por si misma se podría considerar como un conjunto de objetos en su interior. El lenguaje JavaScript, usando para dar dinamismo a los clientes Web es un lenguaje orientado a objetos, pero este concepto fue subutilizado durante mucho tiempo. En la actualidad es utilizado por las nuevas bibliotecas y frameworks de Front-End como React, Angular, vue.js. a continuación un ejemplo de objeto en JavaScript:

```
let Country = {  
    name: "Ecuador"  
    population: 17640000,  
    currency: "dollar"  
    provinces: 24  
    continent: America  
}
```

En la sección 3.3.4, se trata en detalle el patrón MVVM (View-ViewModel-Model) usado por React y una comparativa con MVC.

En la actualidad, se debe considerar que la mayoría de lenguajes de programación son multiparadigma, esto quiere decir que soportan más de dos paradigmas de programación: a) Orientado a Objetos, b) Funcional, c) Estructurado, d) Lógico, (entre otros). Por ejemplo, Python es orientado a objetos, procedural, funcional, estructurado; Java es genérico, orientado a objetos, funcional; JavaScript es funcional, procedural, orientado a objetos; PHP es funcional, orientado a objetos, procedural.

Como se observa, hay un factor común en todos los lenguajes, ese factor común es la orientación a objetos. Sin embargo, muchos programadores siguen el paradigma procedural que es uno de los paradigmas más fáciles de usar y más eficiente en varias

La orientación a objetos ayuda a organizar el código de un proyecto web. La interfaz de usuarios por si misma se podría considerar como un conjunto de objetos en su interior. El lenguaje JavaScript, usando para dar dinamismo a los clientes Web es un lenguaje orientado a objetos, pero este concepto fue subutilizado durante mucho tiempo. En la actualidad es utilizado por las nuevas bibliotecas y frameworks de Front-End como React, Angular, vue.js. a continuación un ejemplo de objeto en JavaScript:

```
let Country = {  
    name: "Ecuador"  
    population: 17640000,  
    currency: "dollar"  
    provinces: 24  
    continent: America  
}
```

En la sección 3.3.4, se trata en detalle el patrón MVVM (View-ViewModel-Model) usado por React y una comparativa con MVC.

En la actualidad, se debe considerar que la mayoría de lenguajes de programación son multiparadigma, esto quiere decir que soportan más de dos paradigmas de programación: a) Orientado a Objetos, b) Funcional, c) Estructurado, d) Lógico, (entre otros). Por ejemplo, Python es orientado a objetos, procedural, funcional, estructurado; Java es genérico, orientado a objetos, funcional; JavaScript es funcional, procedural, orientado a objetos; PHP es funcional, orientado a objetos, procedural.

Como se observa, hay un factor común en todos los lenguajes, ese factor común es la orientación a objetos. Sin embargo, muchos programadores siguen el paradigma procedural, que es uno de los paradigmas más fáciles de usar y más eficiente en varias circunstancias (Brborich et al., 2020). Sin considerar el paradigma de programación que el desarrollador siga, el diseño de aplicaciones modernas de una u otra forma sigue el patrón MVC, lo que varía es el componente que encierra la lógica, o los datos, o la interfaz de usuario. Por ejemplo, si usáramos el lenguaje de programación C y siguiéramos este paradigma, el modelo podría ser representado por una estructura (*struct*), el controlador podría ser un archivo de cabecera (*.h) y la vista podría ser un programa de línea de comandos o una interfaz de usuario Gráfica (*GUI*). En la Figura 6, se observa que estos componentes se encuentran representados en clases. La Figura 8 representa la percepción de los componentes del patrón MVC desde el punto de vista de los patrones de diseño de (Freeman et al., 2004), en donde, el Controlador implementa el patrón Strategy, la Vista implementa el patrón Composite, y el Modelo implementa el patrón Observer. La Figura 9 representa el diseño UML (Diagrama de clases) de la aplicación de escritorio de la Figura 7, que sigue el patrón MVC.

Figura 8

Patrón MVC desde el punto de vista de los patrones Strategy, Composite y Observer

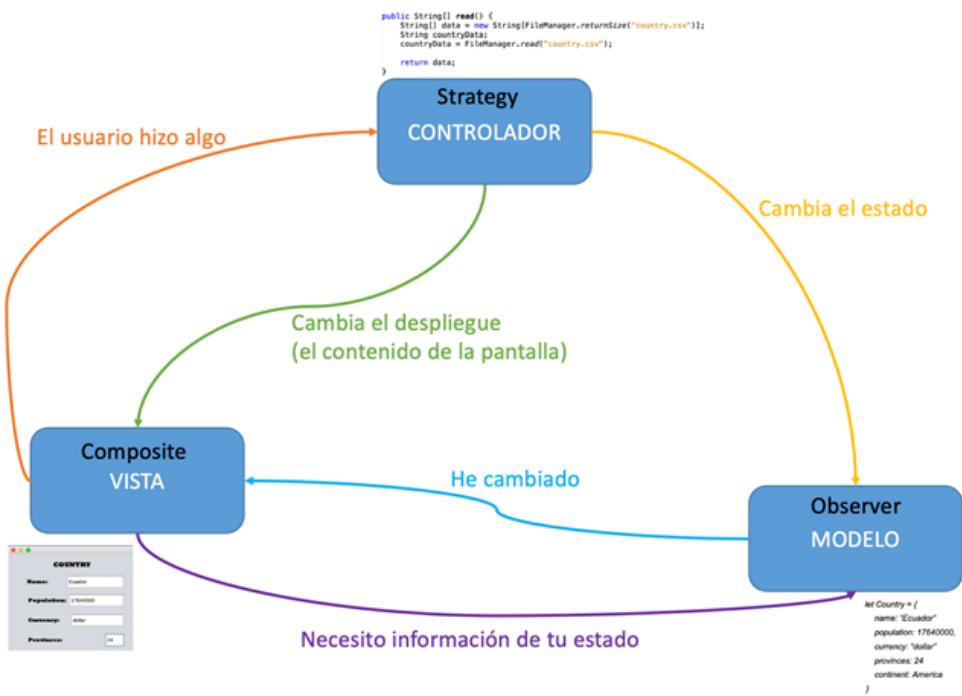
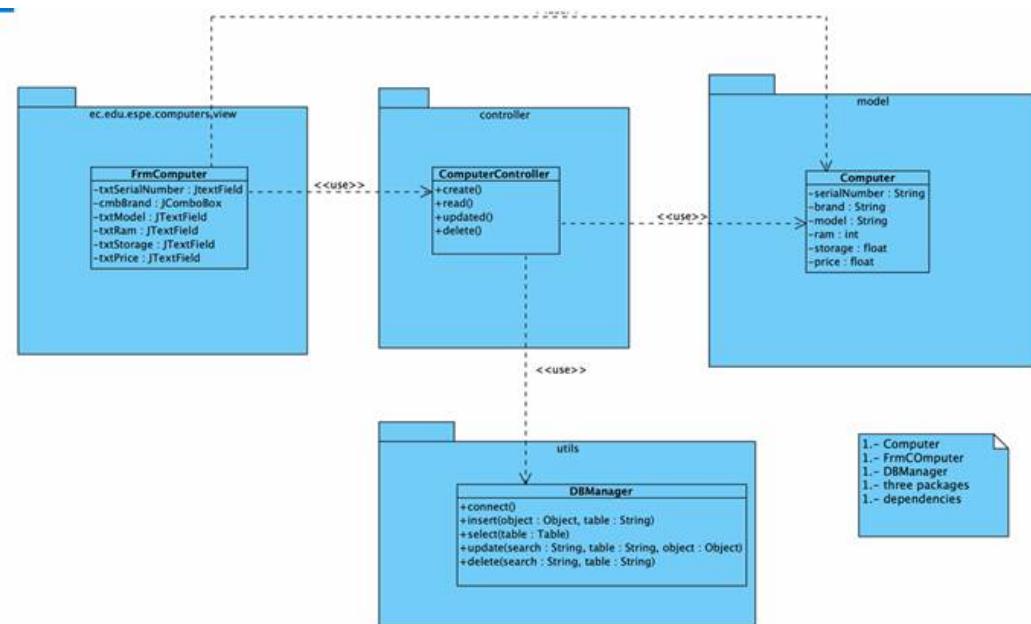


Figura 9

Diagrama UML de una Aplicación Orientada a Objetos siguiendo el patrón MV



3.2 Introducción al Desarrollo Front-End

En la sección anterior (2), se estudió y se desarrolló el Back-End de una aplicación Web, se entiende por back-End el componente de un sistema informático que procesa los datos, ya sea a través de algoritmos y de almacenamiento y recuperación de una base de datos. Sin embargo, el sólo procesamiento de información no es suficiente para que un sistema informático sea útil. El desarrollador debe diseñar y programar un componente que muestre los datos al usuario y que además le permita interactuar con el sistema en caso de ser necesario modificar, crear o eliminar información del sistema.

Este componente permite la interacción con el usuario (interacción humano-computadora) y se conoce como Interfaz de Usuario o front end. Los desarrolladores de front end deben diseñar el contenido de las páginas web, de ser el caso de una página informativa y formularios de entrada y salida de datos de ser el caso de una aplicación transaccional. Pero, estos no son los únicos tipos de aplicaciones web que existen, entonces el diseño de una página Web va más allá de una simple división de la página o de la colocación de entradas de datos en un formulario. Muchas veces, el desarrollador Front-End deberá trabajar en conjunto con un diseñador gráfico para resolver los problemas de interacción con el ser humano y para transmitir lo que desea que la página transmita. El desarrollador Front-End por lo general se encarga de la programación. Para lo cual, al menos debe conocer HTML, JavaScript y CSS. Pero esos lenguajes/tecnologías no son suficientes en la actualidad para ser efectivos y eficientes al momento de programar una página Web. Entonces el desarrollador debe recurrir al uso de frameworks y bibliotecas como React, Vue.js, Bootstrap y muchas más. No obstante, más importante que el uso de lenguajes de programación modernos o tecnologías que aceleren el desarrollo es el diseño de la interfaz de usuario.

3.2.1 Organización del contenido

Las interfaces de usuario deben estar orientadas al usuario, por lo tanto un poco de “UX Research” (User Experience) debe hacerse. Se debe tomar en cuenta dos aspectos: se debe escuchar al usuario y observar al usuario. En ocasiones, la actitud del usuario puede cambiar durante una entrevista para levantar requerimientos y más aún en la presentación de un producto software, del cual, el usuario únicamente verá la interfaz de usuario. Por lo tanto, una interfaz de usuario debe ser lo más simple posible, no se debe incrementar la complejidad con contenido innecesario o amontando información en la misma página, se debe seguir el principio de diseño KISS (Keep it simple, stupid).

Básicamente, se debe seguir cuatro leyes (Raskin, 2000): 1ra Ley: Una computadora no debe dañar su trabajo, o a través de su inacción permitir que su trabajo no llegue a hacer daño. 2da Ley: Una computadora no debe gastar su tiempo o requerir que usted haga más trabajo del estrictamente necesario. 3ra Ley: Una interfaz es humana (humanizada) si es responsive a las necesidades del humano y considera las debilidades humanas. 4ta Ley: El usuario debería marcar el ritmo de la interacción. Adicional a las 4 leyes, el desarrollador debe pensar si el diseño es centrado en el Humano, centrado en el Usuario o centrado en el Uso y en el uso de patrones (los nombres de los patrones son nombres propios por lo tanto no se traducen al español). Para diseñar interfaces de usuario, se puede seguir los patrones de comportamiento del usuario (Tidwell et al., 2020):

Tabla 2

Patrones de Comportamiento del Usuario

Patrón (Nombre Propio)	Descripción	Ejemplo (en Inglés)
Safe Exploration	Déjeme explorar sin Deshacer (Undo) perderme o meterme en multnivel problemas.	

Las interfaces de usuario deben estar orientadas al usuario, por lo tanto un poco de “UX Research” (User Experience) debe hacerse. Se debe tomar en cuenta dos aspectos: se debe escuchar al usuario y observar al usuario. En ocasiones, la actitud del usuario puede cambiar durante una entrevista para levantar requerimientos y más aún en la presentación de un producto software, del cual, el usuario únicamente verá la interfaz de usuario. Por lo tanto, una interfaz de usuario debe ser lo más simple posible, no se debe incrementar la complejidad con contenido innecesario o amontando información en la misma página, se debe seguir el principio de diseño KISS (Keep it simple, stupid).

Básicamente, se debe seguir cuatro leyes (Raskin, 2000): 1ra Ley: Una computadora no debe dañar su trabajo, o a través de su inacción permitir que su trabajo no llegue a hacer daño. 2da Ley: Una computadora no debe gastar su tiempo o requerir que usted haga más trabajo del estrictamente necesario. 3ra Ley: Una interfaz es humana (humanizada) si es responsive a las necesidades del humano y considera las debilidades humanas. 4ta Ley: El usuario debería marcar el ritmo de la interacción. Adicional a las 4 leyes, el desarrollador debe pensar si el diseño es centrado en el Humano, centrado en el Usuario o centrado en el Uso y en el uso de patrones (los nombres de los patrones son nombres propios por lo tanto no se traducen al español). Para diseñar interfaces de usuario, se puede seguir los patrones de comportamiento del usuario (Tidwell et al., 2020):

Tabla 2
Patrones de Comportamiento del Usuario

Patrón (Nombre Propio)	Descripción	Ejemplo (en Inglés)
Safe Exploration	Déjeme explorar sin Deshacer perderme o meterme en multnivel problemas.	(Undo)
Instant Gratification	Quiero conseguir algo Subir imágenes ahora, no después	
Satisficing	Esto es suficiente. No quiero pasar tiempo aprendiendo otras maneras de hacerlo	Un playlist de Youtuble
Changes in Midstream	Cambié de idea acerca de lo que estaba haciendo	Ctrl-Z

Patrón (Nombre Propio)	Descripción	Ejemplo (en Inglés)
Deferred Choices	No quiero contestar ahora, sólo déjeme diseñadas terminar	Encuestas bien
Incremental Construction	Permítame cambiar esto. Eso no se ve bien, permítame cambiarle de nuevo.	Capas de Photoshop
Habituation	Ese gesto trabaja donde sea, ¿por qué no funciona aquí?	Atajos en Office
Microbreaks	Estoy esperando el tren. Déjeme hacer algo útil por unos minutos.	Juegos como snake, minesweeper, solitario
Spatial Memory	Yo juro que ese botón estaba ahí hace un minuto. ¿A dónde se fue?	Email/Desktop
Prospective Memory	Pongo esto aquí para recordarlo luego y poder resolverlo.	Bookmarks/email
Streamlined Repetition	Debo repetir eso cuantas veces.	Wizzards/shortcuts (atajos)
Keyboard Only	Por favor no me haga usar el ratón, si no es necesario	Shortcuts
Other People's advice	¿Qué dice la gente acerca de esto?	Juegos
Personal Recommendations	Mi amigo me dijo que lea esto, así que debe ser bueno	Google Docs

La interacción Humano-Computador (HCI) se concentra en ciertas constantes: La *memoria* (donde el humano almacena la información), el *sistema perceptual* (para sentir el medio ambiente), el *sistema cognitivo* (para procesar o para pensar), el *sistema motor* (para ejecutar una respuesta). Cada una de ellas tiene sus limitaciones, por ejemplo la memoria de corta

duración solo puede almacenar hasta siete cosas de manera automática, de ahí la regla de poner 5±2 elementos en una interfaz de usuario para que sea más fácil recordar las cosas, si se tiene más de siete elementos, se podría dividir la información en trozos o partes, muchas veces esto se logra con el uso de tabs en una misma página (pantalla) o de divs para separar la información. Las limitaciones de percepción se relacionan con la pérdida de atención cuando tiene que concentrarse en otras cosas al mismo tiempo, la interfaz de usuario debería ser llamativas y absorber al usuario para que realice su tarea a pesar de las distracciones que puedan haber a su alrededor. La mayor limitación de cognición es que el humano no es multitarea y maneja un solo flujo de pensamiento.

Algunos principios clave a la hora de diseñar interfaces de usuario con los factores humanos en mente son (Raskin, 2000):

- Evite listas largas de elecciones múltiples
 - agrúpelas.
- Evite modos,
 - mantenga el mismo modo, por ejemplo, mayúsculas.
- Habilite la automatización de comportamientos que no requieran procesamiento de decisiones significativas.
 - Ctrl-S/Ctrl-G
- La facilidad de aprendizaje no debería darse a costa de mejorar la productividad,
 - reduzca el número de clics.
- Provea “en el mundo” conocimiento en lugar de confiar en la memoria “en recuperación”
 - Encuéntrelo en la IU, no en la memoria
- Use términos y conceptos que sean familiares para el usuario, siempre que sea posible
 - Use terminología del dominio del problema.

En base a los conceptos revisados, ahora se debe pensar en la organización del contenido de una interface de usuario. Figura 10 y Figura 11 representan dos maneras de organizar el contenido de la interface de usuario de dos aeronaves de diferentes épocas, la primera un helicóptero de alrededor de los 50s 60s, en donde la electrónica permitía maniobrar las naves, la segunda es de una avión de pasajeros de los 2000s, en donde la computación tomó un aspecto importante en el diseño de los instrumentos y los paneles para los pilotos.

En ambos casos, el contenido se encuentra organizado de acuerdo a la necesidad de los pilotos y a la relación entre los datos que se observan. A

pesar de ser información analógica (Figura 10), los elementos que componen la consola del avión siguen de alguna manera los principios antes mencionados. Por ejemplo, las agrupaciones de instrumentos de medición no tienen más de 7 elementos. Sin embargo, la cantidad de información que se puede ver al instante es abrumadora, puede no permitir la concentración en una tarea específica. En el segundo caso (Figura 11), los elementos se encuentran mejor distribuidos, pero siguen los mismos principios. En ambos casos, las interfaces de usuario están orientadas al uso, al usuario, al humano.

Figura 10

Organización del contenido de un panel de control de un helicóptero



Figura 11

Organización del Contenido de la Consola de un Avión Comercial de los 2000s



Para dar inicio al diseño de una interfaz de usuario, se debe primero determinar el contenido en base a los casos de uso, a los diagramas de estado, a los diagramas de estado, en general en base a los requerimientos del usuario. Se debe explorar los escenarios actuales, para poder capturar las metas del usuario y determinar los actores (usuarios) correspondientes. Con la lista de datos y acciones, se procede a agruparlos, rompiendo los escenarios en grupos de interacción lógicos. Se debe buscar grupos recurrentes, mantenerlos cerca si se relacionan, mantenerlos bajamente acoplados, mantenerlos cohesivos, los elementos se clasifican y se generalizan, con esta primera idea, se vuelve a revisar las metas y los requerimientos del usuario para saber si no queda algún dato o acción por cubrir. Luego se procede a hacer una lluvia de ideas para hacer bosquejos y visualizar el diseño. Se pueden usar herramientas de diseño, como se lo hizo en el hands-on lab del tema 2, o se puede programar prototipos (aplicaciones con funcionalidad reducida que demuestran como quedaría la página Web final).

Por ejemplo, considere que se necesita un sistema para la administración de admisiones y revisión de papers (artículos científicos) para una conferencia académica. Responda las siguientes preguntas a usted mismo, (por favor no lea las repuestas, hasta no tener las suyas):

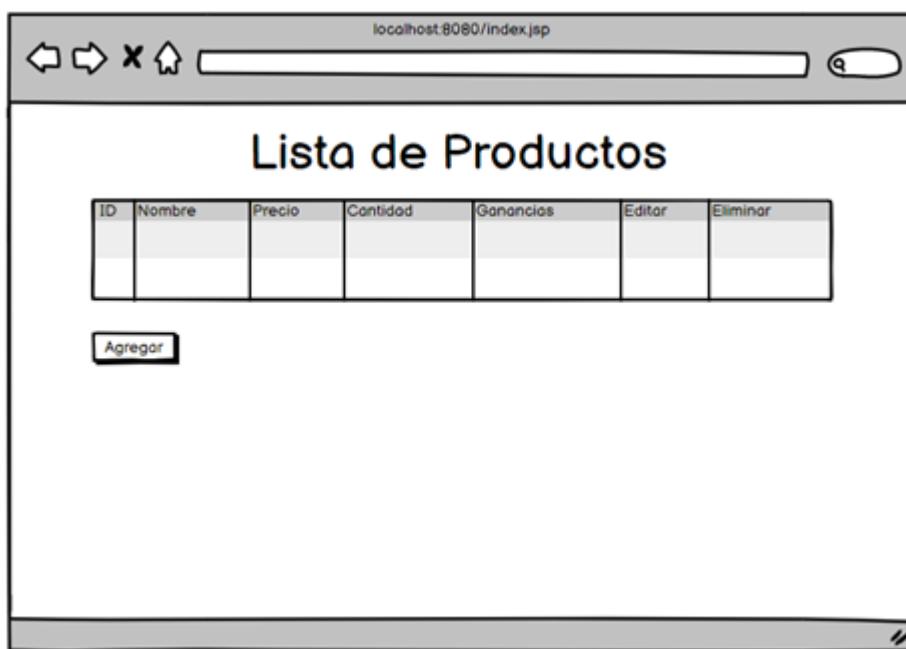
- 1. ¿Cuáles son los actores clave?**
 - 2. ¿Cuáles son las metas del usuario?**
 - 3. ¿Cuáles son los escenarios típicos?**
 - 4. ¿Cuáles son los requerimientos funcionales?**
-
- 1. Revisor, Conferencista, ...**
 - 2. Registrar el paper para su revisión, ...**

3. Paper aprobado, negado, en revisión , aprobado con cambios.
4. El sistema debe permitir el registro de autores y sus papers para su revisión, ...

Para diseñar una interfaz de usuario que sea familiar, se puede usar metáforas, por ejemplo, la oficina de registros, el escritorio de la persona a cargo de los registros, una imagen tridimensional del edificio y las charlas que se darán en cada una de las salas. A continuación un sketch de un sistema de manejo de productos Figura 12. La metáfora en este diseño es una hoja de cálculo con la lista de productos, representado en una tabla.

Figura 12

Sketch: Diseño Preliminar de una Aplicación para manejo de Productos



Existen varios patrones organizacionales para el diseño de interfaces de usuario. Pero primero debemos pensar en lo siguiente: mostrar solo un ítem, mostrar una lista de ítems, proveer herramientas de creación, y facilitar un sola tarea. Continuación la lista de patrones que se podría aplicar en la organización del contenido de una página Web (Tidwell et al., 2020):

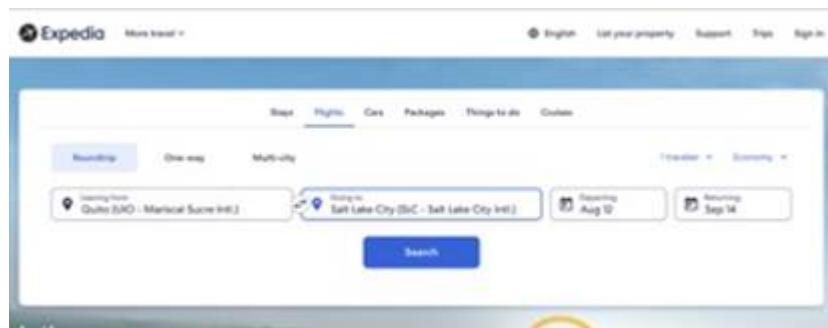
- Patrones para información
 - Future, search, and browse
 - News stream
 - Picture manager
 - Dashboard
 - Settings editor
 - Many workspaces

- Patrones para diseño de ventanas (páginas)
 - Two panel selector
 - Canvas plus palette
 - One-window drilldown
 - Alternative view
- Otros patrones
 - Wizard
 - Extras on demand
 - Intriguing branches
 - Multi-level help

La Tabla 4 de la sección 3.2.3 presenta un resumen de los patrones para organización de contenido con un pequeño ejemplo, que es lo que manejan, cuando se los debe usar y el porque de su existencia. Adicionalmente, las interfaces de usuario deben recibir información del usuario (Figura 13), para lo cual, se debería seguir varios principios: asegúrese de que el usuario entienda lo que se le pide, evite preguntas innecesarias, tenga cuidado con traducciones literales en base al modelo de programación subyacente, haga pruebas de usabilidad, escoja apropiadamente el control de entrada ya que esto afectará directamente en las expectativas del usuario, recuerde que el conocimiento del mundo es a menudo más preciso que el conocimiento en la cabeza.

Figura 13

Ingreso de Datos Para Comprar Pasajes de Vuelo



Ejercicio no evaluado. En las siguientes páginas (Figura 14 y Figura 15) reconozca los patrones de organización de contenido aplicados, recuerde que en cada página se aplicaron más de un patrón de diseño de interfaz de usuario.

Figura 14

Página Web de la Universidad de las Fuerzas Armadas ESPE

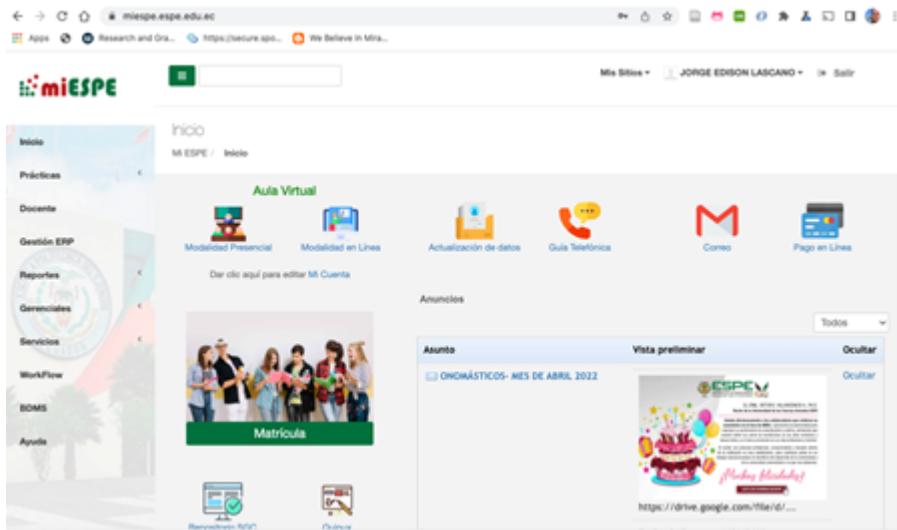
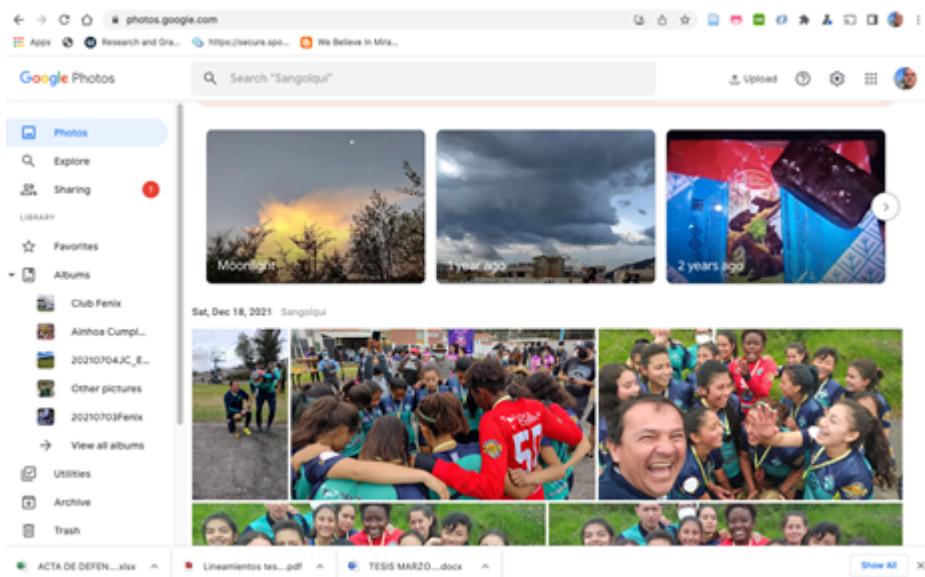


Figura 15

Página Principal de Google Photos de edisonlascano



3.2.2 Navegabilidad

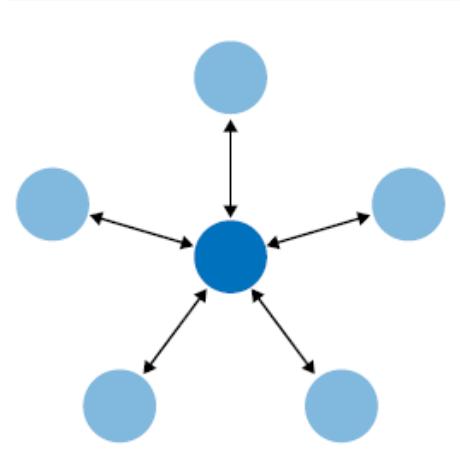
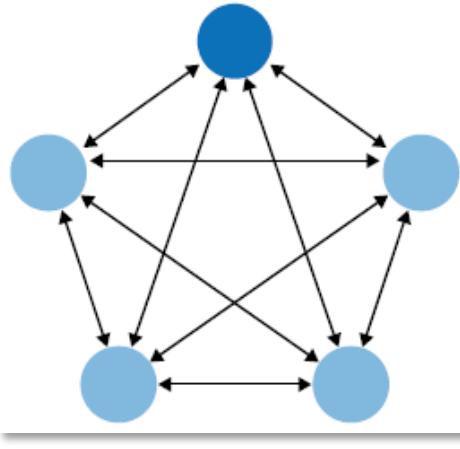
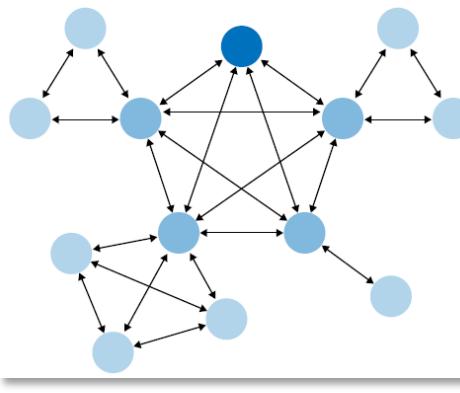
Luego de haber modelado cada una de las páginas de la aplicación, se debe modelar la navegabilidad entre cada una de ellas. Es decir, el como el usuario se va a trasladar de una sección de la aplicación a otra. Por ejemplo, una vez que haya ingresado al sistema, a dónde es redireccionado al usuario, podría ser enviado a un menú principal o a una página en particular, luego de llegar a esa página, se debe facilitar al usuario las herramientas necesarias para que regrese a donde estaba antes o avanzar hacia donde se le quisiera orientar que vaya. El usuario debería hacerse las siguientes preguntas en cada página y su respuesta debería ser inmediata:

1. ¿Dónde estoy ahora?
2. ¿A dónde me quiero ir?
3. ¿Cómo llego allá?
4. ¿Ya llegué a donde quería llegar?

Si el usuario obtiene una respuesta con solo observar los elementos de la aplicación, y recuerda fácilmente de donde vino con tan sólo un vistazo de la interfaz de usuario, su navegabilidad está cumpliendo su objetivo. Para eso se puede usar señales (signposts) que guíen al usuario en caso de no saber a donde ir observando los alrededores. En otras palabras le ayuda a encontrarse al usuario en el contexto de la aplicación. Algunas señales comunes incluyen:

- Títulos de ventanas
- Logotipos
- tres dispositivos como: colores, líneas
- Pestañas (Tabs), siempre el tab actual se encuentra resaltado
- Indicadores de selección

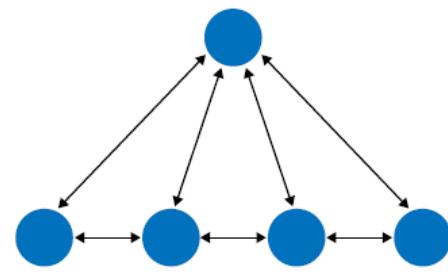
Pensando un poco en la primera pregunta. El desarrollador debería ser capaz de implementar elementos gráficos para avisarle al usuario donde está sin que el usuario se de cuenta. Para esto, de pueden usar varios patrones “You are here”. El usuario con solo mirar el contexto, sabrá donde esta en cualquier momento. Varios de estos patrones son:

Modelo	Representación de navegabilidad
Hub and Spoke	
Fully Connected	
Multi-level	
Stepwise	

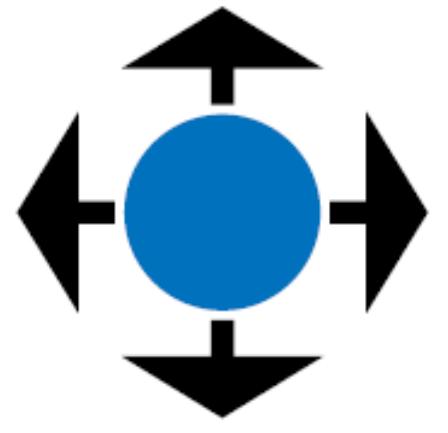
Modelo

Representación de navegabilidad

Pyramid

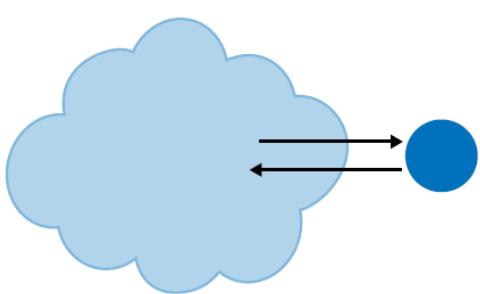


Pan-and-zoom

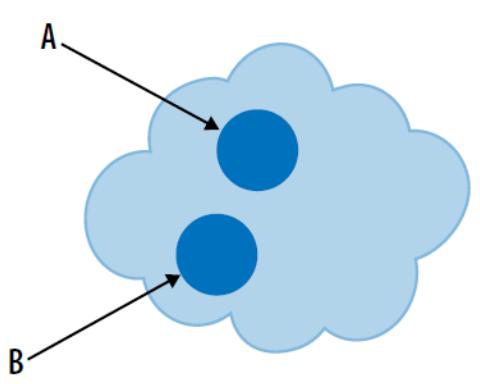


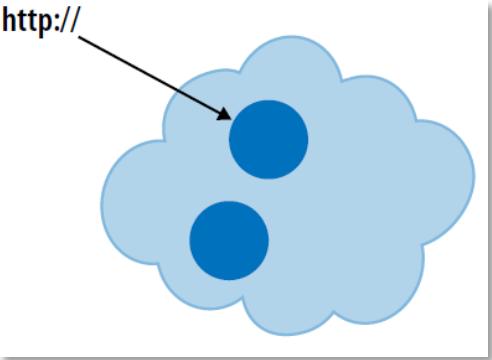
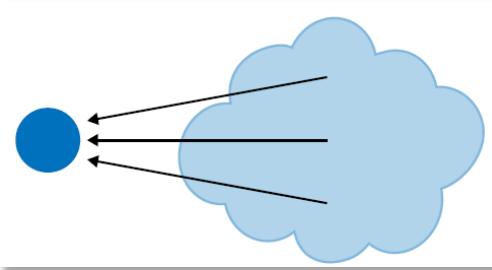
Flat navigation

Modal Panel



Clear Entry Point



Modelo	Representación de navegabilidad
Bookmarks	 <p>A diagram showing a light blue cloud shape containing two dark blue circular nodes. A black arrow points from the text "http://" to the top node.</p>
Escape hatch	 <p>A diagram showing a dark blue circular node on the left connected by four black arrows to a light blue cloud shape on the right.</p>

3.2.3 Patrones de Diseño de UI

Tabla 4

Resumen de Patrones Organizacionales

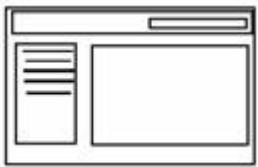
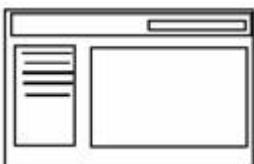
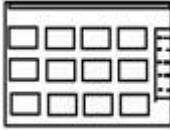
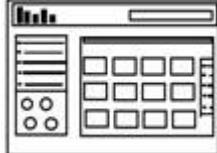
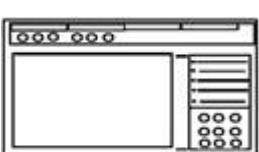
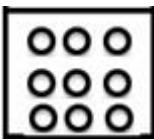
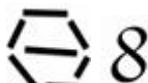
Patrón	¿Qué?	¿Cuándo?	¿Por qué?
Feature, Search, and Browse	Tres elementos en la página principal	Larga lista de ítems	Buscar y navegar van de la mano
			
News Stream	Listas cronológicas en reversa	Email, social networking, blogging.	El último ítem/siguiente ítem es importante.
			
Picture Manager	Thumbnails, vistas de ítems.	Se necesita trabajar con listas de colecciones.	Acciones predecibles en los ítems.
			

Tabla 4*Resumen de Patrones Organizacionales*

Patrón	¿Qué?	¿Cuándo?	¿Por qué?
Feature, Search, and Browse	Tres elementos en la página principal	Larga lista de ítems	Buscar y navegar van de la mano
			
News Stream	Listas cronológicas en reversa	Email, social networking, blogging.	El último ítem/siguiente ítem es importante.
			
Picture Manager	Thumbnails, vistas de ítems.	Se necesita trabajar con listas de colecciones.	Acciones predecibles en los ítems.
			
Dashboard	Despliegue de información condensada en una sola página.	El flujo de información entrante viene de diferentes fuentes.	Estilo de página reconocible y familiar.
			
Canvas Plus Palette	Paleta de íconos y canvas en blanco con objetos.	Editores gráficos.	Mapeo natural (metáfora).
			
Wizard	Se guía al usuario en una actividad.	Tareas largas y complicadas.	Lidiar con “espacios mentales” discretos.
			

Patrón	¿Qué?	¿Cuándo?	¿Por qué?
Settings Editor 	Settings/preferencias bajo demanda.	Sistemas Operativos, apps, configuración de productos, sing-ins.	Cambiar una propiedad sin estar forzado a hacerlo todo.
Alternative Views 	Permita a los usuarios escoger su opción.	Dos feature no pueden ser vistos al mismo tiempo con la misma información.	No puede siempre acomodar todos los posibles escenarios.
Many Workspaces 	Vistas múltiples de diferente información.	Las vistas de los contenidos no están relacionadas.	Personas Multitarea, diferentes actividades en diferentes tiempos (breaks).
Multi-Level Help: Tooltips Help queries Help menu Input prompt Content-specific Online Resource	Técnicas de ayuda ligeras y pesadas.	Aplicaciones complejas y diferentes experiencias de usuario.	Los usuarios avanzados no necesitan lo que necesita un usuario novato.

Patrón	¿Qué?	¿Cuándo?	¿Por qué?
Two-panel Selector	Listas seleccionables con detalles de cada elemento.	Grandes listas limitadas, cada ítem tiene bastante información.	Ayudar al usuario a mantenerse en su contexto.

Otros:

- Extras on Demand
- One-Window Drill Down
- Intriguing Branches

Nota. Los patrones aquí expuestos se encuentran definidos en (Tidwell et al., 2020)

Tabla 5

Patrones de Modelos Navegacionales

Patrón	¿Qué?	¿Cuándo?	¿Por qué?
Clear Entry Points	Pocos puntos de entrada principales (orientado a tareas).	Hay muchos usuarios que usan la página por primera vez.	Guiar al usuario en lo que tiene que hacer primero.
Menu Page	Una lista de enlaces a páginas ricas en contenido. No hay otro contenido significativo.	Se necesita una tabla de contenido y los usuarios saben porque están ahí.	Se necesita mostrar únicamente los enlaces apropiados.

Patrón	¿Qué?	¿Cuándo?	¿Por qué?
Pyramid	Una secuencia de páginas que pueden ser vistas o no vistas en secuencia.	Usted puede ver la secuencia, pero no es obligatorio verlas en orden.	Reduce el número de clics, da libertad al usuario.
Modal Panel	Muestra una página sin opciones de navegación.	Antes de que los usuarios continúen, para forzarlos a ejecutar una acción necesaria.	Los usuarios no lo pueden ignorar y deben hacerlo ahora. “Save Cookies”.
Deep/linked State	Captura el estado de un sitio o una app.	Tiene páginas interactivas.	Los usuarios necesitan volver a estados anteriores. “Use cookies”.
Escape Hatch	Un botón que ayuda a los usuarios a salir de una página.	Un proceso (serial o no serial), pero no hace falta que se completen todos los pasos.	Los usuarios necesitan una salida.

Nota. Los patrones aquí expuestos se encuentran definidos en (Tidwell et al., 2020)

Tabla 6

Patrones Layout+Model

Patrón	¿Qué?	¿Cuándo?	¿Por qué?
Fat Menus	Una larga	Hay muchas	Pueden



lista de menús drop-down con opciones de navegación.

páginas y muchas categorías.

exponer muchas opciones de navegación y esconder complejidad.

Sitemap Footer



Más ... un estándar Web.

En cada pagina de un sitio.

Los usuarios necesitan saltar directamente a cualquier página, una manera de implementar un escape hatch, provee navegación global.

Sign-in Tools



Sign-in, sign-up

Hace falta hacer seguimiento a los usuarios.

Los usuarios buscan esa opción.

Kahoot!

The screenshot shows a "Log in" form for Kahoot!. It has two input fields: "Username or email" and "Password". Below the password field is a link "Forgot password? Reset your password". Underneath the fields are two buttons: "Log in" and "or". Below these are four social media log-in options: "Continue with Google", "Continue with Microsoft", "Continue with Apple", and "Continue with Clever". At the bottom of the form is a link "Don't have an account? Sign up".

Nota. Los patrones aquí expuestos se encuentran definidos en (Tidwell et al., 2020)

Tabla 7

Patrones "You are Here"

Patrón	¿Qué?	¿Cuándo?	¿Por qué?

Patrón	¿Qué?	¿Cuándo?	¿Por qué?
Sequence Maps	Muestra un mapa y un indicador “Here you are”.	Procesos narrativos / wizards / cualquier progreso de actividad.	Dice a los usuarios donde están, donde estuvieron, y hacia donde se dirigen.
Breadcrumbs	Jerarquía navegacional, se muestra el camino (desde – hacia), las migajas de Hanzel y Gretel.	El usuario navega en dos o mas niveles.	Para mostrar el nivel de jerarquía, desde el TOPE al HERE YOU ARE.
Annotated Scrollbar	El Scroll-bar sirve un doble propósito: “you are here” y navegación.	Hay problemas en seguir la información.	Algunas veces la atención del usuario se centra en el scroll-bar.

Nota. Los patrones aquí expuestos se encuentran definidos en (Tidwell et al., 2020)

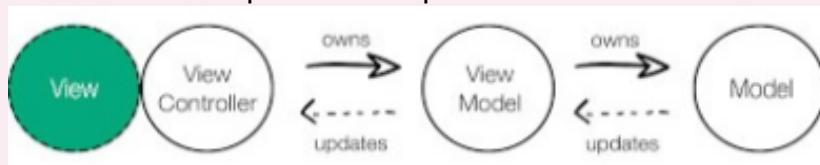
3.3 Herramientas y Entornos/Frameworks de Desarrollo Front-End

El desarrollo Front-End para el web se basa en el uso de HTML, CSS y JavaScript. En el tiempo se han desarrollado otras tecnologías pero han ido desapareciendo del mundo del desarrollo de clientes web, por ejemplo, en una época se creaban Applets que eran aplicaciones cliente que corrían en un navegador web y se programaban usando Java, era la respuesta interactiva de Java a la falta de interacción que ofrecía un naciente cliente web, con el tiempo fue descontinuada por varios motivos, uno de ellos fue la seguridad. Más adelante (1996) aparecen las aplicaciones Macromedia Flash que pasaría a llamarse Adobe Flash que en 2007 fue lanzada por la compañía Adobe que había adquirido a Macromedia en Octubre 3 de 2005, al igual que los Applets, estas aplicaciones se ejecutaban en un browser y eran insertadas como parte de una página Web, eran más transparentes en su uso que los applets, entre otros motivos su mayor problema era el presionar el botón de regreso y se cerraba toda la aplicación. Finalmente en 2017 Adobe deprecó Flash y anunció su “End-Of-Life (EOL). Al final del 2020 y dejó de ofrecer soporte, distribución y actualizaciones de seguridad para el Flash Player. Más adelante en enero de 2021, la mayoría de browsers empezaron a bloquear Flash. Por cuestiones de aplicaciones Legacy, IE11 browser aún soportan Flash, en especial en China.

A pesar de ser conocidos muchos frameworks como frameworks MVC, en realidad, cuando se trata de frameworks de Front End, no se tendría el componente “Controlador”. Entonces, son variaciones del patrón MVC, como por ejemplo: React implementa el VVM del patrón MVVM, como se observa en la Figura 16.

Figura 16

Patrón MVVM Implementado por ReactJS



Nota. La imagen fue tomada de (Vincijanovic, 2018)

3.3.1 Lenguajes básicos UI: HTML, CSS, JavaScript

Los lenguajes de Front-End tienen un objetivo único presentar la información al usuario final. Desde los 90s el Web ha evolucionado desde páginas estáticas que no procesan información, pasando por páginas de informativas, pero dinámicas, tipo wikis, hasta sistemas completos desarrollados para la Web, que poco a poco están reemplazando a las aplicaciones de escritorio. Si bien HTML solo sirve para dar formato al contenido de una página web, JavaScript permite hacer la interfaz de usuario dinámica como por ejemplo validaciones de ingreso de datos, y CSS es una manera de estandarizar la manera en como se deben desplegar varias páginas de manera visual consistente. El desarrollo de una página Web usando las tres tecnologías conlleva un tiempo demasiado largo para su programación. La primera solución para no generar HTML manualmente son las herramientas GUI para desarrollar páginas Web, otras herramientas ofrecen templates para que el desarrollador Front-End no inicie desde cero, estas herramientas entre otros componentes ofrecen, por ejemplo: tablas pre diseñadas, formularios listos para conectarse a servicios REST, páginas de seguridad para ingresar a los sistemas (login) usando validaciones de terceros, o para que el mismo programador desarrolle sus validaciones e ingresos a los sistemas. El principal lenguaje utilizado en este sentido es JavaScript (en el lado del cliente).



3.3.2 Herramientas UI

Como se mencionó anteriormente, hace falta herramientas para el desarrollador, faciliten el diseño de la página y dejen lista la posibilidad de conexión con el back-End. HTML, JavaScript y CSS no tienen costo alguno para los desarrolladores, pero las herramientas si lo tienen en su mayoría, aunque también hay herramientas free y open source. Algunas de estas herramientas se basan en frameworks o bibliotecas populares. Según (Martin, 2020), en el año 2020, las mejores herramientas eran: Sancha Ext JS, Creative Tim, Envato HTML Templates, Myfonts, Elfsight, Npm, TypeScript, CodeKit, WebStorm, HTML5 Boilerplate, en la Tabla 8, se describen algunas de estas herramientas. Por otro lado, de acuerdo a («10 Best Tools For Front-End Web Development», 2021), varias herramientas necesarias para el desarrollo Web para Front End son: Chrome DevTools, HTML5 Boilerplate, Sass, AngularJS, JQuery, Visual Studio Code, Git, TypeScript, Npm Grant.



3.3.2 Herramientas UI

Tabla 8

Herramientas para Desarrollo de Front End Web

Herramienta	Tipo	Componentes ofrecidos
Sancha Ext	JavaScript Framework	HTML5 calendar, grids, trees, lists
Creative Tim	Herramientas web	Elementos BootStrap, Vue.js, React, Angular, Node.js, Laravel usados para web apps y mobile apps.
Envalo HTML Templates	Colección de templates	Templates HTML5, CSS optimizado
Myfonts	Herramienta web para diseñadores gráficos	Fonts profesionales,
Elfsight	Herramienta web	Componentes para insertar contenido de Twitter, Facebook, Pinterest, Instagram, etc.
Npm	Administrador de paquetes de Node.js para JavaScript	Comandos para interactuar con repositorios
TypeScript	Lenguaje scripting de front end que es open-source	



3.3.2 Herramientas UI

Como se mencionó anteriormente, hace falta herramientas para el desarrollador, faciliten el diseño de la página y dejen lista la posibilidad de conexión con el back-End. HTML, JavaScript y CSS no tienen costo alguno para los desarrolladores, pero las herramientas si lo tienen en su mayoría, aunque también hay herramientas free y open source. Algunas de estas herramientas se basan en frameworks o bibliotecas populares. Según (Martin, 2020), en el año 2020, las mejores herramientas eran: Sancha Ext JS, Creative Tim, Envato HTML Templates, Myfonts, Elfsight, Npm, TypeScript, CodeKit, WebStorm, HTML5 Boilerplate, en la Tabla 8, se describen algunas de estas herramientas. Por otro lado, de acuerdo a («10 Best Tools For Front-End Web Development», 2021), varias herramientas necesarias para el desarrollo Web para Front End son: Chrome DevTools, HTML5 Boilerplate, Sass, AngularJS, JQuery, Visual Studio Code, Git, TypeScript, Npm Grant.



3.3.3 Entornos WYSIWYG

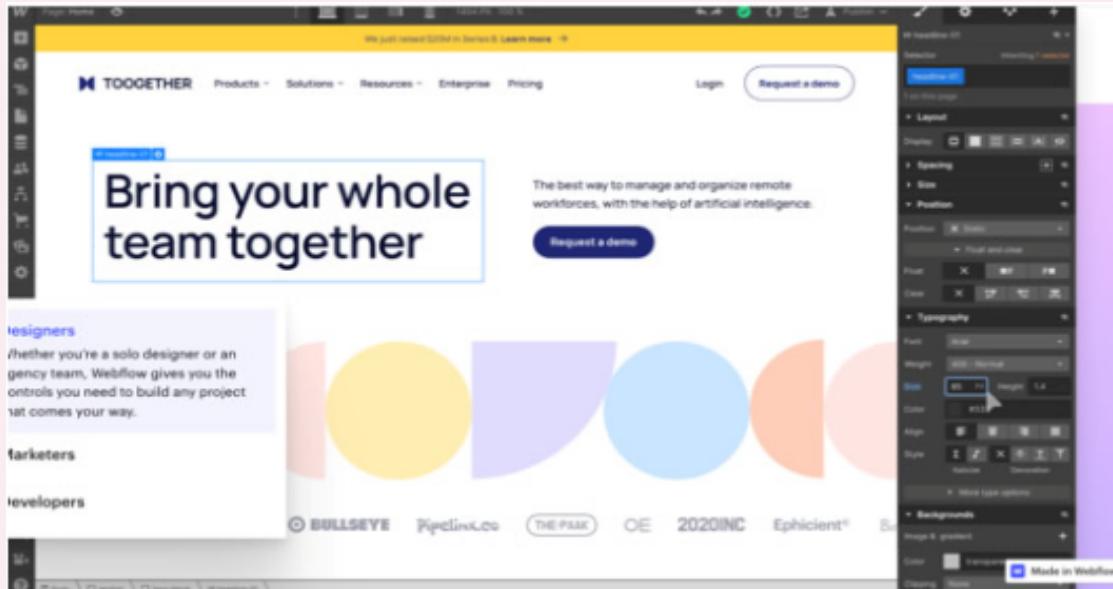
El desarrollador de aplicaciones Web usa en general herramientas de programación o que implican cierto nivel de conocimiento para editar los archivos .html, .js y .css. Sin embargo, para un diseñador Front-End le sería más útil una herramienta que le permita manipular los componentes gráficos de una manera visual, antes que programática. Esta área es cubierta para las herramientas WYSIWYG (What You See Is What You Get), por ejemplo Webflow (Webflow, s. f.), que permite seleccionar colores con color-pickers en lugar de escribir el número hexadecimal de un color, #FF0000 representa el color rojo (RED). Este color se lo define siguiendo el modelo de colores RGB, donde cada dos dígitos del grupo de 6 dígitos representa un color del modelo RGB (Red, Green, Blue), 00 representa ausencia de luz, por lo tanto #000000 representa el color negro. #FFFFFF representa todos los focos prendidos al máximo, por lo tanto representa el color blanco (todos los focos prendidos). Un ejemplo del uso de webflow se presenta en la Figura 18, la cual es una captura de su demo presentado en la página web de la aplicación (<https://webflow.com/>).





3.3.3 Entornos WYSIWYG

Figura 18
Ejemplo de Uso de una Herramienta WYSIWYG



Nota. La imagen es una captura de la página oficial de la herramienta webflow (<https://webflow.com/>)

3.3.3 Entornos WYSIWYG

El desarrollador de aplicaciones Web usa en general herramientas de programación o que implican cierto nivel de conocimiento para editar los archivos .html, .js y .css. Sin embargo, para un diseñador Front-End le sería más útil una herramienta que le permita manipular los componentes gráficos de una manera visual, antes que programática. Esta área es cubierta para las herramientas WYSIWYG (What You See Is What You Get), por ejemplo Webflow (Webflow, s. f.), que permite seleccionar colores con color-pickers en lugar de escribir el número hexadecimal de un color, #FF0000 representa el color rojo (RED). Este color se lo define siguiendo el modelo de colores RGB, donde cada dos dígitos del grupo de 6 dígitos representa un color del modelo RGB (Red, Green, Blue), 00 representa ausencia de luz, por lo tanto #000000 representa el color negro. #FFFFFF representa todos los focos prendidos al máximo, por lo tanto representa el color blanco (todos los focos prendidos). Un ejemplo del uso de webflow se presenta en la Figura 18, la cual es una captura de su demo presentado en la página web de la aplicación (<https://webflow.com/>).



3.3.4 Frameworks y Bibliotecas Front-End por ejemplo: React

Existe una serie de frameworks y bibliotecas para desarrollo Web Front-End. Por ejemplo: Vue.js, Angular, Svelte, AngularJS, BootStrap, Meteor. A diferencia de los Frameworks para Back-End que ayudan en el desarrollo de los servicios Web o de las aplicaciones Web, estos Frameworks permiten usar patrones y templates en el desarrollo de las Interfaces de usuario, por ejemplo, existen templates para la creación de tablas que recuperan datos de un servicio Web. Si no existieran estas herramientas, el desarrollador Front-End debería programar la tabla usando HTML, es decir a través de la etiquete `<table>` `</table>`, y sus múltiples etiquetas embebidas `<th>`, `<td>`, `<tr>`, y debería formatear la tabla usando CSS, adicionalmente, deberá usar JavaScript para su interacción con el Back-End (controller).

En resumen, un framework de Front-End (CSS + JavaScript) es un conjunto de paquetes con código homogéneo pre-establecido en una estructura predefinida de carpetas y archivos. Ofrecen al desarrollador la base para desarrollar con flexibilidad un diseño final. Por lo general un framework de Front-End se compone de los siguientes componentes:

- Una grilla que permite establecer un diseño del sitio a desarrollar de una manera fácil.
- Estilos de Fuentes bien definidos basados en su propósito en particular, tipografía diferentes para cabeceras y para contenido.
- Componentes pre-construidos como botones, barras de navegación, y paneles.

Una de las bibliotecas más usadas en la actualidad es ReactJS o también conocido como React. React es una biblioteca JavaScript que ayuda a crear interfaces de usuario para aplicaciones web y para aplicaciones móviles. Presenta una colección de componentes reutilizables. Uno de los conceptos más usados en la actualidad para el desarrollo de páginas Web se conoce como AJAX (Asynchronous JavaScript and XML), la aplicación de AJAX permite que el envío y recepción de información desde la página sea asíncrona, de esta manera se puede tener páginas que solo actualizan una parte de ella, es decir la sección de la página donde se muestra la información cambiante, mientras tanto, el resto de la página se mantiene sin alteraciones.

3.3.4 Frameworks y Bibliotecas Front-End por ejemplo: React

La mayoría de frameworks para front-end web incluyen un paquete de ruteo (routing) que habilita la actualización de secciones de la página cuando el usuario da clic en varios enlaces que se encuentren presentes en la página. Un router en un framework front-end escucha los cambios en el URL (el controller del MVC) y mantiene la aplicación en sincronía renderizando los componentes de la vista correspondiente. En vista que React no provee de un componente Router, entonces se debería buscar una, por ejemplo React-Router (Ganatra, 2018).

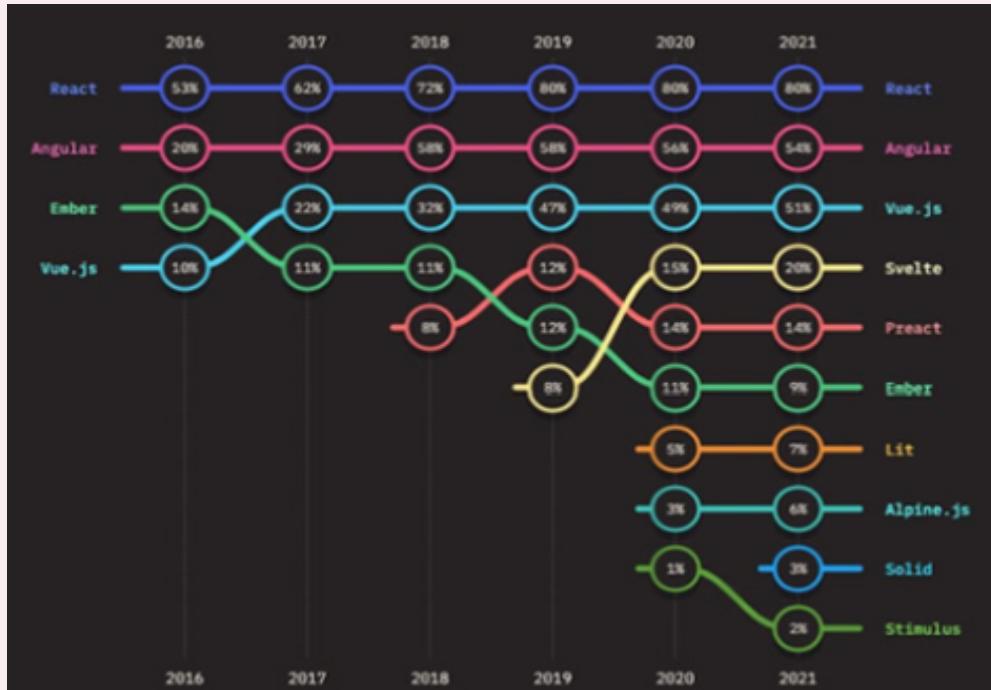
Las posibilidades para el desarrollador de Front End son muchas, entre ellas, por ejemplo usar React (biblioteca) y añadir bibliotecas adicionales (como React-Router) cuando se necesita algo que React no provee. Otra posibilidad sería que el desarrollador programe esa funcionalidad, pero es no es el objetivo actual del desarrollo web, su objetivo para tener un desarrollo más eficiente es la reutilización de código. La otra opción es el uso de Frameworks que en base a las bibliotecas disponibles ofrecen una funcionalidad más completa para el desarrollador, no sólo son bibliotecas, sino también utilitarios y paquetes. El estudiante debería realizar su investigación y aprender a usar las nuevas bibliotecas y los frameworks de desarrollo. En este documento se presenta al final un taller (hands-on-lab) para desarrollar una aplicación web (Front End) utilizando React y BootStrap, pero como la tecnología y los lenguajes de programación evolucionan, es posible que para cuando usted esté estudiando este documento o tomando la asignatura, ya haya aparecido una nueva herramienta de desarrollo web, y también es muy posible que otras hayan desaparecido. A continuación se presentan dos gráficos, tomados de (The State of JS 2021, s. f.), en los cuales se observa que a pesar de ser los más usados React, Angular y Vue.js (FIGURA), se observa que la satisfacción de los desarrolladores al usar Angular ha descendido (FIGURA), razón por la cual es muy probable que algunos desarrolladores dejen de usar Angular en un tiempo.



3.3.4 Frameworks y Bibliotecas Front-End por ejemplo: React

Figura 19

Ranking de Uso de Frameworks Web Front-End



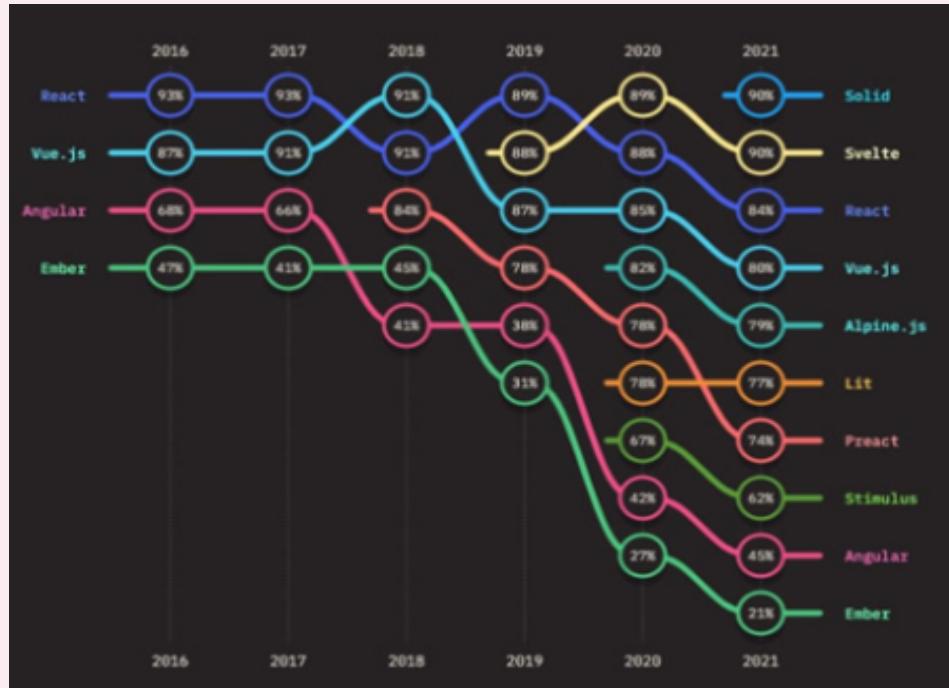
Nota. Gráfico tomado de (The State of JS 2021, s. f.)



3.3.4 Frameworks y Bibliotecas Front-End por ejemplo: React

Figura 20

Ranking de Satisfacción de Frameworks Web Front-End



Nota. Gráfico tomado de (The State of JS 2021, s. f.)

3.3.4 Frameworks y Bibliotecas Front-End por ejemplo: React

Existe una serie de frameworks y bibliotecas para desarrollo Web Front-End. Por ejemplo: Vue.js, Angular, Svelte, AngularJS, BootStrap, Meteor. A diferencia de los Frameworks para Back-End que ayudan en el desarrollo de los servicios Web o de las aplicaciones Web, estos Frameworks permiten usar patrones y templates en el desarrollo de las Interfaces de usuario, por ejemplo, existen templates para la creación de tablas que recuperan datos de un servicio Web. Si no existieran estas herramientas, el desarrollador Front-End debería programar la tabla usando HTML, es decir a través de la etiquete `<table>` `</table>`, y sus múltiples etiquetas embebidas `<th>`, `<td>`, `<tr>`, y debería formatear la tabla usando CSS, adicionalmente, deberá usar JavaScript para su interacción con el Back-End (controller).

En resumen, un framework de Front-End (CSS + JavaScript) es un conjunto de paquetes con código homogéneo pre-establecido en una estructura predefinida de carpetas y archivos. Ofrecen al desarrollador la base para desarrollar con flexibilidad un diseño final. Por lo general un framework de Front-End se compone de los siguientes componentes:

- Una grilla que permite establecer un diseño del sitio a desarrollar de una manera fácil.
- Estilos de Fuentes bien definidos basados en su propósito en particular, tipografía diferentes para cabeceras y para contenido.
- Componentes pre-construidos como botones, barras de navegación, y paneles.

Una de las bibliotecas más usadas en la actualidad es ReactJS o también conocido como React. React es una biblioteca JavaScript que ayuda a crear interfaces de usuario para aplicaciones web y para aplicaciones móviles. Presenta una colección de componentes reutilizables. Uno de los conceptos más usados en la actualidad para el desarrollo de páginas Web se conoce como AJAX (Asynchronous JavaScript and XML), la aplicación de AJAX permite que el envío y recepción de información desde la página sea asíncrona, de esta manera se puede tener páginas que solo actualizan una parte de ella, es decir la sección de la página donde se muestra la información cambiante, mientras tanto, el resto de la página se mantiene sin alteraciones.

3.4 Seguridad Web

La información que se encuentra siendo transmitida en el Web por aplicaciones Web o por servicios Web puede ser fácilmente interceptada por agentes maliciosos que de una u otra forma buscan realizar ataques informáticos, en especial el ataque man-in-the-middle (MiM), pretendiendo ser el receptor final de un mensaje, inclusive en algunos casos alterando los datos del mensaje final (Bojinov, 2018). Por otro lado, cuando se programa micro servicios, ellos se encuentran en un contexto, y la seguridad es relegada a gateways de administración de APIs, esto permite que los desarrolladores se concentren en la programación de micro servicios sin considerar tópicos especiales como la gobernanza o la seguridad.

Sin embargo de lo anterior, el desarrollador debe proveer cierto nivel de seguridad a sus aplicaciones expuestas en el Web a un gran número de requests. A pesar de que la gran mayoría de clientes Web únicamente proveen información consultada del repositorio de datos, y no se verían afectados los datos. pero la fuga de información también es un punto a considerar cuando se publica aplicaciones Web.

Por ejemplo, los agentes maliciosos tendrían acceso a direcciones de correo electrónico, a nombres de personas y sus familiares, etc. Entonces el protocolo http no es suficiente y puede darse fuga de información. Para prevenir de cierta manera que estos casos sucedan, se debe usar TLS (Transport Layer Security) a través del protocolo HTTPS, para encriptar los datos transmitidos, de esta manera se asegura que solo los usuarios con la clave correcta de descifrado puedan consumir los datos expuestos por los servicios web. Algunos features que se deben implementar para asegurar los datos son:

- Autenticación básica
- Autenticación básica basada en pasaportes
- Autenticación básica basada en pasaportes de terceros
- Autorización
- Transport Layer Security



CC

< 15/32 >



3.4 Seguridad Web

Existen varios métodos de autenticación, por ejemplo, la autenticación básica se basa en el envío de datos de autenticación en la cabecera HTTP (request) que provee de las credenciales del usuario necesarias. Por otro lado, el servidor puede responder (reply) con una cabecera pidiendo que el cliente se autentique explícitamente. De darse el caso negativo, el servidor puede enviar un mensaje HTTP/1.1 401 Unauthorized, en el caso positivo el mensaje puede enviar un código 200 OK, y un objeto Json con los datos necesarios como login, name.

En la actualidad, los desarrolladores de aplicaciones web prefieren que la validación de usuarios se realice utilizando APIs de autenticación terceros, donde los usuarios finales tienen sus nombres y claves y pueden identificarse con servicios públicos (APIs públicas) como por ejemplo, Facebook, LinkedIn, Twitter, Gmail.

Las diferentes tecnologías de desarrollo proveen de bibliotecas listas para programar la autenticación de las aplicaciones o los servicios Web, por ejemplo Node.js provee un middleware llamado Passport creado especialmente para casos en que se puede seleccionar entre varios métodos de autenticación, en vista que es modular, permite usar diferentes estrategias como autenticación del mismo sistema, autenticación a través de redes sociales como Facebook, autenticación usando OAuth.



3.4 Seguridad Web

La información que se encuentra siendo transmitida en el Web por aplicaciones Web o por servicios Web puede ser fácilmente interceptada por agentes maliciosos que de una u otra forma buscan realizar ataques informáticos, en especial el ataque man-in-the-middle (MiM), pretendiendo ser el receptor final de un mensaje, inclusive en algunos casos alterando los datos del mensaje final (Bojinov, 2018). Por otro lado, cuando se programa micro servicios, ellos se encuentran en un contexto, y la seguridad es relegada a gateways de administración de APIs, esto permite que los desarrolladores se concentren en la programación de micro servicios sin considerar tópicos especiales como la gobernanza o la seguridad.

Sin embargo de lo anterior, el desarrollador debe proveer cierto nivel de seguridad a sus aplicaciones expuestas en el Web a un gran número de requests. A pesar de que la gran mayoría de clientes Web únicamente proveen información consultada del repositorio de datos, y no se verían afectados los datos. pero la fuga de información también es un punto a considerar cuando se publica aplicaciones Web.

Por ejemplo, los agentes maliciosos tendrían acceso a direcciones de correo electrónico, a nombres de personas y sus familiares, etc. Entonces el protocolo http no es suficiente y puede darse fuga de información. Para prevenir de cierta manera que estos casos sucedan, se debe usar TLS (Transport Layer Security) a través del protocolo HTTPS, para encriptar los datos transmitidos, de esta manera se asegura que solo los usuarios con la clave correcta de descifrado puedan consumir los datos expuestos por los servicios web. Algunos features que se deben implementar para asegurar los datos son:

- Autenticación básica
- Autenticación básica basada en pasaportes
- Autenticación básica basada en pasaportes de terceros
- Autorización
- Transport Layer Security



CC

< 15/32 >



3.4.1 Sesiones

Una vez autenticados los usuarios, a través de sus redes sociales o APIs de autenticación, un identificador único de usuario debe almacenarse en una sesión, y el servicio (y el cliente también) debe manejar esta sesión apropiadamente, para que la información que el servicio produce o cambia no pueda ser accedida o cambiada sin una correcta autenticación, evitando de esta manera, el ataque MiM. Por lo general, la tarea de encriptación (cifrado) y descifrado se lo hace usando un par de claves, una pública y una secreta (privada/token) que esta asociada con el API de autenticación. Por ejemplo, si usáramos LinkedIn se debería ir al siguiente enlace <http://www.linkedin.com/secure/developer>, y llenar la información necesaria para aplicar como desarrollador. La aplicación LinkedIn retornará una clave secreta y un token para habilitar la autenticación. En Node.js, para habilitar el almacenamiento de sesiones se lo debe hacer con Passport y Express. Se inicializa el middleware Express session, antes de inicializar Passport y su middleware session. Como se observe en la siguiente sección de código:

```
app.use(express.session());  
app.use(passport.initialize());  
app.use(passport.session());
```

La descripción completa del manejo de autenticación y sesiones y el código completo Node.js, se encuentra en la página 147 de “RESTful Web API Design with Node.js 10, Third Edition” (Bojinov, 2018)

3.4.2 OAuth

OAuth es un estándar abierto para autorización de terceros, define un protocolo de delegación utilizado para proveedores de autorización. Por ejemplo, se podría usar OAuth para autenticar credenciales con Twitter.

OAuth utiliza tokens especiales, que una vez entregados por una aplicación, identifican al usuario, en lugar de a las credenciales del usuario. Los principales actores que interactúan en OAuth son un usuario que interactúa con la aplicación web, esta aplicación consume un servicio REST desde un sistema back-end que provee algún tipo de datos. A su vez, la aplicación web delega la autorización a un servidor de autorización de terceros. A continuación el flujo del proceso de OAuth en un escenario ejemplo (Bojinov, 2018):

1. El usuario solicita una aplicación web que requiere autenticación para establecer comunicación con el servicio back-end. En este paso, el usuario aún no ha sido autenticado, por lo tanto es redirigido a una página donde se le solicita sus credenciales, para que sean validadas por un tercero.
2. Luego que se ha autenticado satisfactoriamente, el servidor de autenticación emite un código de autorización que es una combinación de el client-id y la clave enviada por el proveedor, un token es proveído y tiene un tiempo de vida limitado.
3. La aplicación web utiliza el token de autenticación para la autenticación hasta que este expira. Cuando el token expira se debe solicitar un nuevo token utilizando el código proveído anteriormente

3.4.3 APIs de Autenticación



En general, varias redes sociales, Twitter, Facebook, LinkedIn ofrecen servicios de autenticación a través de sus APIs públicas. Y lo hacen a través de cuatro métodos bien conocidos: HTTP Basic Authentication, se debe enviar el usuario y la clave a través del uso de HTTP con cada llamada a la API; API Key Authentication, cada proveedor crea sus propias keys de autenticación para los desarrolladores y se envían en cada pedido (request), la API genera una clave secreta (secuencia de números) lo suficientemente larga para no ser fácilmente adivinada; OAuth Authentication, este framework puede orquestar aprobaciones automáticas entre los dueños de los APIs y el servicio; OR .. NO Authentication, siempre esta la opción de no aplicar autenticación del todo, se lo hace cuando se tiene APIs internas en una empresa, pero esta no es una práctica recomendada. Por ejemplo, Twitter utiliza OAuth2, y como tal el servicio tiene un backend administrativo para registrar aplicaciones nuevas (Herron, 2018). El propósito es describir la aplicación al servicio para que pueda ser reconocida correctamente cuando se use el token de autorización. Twitter requiere para el proceso: el nombre de la aplicación, una frase descriptiva, el URL del sitio web, y el URL callback hacia donde volver luego de una autenticación satisfactoria.

Hands-on Lab: Uso de React para consumo de una API

El siguiente tutorial fue desarrollado por Camila Venegas bajo la supervisión del docente autor Edison Lascano. El código ha sido publicado en un repositorio GitHub y puede ser libremente utilizado y replicado por el lector.

i. Tema:

Consumo de datos de una API Web tipo REST para mostrar los datos en una aplicación de Front-end usando la biblioteca React.

ii. Objetivos:

- Desarrollar una aplicación web Front-end que muestre datos producidos por un API, aplicando los principios de usabilidad y manejo de interfaces gráficas responsivas.
- Aprender a desarrollar aplicaciones Front-end usando bibliotecas como React y sus herramientas.
- Crear una interfaz gráfica amigable con el usuario.

iii. Descripción:

La presente aplicación web tiene como objetivo consumir una API pública (<http://universities.hipolabs.com/search?Ecuador>), y enviar los datos a una aplicación de front-end usando la biblioteca React. El sistema contará con una pantalla de login en la cual se puede ingresar un solo tipo de usuario, la aplicación proporciona una tabla detallada de los atributos obtenidos de la API.

El siguiente tutorial fue desarrollado por Camila Venegas bajo la supervisión del docente autor Edison Lascano. El código ha sido publicado en un repositorio GitHub y puede ser libremente utilizado y replicado por el lector.

i. Tema:

Consumo de datos de una API Web tipo REST para mostrar los datos en una aplicación de Front-end usando la biblioteca React.

ii. Objetivos:

- Desarrollar una aplicación web Front-end que muestre datos producidos por un API, aplicando los principios de usabilidad y manejo de interfaces gráficas responsivas.
- Aprender a desarrollar aplicaciones Front-end usando bibliotecas como React y sus herramientas.
- Crear una interfaz gráfica amigable con el usuario.

iii. Descripción:

La presente aplicación web tiene como objetivo consumir una API pública (<http://universities.hipolabs.com/search?Ecuador>), y enviar los datos a una aplicación de front-end usando la biblioteca React. El sistema contará con una pantalla de login en la cual se puede ingresar un solo tipo de usuario, la aplicación proporciona una tabla detallada de los atributos obtenidos de la API.

Se creará una interfaz gráfica basada en principios de usabilidad, haciendo uso de la biblioteca React y otras herramientas que el ambiente de desarrollo proporciona.

Para la cuál la aplicación web será desarrollada utilizando distintas tecnologías:

- *React*. Biblioteca de JavaScript
- *Node.js*. Entorno de tiempo de ejecución de JavaScript, que permite ejecutar programas en tiempo real.
- *CSS*. Lenguaje de hojas de estilo que permite dar formato a una aplicación web.
- *Visual Studio Code*. Editor de código fuente, que permite programar en varios lenguajes de programación.
- *Axios*. Cliente HTTP basado en promesas para Node.js, ayuda a consumir la API proporcionada.
- *API*. Conjunto de definiciones y protocolos que ofrecen servicios (funciones) a través de una red de computadoras, las APIs permiten que dos o más aplicaciones de software se comuniquen entre sí a través de un conjunto de reglas definidas unilateralmente.

iv. Análisis:

Requisitos Funcionales

La aplicación web deberá cumplir con los siguientes requerimientos:

1. El sistema deberá proporcionar una lista o tabla donde se manejen los datos consumidos de la API.
2. El sistema deberá mostrar un menú de opciones para que el usuario pueda navegar de forma dinámica en la aplicación
3. El sistema deberá contar con una pantalla de login que permita al usuario acceder al sitio principal.

Requisitos No Funcionales

4. El sistema podrá ser utilizado desde cualquier navegador web.
5. El sistema deberá consumir datos de una API, usando un lenguaje de programación o una biblioteca, por ejemplo Node.js.

v. Diseño:

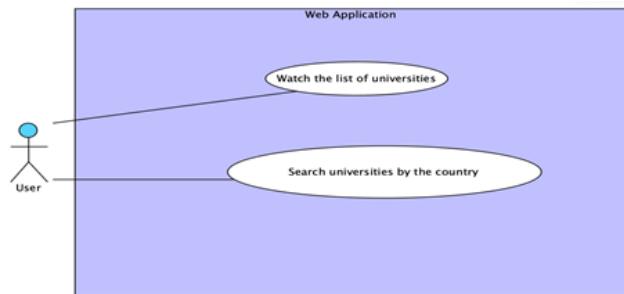
5.1 Diagrama de Casos de Uso

En el siguiente diagrama (Figura 21), se muestran las acciones que pueden realizarse por parte del actor del programa (usuario final), como se explicó al inicio, en este caso solo se cuenta con un usuario el cual puede entrar al programa usando su nombre y contraseña, además podrá visualizar la lista de universidades proporcionada por la API.

En este caso se manejará un solo tipo de usuario, sin embargo, puede ser posible que la aplicación crezca y se necesiten otros tipos de usuarios a futuro, los cuales tendrían asignadas las mismas acciones u otras distintas, y deben ser compatibles con los requisitos obtenidos para este sistema.

Figura 21

Diagrama de Casos de uso



5.2 Diagrama de clases

Figura 22

Diagrama de Clases

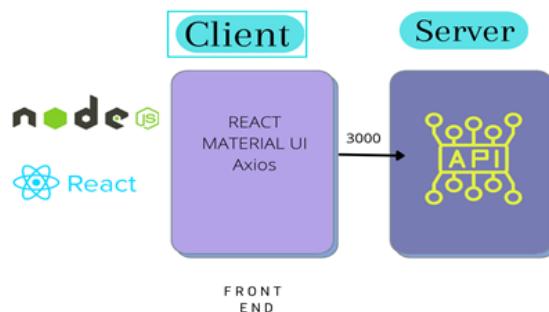


En el siguiente diagrama se muestra la clase que se va a manejar en la aplicación, se debe tomar en cuenta que, se deben detallar los atributos y métodos. Para este ejemplo, se usarán los datos de las universidades proporcionados por una API, la Figura 22 detalla el objeto con los siguientes atributos: country, name, domain, state y web Page.

5.3 Arquitectura

Figura 23

Arquitectura de la aplicación



En la Figura 23, se representa la arquitectura del programa, si bien la aplicación esta enfocada en la capa de front-end, se debe modelar la parte del back-end, que en este caso corresponde al manejo de los datos recuperados a través de la API.

El puerto TCP, donde se ejecutará la aplicación será el puerto 3000, debido a que es el que React toma por defecto, también se muestran las herramientas a usarse, como es el caso de Axios (consumo de API) y material UI (formatos del diseño).

5.4 Diseño de la aplicación

Como primer punto se presenta la API de la cual será consumida la información.

URI: <http://universities.hipolabs.com/search?country>

La Información se retorna en formato JSON, el cual se estudió en las anteriores unidades. Se identifican los atributos del objeto, y de ese modo se los plasma en la aplicación Front-end.

Figura 24

Datos JSON que retorna la API de universidades

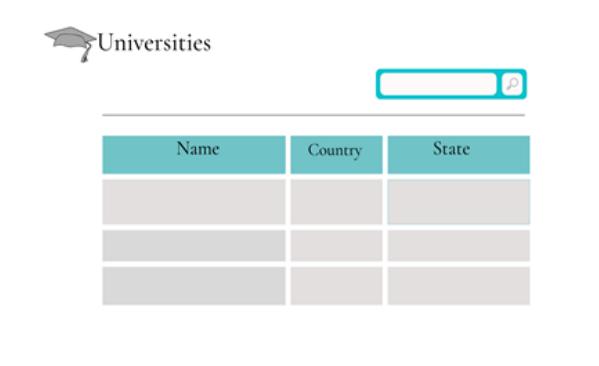


```
[{"id": 1, "name": "Marywood University", "country": "United States", "state": null, "domains": ["marywood.edu"], "web_pages": ["https://www.marywood.edu"]}, {"id": 2, "name": "University of Petroleum and Energy Studies", "country": "India", "state": "Dehradun", "domains": ["upes.ac.in"], "web_pages": ["https://www.upes.ac.in"]}, {"id": 3, "name": "Cégep de Saint-Hyacinthe", "country": "Canada", "state": null, "domains": null, "web_pages": null}], [{"text": "Universities"}]
```

Por otro lado, se presenta el diseño en maquetación de la aplicación a realizarse, como se observa la Figura 25. Se propone una interfaz para que el usuario ingrese sus credenciales y pueda acceder al sistema. En este caso se recalca que la interfaz debe poseer un buen manejo del espacio, haciendo referencia a la acción que se va a ejecutar en dicha pantalla.

Figura 25

Maquetación Pantalla, tabla de datos



Una vez que el usuario haya ingresado correctamente sus credenciales, se le presenta la página principal de la aplicación. En esta pueden existir varias opciones que faciliten su navegabilidad, en

este caso se debe tomar en cuenta el manejo de títulos para mantener a la persona informada de lo que se muestra en pantalla. Una buena práctica es hacer el uso de colores sobrios que no distraigan al usuario de la finalidad de la aplicación.

vi. Herramientas y Entornos de Desarrollo:

Es posible que cuando realice este ejercicio, las versiones de las aplicaciones de software que instale se hayan actualizado, sin embargo, este hands-on lab considera que el software y sus versiones son compatibles y estarán estables por un período de tiempo razonable. Se le aconseja al lector, analizar las versiones más actuales. Además, el estudiante debe estar consciente que la instalación dependerá del Sistema Operativo de su elección, Windows, MacOS, Linux. Para el ejercicio es necesario el editor de código *Visual Studio Code*, en caso de no haberlo instalado, se lo puede descargar del siguiente enlace <https://code.visualstudio.com/download>

6.1 Descarga de Node.js

- a. Acceder a la página oficial de descarga, se selecciona la versión compatible con el sistema operativo pertinente e inmediatamente empieza la descarga: <https://Node.js.org/es/download/>
- b. Abrir el ejecutable y seleccionar continuar

Figura 26

Ventana de instalación de Node.js

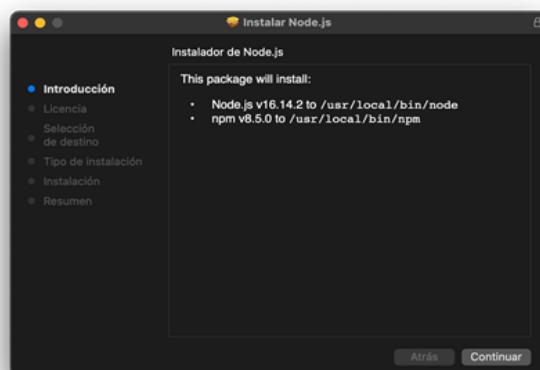


Figura 27

Página oficial para descargar Node.js



- c. Aceptar los términos de licencia y continuar
- d. Escoger la opción instalar y empieza la instalación

Figura 28

Ventana de instalación de Node.js, "aceptar" licencia

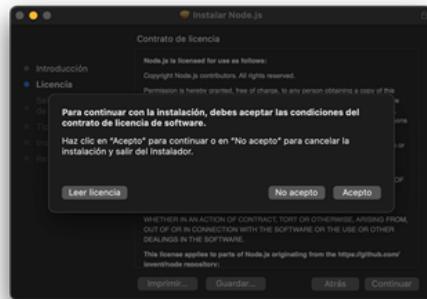


Figura 29
Ventana de instalación de Node.js, "instalar"



- e. Para verificar que la instalación se realizó con éxito, se abre una consola de comandos y se ejecuta el siguiente comando: **% node -v**

Figura 30
Consola de comandos, ejecución del comando "node -v"

```
camilavenegas ~ % node -v
v16.14.2
camilavenegas ~ %
```

Se mostrará la versión de Node.js que se ha instalado

6.2 Creación de un proyecto usando React

- a. Abrir una consola de comandos y la raíz en donde se va a guardar el proyecto. Primero se debe agregar React al equipo, para lo cual, se ejecuta el siguiente comando:

% npm install -g create-react-app

Nota. Es importante ejecutar estos comandos como administrador para que no existan fallos en la instalación.

- b. Una vez que ha finalizado la descarga de todas las bibliotecas creamos la aplicación con el comando

% npx create-react-app my-app

Donde my-app es el nombre del proyecto que se está creando.

Figura 31
Creación del proyecto React usando el comando, "npx create-react-app"

```

Last login: Sun Mar 29 18:16:34 on ttys005
camilavenegas@MacBook-Pro-de-Camila ~ %
camilavenegas@MacBook-Pro-de-Camila ~ % cd Documents
camilavenegas@MacBook-Pro-de-Camila Documents % cd Quinto_Semestre
camilavenegas@MacBook-Pro-de-Camila Quinto_Semestre % cd AwD
camilavenegas@MacBook-Pro-de-Camila AwD % npx create-react-app my-app
Creating a new React app in /Users/camilavenegas/Documents/Quinto_Semestre/AwD/my-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
(   ) r idealTree:my-app: 5111 idealTree buildDeps

```

c. Una vez se haya finalizado la creación se procede a revisar que todo haya salido correctamente

Figura 32
Creación de proyecto React finalizado en consola

```

Success! Created my-app at /Users/camilavenegas/Documents/Quinto_Semestre/AwD/my-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:
  cd my-app
  npm start

Happy hacking!
Camilavenegas@MacBook-Pro-de-Camila AwD %

```

6.3 carga del proyecto de React

a. Se ejecuta Visual Studio Code y se abre la carpeta del proyecto creado, se puede observar que se crearon automáticamente varias carpetas, así como un apartado de módulo que es lo que React.

Figura 33
Proyecto React abierto desde Visual Studio Code

```

my-app
  node_modules
  public
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md

```

```

2  import './App.css';
3
4  function App() {
5    return (
6      <div className="App">
7        <header className="App-header">
8          <img src={logo} className="App-logo" alt="logo" />
9          <p>
10            Edit <code>src/App.js</code> and save to reload.
11          </p>
12          <a
13            className="App-link"
14            href="https://reactjs.org"
15            target="_blank"
16            rel="noopener noreferrer"
17          >
18            Learn React
19          </a>
20        </header>
21      </div>
22    );
23  }
24
25  export default App;
26

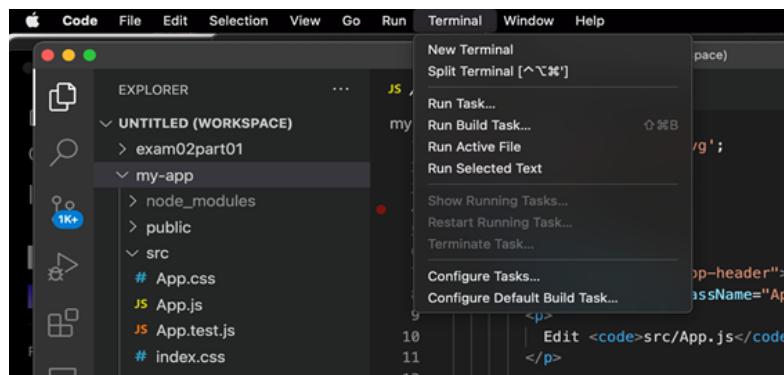
```

b. Agregar Axios al proyecto.

Se elige la opción terminal en la parte superior de Visual Studio Code

Figura 34
Nueva terminal en el editor Visual Studio Code

Figura 35
Opción "New Terminal" en Visual Studio Code



Se escoge la opción new terminal, o nueva terminal y selecciona la terminal del proyecto.

Inmediatamente en la parte inferior de visual aparecerá la sección de consola, donde se deberá ejecutar el comando siguiente

% npm install axios

Figura 36
Instalación de "axios" en el proyecto

```

17 | > Learn React
18 |
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
camilavenezas@MacBook-Pro-de-Camila my-app % npm install axios
added 1 package, and audited 1413 packages in 4s
169 packages are looking for funding
  run `npm fund` for details
6 moderate severity vulnerabilities

  To address all issues (including breaking changes), run:
    npm audit fix --force
  
```

Se procede de la misma manera a instalar la herramienta mui-material

% npm install @mui/material @emotion/react @emotion/styled

Figura 37
Instalación de "@mui/material"

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Run `npm audit` for details.
camilavenegas@MacBook-Pro-de-Camila my-app % npm install @mui/material @emotion/react @emotion/styled
added 10 packages, and audited 1447 packages in 5s
176 packages are looking for funding
  run `npm fund` for details
6 moderate severity vulnerabilities
```

The terminal window shows the command `npm install @mui/material @emotion/react @emotion/styled` being run. The output indicates 10 packages were added and 1447 packages were audited. A note says 176 packages are looking for funding and to run `npm fund`. It also mentions 6 moderate severity vulnerabilities.

En el caso de requerir otras herramientas aplicables en el entorno de React. Se debe usar el comando `% npm install` con los parámetros apropiados.

6.4 Ejecución del proyecto React

Para ejecutar el proyecto React se debe poner el siguiente comando:

`% npm start`

En el caso de que se tenga el puerto 3000 ocupado por otro proyecto, React pedirá ejecutar el proyecto en otro puerto. Luego, se abre la ventana en el navegador donde el proyecto se está ejecutando.

Figura 38

Uso del comando "npm start" para ejecutar el proyecto, en caso del puerto 3000 encontrarse ocupado se debe aceptar que el sistema se ejecute en otro puerto

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
? Something is already running on port 3000. Probably:
/usr/local/bin/node /Users/camilavenegas/Documents/Proyectos Github/Gym-System-Group-7/06-Code/Unit3/project_TheWellness/node_modules/react-scripts/scripts/start.js (pid 32324)
  in /Users/camilavenegas/Documents/Proyectos Github/Gym-System-Group-7/06-Code/Unit3/project_TheWellness
Would you like to run the app on another port instead? > (Y/n)
```

The terminal shows an error message indicating that port 3000 is already in use by another process. It provides the path to the script and asks if the user wants to run the app on another port instead, with options (Y/n).

Figura 39

Proyecto React corriendo en el navegador



vii. Desarrollo de la Aplicación

7.1 Consumo de datos de la API con AXIOS

Primero se debe entender que es AXIOS. Axios se considera una biblioteca que a modo de cliente permite hacer peticiones HTTP a un servidor, las tareas son realizadas por medio del objeto Promise() de JavaScript con Node.js

```
import axios from 'axios'

const baseUrl = 'http://universities.hipolabs.com/search?country'

export async function getCountry(nameCountry){
  try{
    const response = await axios({
      url: `${baseUrl}?${nameCountry}`,
      method: 'GET',
    })
    return response
  }catch(error){
    console.log(error)
  }
}
```

En la carpeta *components* se crea al archivo *FormCountry.js* en el cual se estructura el formulario con una sola entrada (input) para que se tome el dato del país y se muestren las universidades correspondientes en una tabla.

Se hace uso de los componentes de material ui, como Box y DataGrid que permiten estilizar de mejor manera la interfaz de usuario.

```
import React, {useState} from 'react';
import './App.css';
import './index.css';
import {Box, TextField, Button} from "@mui/material"

const FormCountry = (props) => {

  const {formCountry, setformCountry} = props

  const handleChange = (event) => {
    const { name, value } = event.target
    setformCountry({ ...formCountry, [name]: value})
  }

  const handleSubmit = (e) => {
    e.preventDefault()
  }

  return (
    <form>

      <Box
        sx={{
          width: '30%',
          height: '100%',
          marginLeft:'auto',
          marginRight:'auto',
          justifyContent: 'center',
          display: 'flex',
          alignItems: 'center',
          flexDirection: 'column',
          paddingLeft:'20px',
        }>
        <TextField
          value={formCountry}
          onChange={handleChange}
        />
        <Button
          type='submit'
          variant='contained'
          color='primary'
        >Search</Button>
      </Box>
    </form>
  )
}

export default FormCountry
```

```

        paddingRight:'20px',
        background: '#fff', /* fallback for old browsers */
        borderRadius: '15px',
        boxShadow: '1px 1px 20px #333'
    )}
>
<br/>

/* Country */
<TextField fullWidth
    name='country'
    id="country"
    value={formCountry.country}
    onChange={handleChange}
    placeholder="Country"
    label="Country"
/>
<br/>
</Box>

</form>

);
};

export default FormCountry;

```

Luego, Se debe crear un archivo *Table.js*, en cual se crea la tabla de datos, en este caso se toman los atributos correspondientes al JSON y a la clase modelada anteriormente: country, name, domain, code y state. Se los usa como etiquetas principales para mostrar los datos obtenidos de la API.

Se debe tomar en cuenta el uso de las etiquetas obtenidas de la biblioteca MUI/MATERIAL, en la cual se obtiene un formato ya diseñado para el layout de la aplicación.

```

import React, { useState } from 'react';
import {Box} from "@mui/material"
import { DataGrid } from '@mui/x-data-grid';
import './index.css'

const TableCountries = (props) => {

    const countries = props.countries

    console.log(countries)
    const columns = [
        { field: 'country', headerName: 'Country', width: 200 },
        { field: 'domains', headerName: 'Domain', width: 200 },
        { field: 'name', headerName: 'Name', width: 200 },
        { field: 'webPages', headerName: 'Web page', width: 200 },
        { field: 'alphaTwoCode', headerName: 'Code', width: 200 }
    ]

    return (
        <Box
            sx={{
                width: '70%',
                height: '82vh',
                marginLeft:'auto',
                marginRight:'auto',
                justifyContent: 'center',
                display: 'flex',
                alignItems: 'center',
                flexDirection: 'column',

```

```

        paddingLeft:'20px',
        paddingRight:'20px',
        background: '#fff',
        borderRadius: '15px',
        boxShadow: '1px 1px 20px #333'
    )}
>

<h1>Lista de Universidades</h1><br/>

<div style={{ height: 400, width: '100%' }}>
    <DataGrid
        rows=
        {
            countries.map(item => (
                {
                    id: Math.random() * (100000 - 1) + 1,
                    country: item.country,
                    domains: item.domains,
                    name: item.name,
                    webPages: item.web_pages,
                    alphaTwoCode: item.alpha_two_code
                }
            ))
        }
    }

    columns={columns}
    pageSize={10}
    rowsPerPageOptions={[10]}
    />
</div>
<Box>
);
}
export default TableCountries

```

Se debe crear el archivo `TableLayout.js`, en el que se unen los dos componentes creados anteriormente.

```

import Table from './Table'
import FormCountry from './FormCountry'
import {getCountry} from '../services/countryAxios'
import './index.css';
import {Box} from "@mui/material"
import React, {useEffect, useState} from 'react';

const TableLayout = () => {

    const [formCountry, setformCountry] = useState({
        country: ""
    })
    const [countries, setCountries] = useState([])
    console.log(formCountry.country)
    useEffect(() => {
        async function loadCountries() {
            const response = await getCountry(formCountry.country)

            if (response.status === 200) {
                setCountries(response.data)
            }
        }
        loadCountries()
    }, [formCountry])
}

```

```

return (
  <>
  <Box>
    <img src='../public/logo.png'></img>
    <br/><br/> <br/><br/>
    <FormCountry formCountry={formCountry} setformCountry={setformCountry} />
    <br/><br/>
    <Table countries={countries} />
  </Box>

  </>
)
}
export default TableLayout

```

Finalmente, se debe realizar el llamado del componente realizado en el archivo *App.js* que fue creado por defecto en el proyecto React.

```

import React from 'react'
import { BrowserRouter, Route, Routes } from 'react-router-dom';
import TableCountries from './components/TableLayout';

const App = () => {

  return (
    <BrowserRouter>

      <Routes>
        <Route path="/" element={<TableCountries />} />
      </Routes>
    </BrowserRouter>

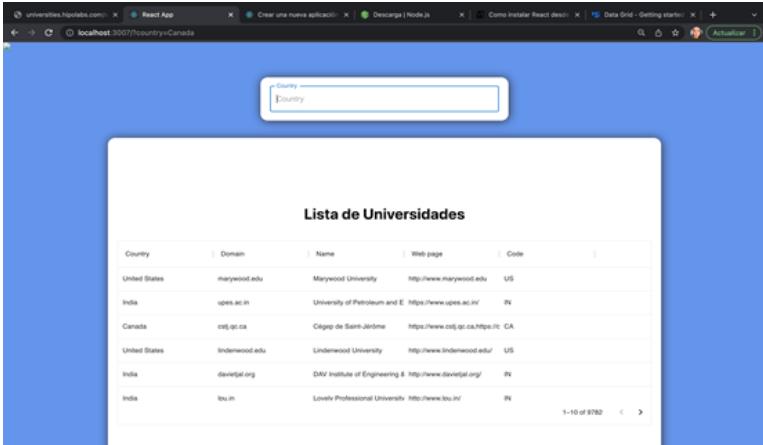
  )
}
export default App;

```

viii. Resultados de la aplicación

Figura 40

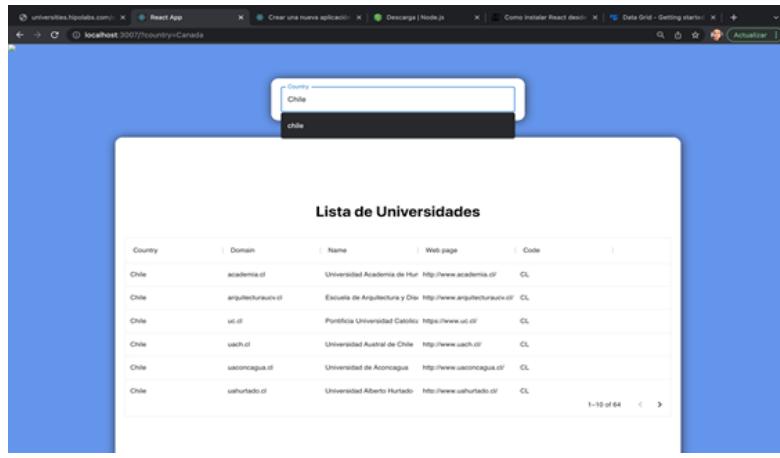
Aplicación Front-end, tabla de datos consumidos desde la API



Country	Domain	Name	Web page	Code
United States	marywood.edu	Marywood University	http://www.marywood.edu	US
India	upes.ac.in	University of Petroleum and Energy Studies	https://www.upes.ac.in/	IN
Canada	cstj.qc.ca	Cégep de Saint-Jérôme	https://www.cstj.qc.ca/https://cstj.ca	CA
United States	lindenwood.edu	Lindenwood University	http://www.lindenwood.edu/	US
India	davetjal.org	DAV Institute of Engineering & Technology	http://www.davetjal.org/	IN
India	bou.in	Loyola Professional University	http://www.bou.in/	IN

Figura 41

Obtención de datos según el país



A continuación se presentan los enlaces del repositorio en GitHub donde se puede encontrar el código de este taller, del video tutorial en el que se explica paso a paso el ejercicio, y de la URI que produce los datos a ser consumidos en el cliente desarrollado usando React:

- GitHub: <https://github.com/elascano/ESPE-DESARROLLO-WEB-2022>
- Video Tutorial: <https://www.youtube.com/watch?v=TzAfN8qlCtg>
- API consumida: <http://universities.hipolabs.com/search?country>

Nota. Los enlaces de descarga se encuentran presentes en cada apartado, según el programa. Puede ser probable, que en determinado punto en el tiempo la API aquí utilizada deje de existir, como ha sucedido con muchas APIs públicas en el transcurso del tiempo, el lector debería estar en capacidad de desarrollar el mismo ejercicio utilizando una API diferente, siempre y cuando esa API sea tipo REST y el formato de datos utilizado sea JSON.

Recursos complementarios

How to Make REST API Calls in React - GET, POST, ...



<https://www.youtube.com/watch?v=qXvFaEkkZH8>

How to Make REST API Calls in React - GET, POST, PUT, DELETE - Step by Step



Bibliografía



10 Best Tools For Front-End Web Development. (2021, marzo 26). GeeksforGeeks.
<https://www.geeksforgeeks.org/10-best-tools-for-front-end-web-development/>

Bojinov, V. (2018). RESTful Web API Design with Node.js 10, Third Edition: Learn to create robust RESTful web services with Node.js, MongoDB, and Express.js, 3rd Edition (3rd edition). Packt Publishing.

Brborich, W., Oscullo, B., Lascano, J. E., & Clyde, S. (2020). An Observational Study on the Maintainability Characteristics of the Procedural and Object-Oriented Programming Paradigms. 2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE T), 1-10.
<https://doi.org/10.1109/CSEET49119.2020.9206213>

Eck, D. J. (2016). Introduction to programming Using Java (Seventh). <http://math.hws.edu/javanotes/>

Freeman, E., Bates, B., Sierra, K., & Robson, E. (2004). Head First Design Patterns: A Brain-Friendly Guide (1st edition). O'Reilly Media.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., & Booch, G. (1994). Design Patterns: Elements of Reusable Object-Oriented Software (1 edition). Addison-Wesley Professional.

Ganatra, S. (2018). React Router Quick Start Guide: Routing in React applications made easy. Packt Publishing.

Herron, D. (2018). Node.js Web Development: Server-side development with Node 10 made easy, 4th Edition (4th Revised edition). Packt Publishing.

10 Best Tools For Front-End Web Development. (2021, marzo 26). GeeksforGeeks. <https://www.geeksforgeeks.org/10-best-tools-for-front-end-web-development/>

Bojinov, V. (2018). RESTful Web API Design with Node.js 10, Third Edition: Learn to create robust RESTful web services with Node.js, MongoDB, and Express.js, 3rd Edition (3rd edition). Packt Publishing.

Brborich, W., Oscullo, B., Lascano, J. E., & Clyde, S. (2020). An Observational Study on the Maintainability Characteristics of the Procedural and Object-Oriented Programming Paradigms. 2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE T), 1-10. <https://doi.org/10.1109/CSEET49119.2020.9206213>

Eck, D. J. (2016). Introduction to programming Using Java (Seventh). <http://math.hws.edu/javanotes/>

Freeman, E., Bates, B., Sierra, K., & Robson, E. (2004). Head First Design Patterns: A Brain-Friendly Guide (1st edition). O'Reilly Media.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., & Booch, G. (1994). Design Patterns: Elements of Reusable Object-Oriented Software (1 edition). Addison-Wesley Professional.

Ganatra, S. (2018). React Router Quick Start Guide: Routing in React applications made easy. Packt Publishing.

Herron, D. (2018). Node.js Web Development: Server-side development with Node 10 made easy, 4th Edition (4th Revised edition). Packt Publishing.

Krasner, G., & Pope, S. (1988). A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. Journal of Object-oriented Programming - JOOP, 1.

Martin, M. (2020, noviembre 1). 20 Best Front End Web Development Tools & Software in 2022. <https://www.guru99.com/front-end-web-development-tools.html>

Raskin, J. (2000). The humane interface: New directions for designing interactive systems. ACM Press/Addison-Wesley Publishing Co.

The State of JS 2021: Front-end Frameworks. (s. f.). Recuperado 21 de abril de 2022, de <https://2021.stateofjs.com/en-US/libraries/front-end-frameworks/>

Tidwell, J., Brewer, C., & Valencia, A. (2020). Designing Interfaces: Patterns for Effective Interaction Design (3rd edition). O'Reilly Media.

Vincijanovic, D. (2018, agosto 30). Level up your React architecture with MVVM. Medium. <https://medium.cobeisfresh.com/level-up-your-react-mvvm-1033a2a2a2d>

[architecture-with-mvvm-a471979e3f21](#)

Webflow: Create a custom website | No-code website builder. (s. f.).
Recuperado 21 de abril de 2022, de <https://webflow.com>

Autoevaluación

1. MVC (Modelo-Vista-Controlador) es un

- Patrón de Diseño.
- Modelo.
- Herramienta de Desarrollo.

Pregunta 1 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

2. React es

- Una biblioteca Back-End.
- Una biblioteca Front-End.
- Un Framework de Front-End.

Pregunta 2 de 10

[Atras](#)[Siguiente](#)[Enviar todo](#)

Autoevaluación

3. ¿Cuál de los siguientes no es un paradigma de programación?

- Python.
- Orientación a Objetos.
- Funcional.

Pregunta 3 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

4. Al hablar de MVC y decir que es el encargado de seleccionar la estrategia correcta para procesar y mostrar los datos, la estrategia se refiere a.

- Model.
- View.
- Controller.

Pregunta 4 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

5. Las tecnologías para desarrollo de Front-End sin las cuales no se podría desarrollar interfaces de usuario con

- JavaScript, CSS y React
- HTML, JavaScript y React
- HTML, JavaScript y CSS

Pregunta 5 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

6. De los patrones de Comportamiento del Usuario, ¿Cuál de los siguientes se refiere al hecho que el usuario consiga algo inmediatamente y no luego de un tiempo

- Instant Gratification
- Satisficing
- Deffered Choices

Pregunta 6 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

7. Al diseñar la navegabilidad entre páginas web, cuál de las siguientes preguntas no se debe hacer

- ¿Cómo llego allá?
- ¿Qué datos debe ingresar?
- ¿Dónde estoy ahora?

Pregunta 7 de 10

[Atras](#)[Siguiente](#)[Enviar todo](#)

Autoevaluación

8. ¿Cuál es el patrón de modelo navegacional que presenta un número bajo de puntos de entrada?

- Clear Entry Points
- Menu Page
- Escape Hatch

Pregunta 8 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

9. Las herramientas para desarrollo de interfaces de usuario que permiten el diseño en tiempo real manipulando los componentes gráficos directamente en lugar de usar código se conocen como:

- Frameworks
- Colecciones de templates
- WYSIWYG

Pregunta 9 de 10

Atras

Siguiente

Enviar todo



Desarrollo de Aplicaciones Web

Tema 4 Consumo de APIs y el Futuro del Desarrollo Web



Carrera en Línea
Tecnologías de la Información



< 1/28 >



4.1. Usabilidad Web



Si un sitio web tiene metas de usabilidad y de presentación de información y de opciones que se muestren de manera clara y concisa sin ambigüedad y la ubicación correcta de los elementos de tal forma que, no importe el dispositivo que se use para visitar el sitio, entonces se dice que el sitio web cumple con usabilidad (Brandon, 2014).

4.1. Usabilidad Web



Si un sitio web tiene metas de usabilidad y de presentación de información y de opciones que se muestren de manera clara y concisa sin ambigüedad y la ubicación correcta de los elementos de tal forma que, no importe el dispositivo que se use para visitar el sitio, entonces se dice que el sitio web cumple con usabilidad (Brandon, 2014).

4.1.1. Adaptive design

Cuando se habla de interfaces de usuario gráficas (GUI), se hace referencia a un diseño adaptativo, si el diseño se adapta a diferentes tamaños de pantallas, por ejemplo, computadoras de escritorio, tablets, smart phones, la consola de un vehículo, un smart TVs, inclusive, un Smart watch. Este concepto lo toma prestado el mundo del diseño de páginas web, y se lo aplica utilizando diferentes layouts dependiendo del tamaño del browser utilizado. Cuando el sistema web detecta un determinado dispositivo, por ejemplo un Smart phone, este selecciona el layout correspondiente para teléfonos inteligentes.

El término “Adaptive Web Design” fue introducido en 2011 por el diseñador web Aaron Gustafson en su libro *Adaptive Web Design: Crafting Rich Experiences With Progressive Enhancement* (*Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement*: Aaron Gustafson, Jeffrey Zeldman: 9780983589501: Books, n.d.). Si se va a implementar un diseño adaptativo para un sitio web, se debe considerar los seis tamaños de ancho de pantalla más comunes: 320, 480, 790, 960, 1200 y 1600 pixeles. A pesar de que los primeros tamaños de pantalla se podrían considerar no apropiados para una computadora moderna, en la actualidad, los usuarios acceden a los sitios web desde diferentes dispositivos y no únicamente desde computadoras, razón por la cual una de las dimensiones más usadas durante mucho por las computadoras no se la considera, esta es 640 pixeles.

La mayor ventaja del adaptive design es que las páginas se diseñan para verse optimizadas en los diferentes dispositivos, la mayor desventaja es que su implementación es costosa, pues efectivamente el diseñador debe implementar seis versiones del sitio web siguiendo seis layouts de tamaño fijo. Se puede ayudar del uso de css y JavaScript, sin embargo, los seis layouts deben ser aplicados a cada página del sitio. Otra desventaja es que el usuario podría no contar con un dispositivo de despliegue que mantenga los tamaños estándares.

4.1.1. Adaptive design

Cuando se habla de interfaces de usuario gráficas (GUI), se hace referencia a un diseño adaptativo, si el diseño se adapta a diferentes tamaños de pantallas, por ejemplo, computadoras de escritorio, tablets, smart phones, la consola de un vehículo, un smart TVs, inclusive, un Smart watch. Este concepto lo toma prestado el mundo del diseño de páginas web, y se lo aplica utilizando diferentes layouts dependiendo del tamaño del browser utilizado. Cuando el sistema web detecta un determinado dispositivo, por ejemplo un Smart phone, este selecciona el layout correspondiente para teléfonos inteligentes.

El término “Adaptive Web Design” fue introducido en 2011 por el diseñador web Aaron Gustafson en su libro *Adaptive Web Design: Crafting Rich Experiences With Progressive Enhancement* (*Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement*: Aaron Gustafson, Jeffrey Zeldman: 9780983589501: Books, n.d.). Si se va a implementar un diseño adaptativo para un sitio web, se debe considerar los seis tamaños de ancho de pantalla más comunes: 320, 480, 790, 960, 1200 y 1600 pixeles. A pesar de que los primeros tamaños de pantalla se podrían considerar no apropiados para una computadora moderna, en la actualidad, los usuarios acceden a los sitios web desde diferentes dispositivos y no únicamente desde computadoras, razón por la cual una de las dimensiones más usadas durante mucho por las computadoras no se la considera, esta es 640 pixeles.

La mayor ventaja del adaptive design es que las páginas se diseñan para verse optimizadas en los diferentes dispositivos, la mayor desventaja es que su implementación es costosa, pues efectivamente el diseñador debe implementar seis versiones del sitio web siguiendo seis layouts de tamaño fijo. Se puede ayudar del uso de css y JavaScript, sin embargo, los seis layouts deben ser aplicados a cada página del sitio. Otra desventaja es que el usuario podría no contar con un dispositivo de despliegue que mantenga los tamaños estándares.

4.1.2. Responsive design

Por otro lado, el diseño responsivo sigue un concepto de diseño dinámico, donde los componentes se pueden reacomodar en la página web. Como se mencionó anteriormente, los diseños de las páginas web deben adaptarse desde los monitores gigantes de la industria hasta los relojes inteligentes. Este es el reto para del diseñador web moderno, ya que la información es presentada en cualquier dispositivo que soporte un browser. Tomando las palabras que dijo Bruce Lee:

"Be Water, My Friend.
Empty your mind.
Be formless, shapeless, like water.
You put water into a cup, it becomes the cup.
You put water into a bottle, it becomes the bottle.
You put it into a teapot, it becomes the teapot.
Now water can flow or it can crash.
Be water, my friend."

Bruce Lee

Adaptando las palabras anteriores al diseño de páginas web, se interpreta que la página en cuestión debe adaptarse al dispositivo desde la cual es abierta, y no al revés, tampoco debemos hacer que el usuario se adapte a nuestro diseño, sino que nuestro diseño debe adaptarse al usuario.

El término "Responsive Design" fue acuñado por Ethan Marcotte en su libro *Responsive Web Design* (*Responsive Web Design: Brief Books for People Who Make Websites*, No. 4): Ethan Marcotte: 9781937557188: Books: Amazon.Com, n.d.). Los componentes de una página web deben acomodarse en el espacio disponible en el navegador web. Aun, si el usuario modifica el tamaño del browser en un sistema operativo de escritorio, el contenido debería reordenarse de acuerdo al tamaño seleccionado por el usuario. Varios sitios que han implementado este concepto son Amazon, USA Today, Apple, About.com.

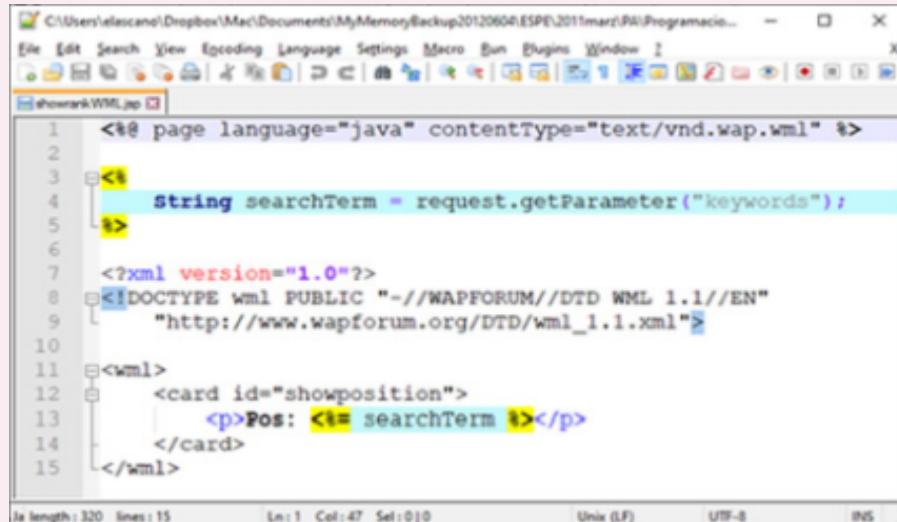
La implementación de un diseño responsivo a diferencia de un diseño adaptativo radica en la implementación de un único layout en el diseño responsivo, por lo tanto en una implementación más efectiva y de bajo costo. A pesar de permitir menor control del tamaño de cada pantalla, es el método más preferido en la actualidad. Inclusive existen varios templates disponibles en los distintos frameworks o bibliotecas front-end, en especial en la mayoría de CMSs como WordPress, Joomla. En conclusión, el diseñador implementa un solo diseño que se adapta a todas las pantallas.

4.1.3. Mobile web applications

WML fue durante varios años el lenguaje de marcado para los celulares inalámbricos que usaban OpenWave o browsers similares. En el back-end el procesamiento de las aplicaciones web se podía realizar con las tecnologías usuales como JSP (Figura 2) o PHP.

Figura 2.

Código JSP que incluye WML en su programación de Front End



The screenshot shows a code editor window with the file name "showrankWML.jsp". The code is a Java JSP page that generates WML. It includes JSP scriptlets to get a search term from the request, an XML declaration, and a WML document structure with a single card containing a search result. The code editor interface includes tabs for file, edit, search, view, encoding, language settings, macro, run, plugins, and window. Status bars at the bottom show file length (320), lines (15), and other details like Unix (LF) and UTF-8 encoding.

```
<%@ page language="java" contentType="text/vnd.wap.wml" %>
<%
String searchTerm = request.getParameter("keywords");
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="showposition">
<p>Pos: <%= searchTerm %></p>
</card>
</wml>
```

En la actualidad (2022), los teléfonos celulares simples han sido reemplazados por los Smart phones (teléfonos inteligentes), que por sus capacidades y recursos físicos, incluso podrían compararse en poder de computación a una computadora personal. El único limitante, se podría mencionar, es el tamaño visual de la pantalla, porque en resolución se compara a varios monitores. Adicionalmente el ancho de banda de las redes Wireless ya no es tan precario y permite compartir (download / upload) archivos y datos de tamaño considerables.

4.1.3. Mobile web applications

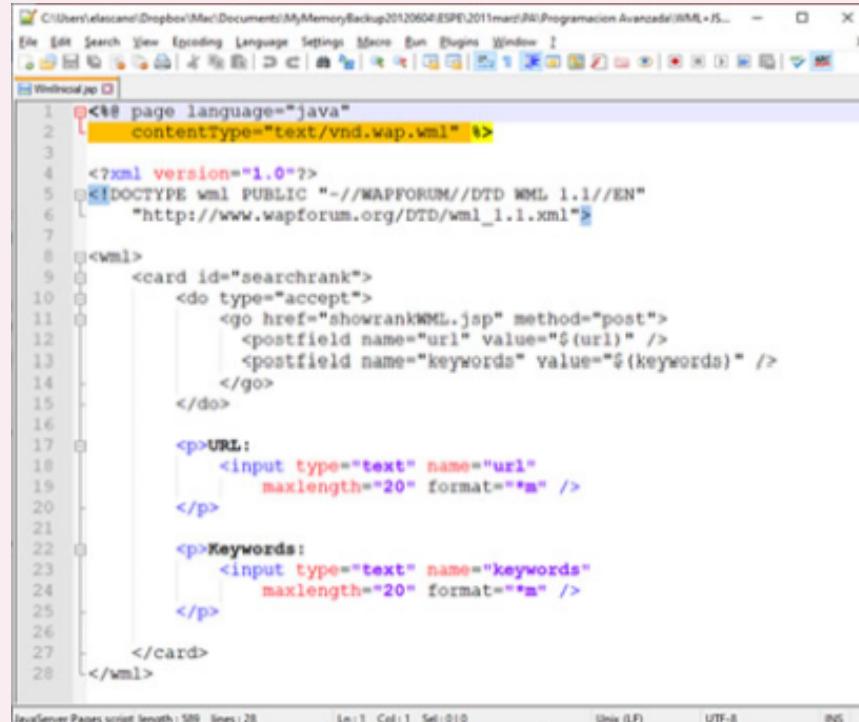
Por lo tanto, ya no hace falta un lenguaje como WML que permitía compartir información en redes de ancho de banda restringido. Las aplicaciones web móviles modernas se desarrollan usando HTML, CSS y JavaScript y sus respectivos frameworks y librerías, tal como si se tratara de una aplicación web para PC. Más aun, por ejemplo se tiene el framework React Native, que permite implementar aplicaciones nativas para teléfonos inteligentes usando JavaScript, HTML, CSS, con la correspondiente ventaja que a diferencia de una aplicación web móvil, una aplicación React Native tiene acceso a los recursos del dispositivo móvil. React Native, además, puede ser usado para diferentes plataformas como: Android, Android TV, iOS, MacOS, tvOS, Web, Windows, UWP (Universal Windows Platform).

En conclusión, se puede desarrollar aplicaciones Web para móviles de la misma manera que las aplicaciones web para PCs, pero se debe considerar el aspecto visual, para ello se debe pensar en soluciones PWA y en Adaptive Web Design.

4.1.3. Mobile web applications

Figura 1

Página WML



```

C:\Users\elascane\Dropbox\Mac\Documents\MyMemoryBackup20120604\ESPE\2011marzo\II\Programación Avanzada\WML+JS...
File Edit Search View Encoding Language Settings Macro Run Scripts Window
WmlSearch.j2e WmlSearch.j2e
1 <# page language="java"
2   contentType="text/vnd.wap.wml" #>
3
4 <?xml version="1.0"?>
5 <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
6   "http://www.wapforum.org/DTD/wml_1.1.xml">
7
8 <wml>
9   <card id="searchrank">
10     <do type="accept">
11       <go href="showrankWML.jsp" method="post">
12         <postfield name="url" value="${url}" />
13         <postfield name="keywords" value="${keywords}" />
14       </go>
15     </do>
16
17   <p><URL:>
18     <input type="text" name="url"
19       maxlength="20" format="*m" />
20   </p>
21
22   <p><Keywords:>
23     <input type="text" name="keywords"
24       maxlength="20" format="*m" />
25   </p>
26
27   </card>
28 </wml>

```

JavaServer Pages script length : 589 lines : 28 Line: 1 Col: 1 Sel: 0|0 Unix (LF) UTF-8 INS

El desarrollo de aplicaciones móviles para la Web durante sus inicios se encontraba restringido al tamaño pequeño de las pantallas y la limitación de los recursos de los celulares, que en comparación con las computadoras personales no permitían ejecutar aplicaciones de todo tipo en un teléfono. Su función principal era las llamadas telefónicas, los SMS, y ciertos juegos para distracción de sus usuarios, como por ejemplo el juego snake.

Con el aparecimiento de los primeros browsers para celulares, nace la necesidad de páginas web para celulares. Los lenguajes HDML y WML (Figura 1) se utilizaron al inicio para desarrollar páginas web que se ejecutaban sobre redes de ancho de banda bajo como fue la red celular a finales de los 90s e inicios de los 2000s.

4.2. Bibliotecas y Frameworks de Front-End para consumo de servicios web.

El desarrollo de front end para web desde sus inicios se lo desarrolló utilizando HTML, más adelante aparecen CSS y JavaScript para complementar a HTML en el desarrollo de Front-End. Hasta la actualidad, y desde hace 25 años, el front-end web se ha desarrollado usando estas tres tecnologías. Pero el desarrollo utilizando las tecnologías (lenguajes) mencionadas, a pesar de ser suficiente para la Interface de usuario, su uso tiene falencias en el tiempo de desarrollo, por ende en el rendimiento de los desarrolladores. Por este motivo, han surgido varias herramientas como Templates, Librerías y Frameworks.



4.2.1. Frameworks/bibliotecas front-End para consumo de servicios web.



Las herramientas más usadas en la actualidad de acuerdo a 2021 JavaScript Rising Stars report que se basa en las estrellas a signadas en GitHub son: React (18.5 K), Vue.js (14.3K), Svelte (13.6 K), Angular (9.3K). En popularidad (2022 Stats on Top JS Frameworks, n.d.). Por lo tanto, React es el Framework (biblioteca) más popular en el mundo de los front end, pues cuenta con 15.7 millones de descargas, reportado por (Angular vs React vs Vue | Npm Trends, n.d.). De aquí en adelante, React, se referirá a la biblioteca para front end Web, pero indistintamente se lo hará referencia como Framework.

React es usado para desarrollar Responsive User Interfaces, sus mayores ventajas son su facilidad de depuración (debugging) y la gran comunidad de desarrolladores que la soportan. React, además presenta un nivel de satisfacción alto para los programadores (70%) y los empleadores actuales ven en React como una destreza requerida para contratar a su personal. Contrario a Angular, uno de los primeros frameworks que los desarrolladores dicen que no lo volverían a usar. Los desarrolladores, colocan a Vue como uno de los framework favoritos, después de React.

Vue es un framework ligero que permite construir aplicaciones altamente dinámicas, tales como las single-page applications, también es considerado fácil de usar. Vue se considera la solución ideal para pequeños proyectos. Además, basado en Google search trends, se puede decir que Vue es el Framework JS más usado en países como en China, U.K, Francia, Bélgica. Sin embargo, Vue no tiene un mercado tan fuerte como Angular.

Angular, a pesar de estar en declive los últimos años, sigue siendo la opción más robusta para el desarrollo de aplicaciones a gran escala. Y mantiene un número estable de desarrolladores y aplicaciones que lo usan. Más adelante, en el hands-on lab de esta sección de presenta un ejercicio utilizando React.

4.2.2. Front-End + REST + BDD

El desarrollo moderno de aplicaciones Web exige el uso de la arquitectura n-capas, con el uso de por lo menos tres capas. La regla de negocio ya no es parte del código de la aplicación en sí. El código que procesa la información se encuentra en una API, y detrás de esta está el código de acceso a los datos y posiblemente APIs de terceros. La base de datos está en una maquina diferente, y muchas veces, los datos se encuentran en la Nube de Computación. En las siguientes posibles arquitecturas para aplicaciones Web, el Cliente o Front End puede ser una aplicación web cualquiera o inclusive una aplicación de escritorio o una móvil.

- Client ↔ REST ↔ EJB ↔ DB (EIS)
- Client ↔ SOAP ↔ EJB ↔ DB (EIS)
- Client ↔ REST ↔ JPA ↔ DB (EIS)
- Client ↔ REST ↔ DB (EIS)
- Client ↔ REST ↔ .NET ↔ ORM ↔ DB(EIS)
- Client ↔ SOAP ↔ .NET ↔ ORM ↔ DB(EIS)



4.3. El futuro del desarrollo web

Luego de su creación en 1989 por su autor Tim Berners-Lee mientras trabajaba en la Organización Europea para la Investigación Nuclear -CERN (European Organization for Nuclear Research) y de su posterior introducción el público en 1991, han pasado más de tres décadas. Sus bases: el HTTP, el HTML, el browser y el servidor web han evolucionado, pero se mantienen como el corazón del World Wide Web. Tim Berners-Lee funda el World Wide Consortium (W3C), quienes crean XML y en 1996 proponen la adopción de una HTML más estricto, el XHTML, al mismo tiempo da inicio la revolución de la Web 2.0, la Web colaborativa. Los tres grandes de la época, Mozilla, Opera y Apple rechazan el uso de XHTML y crean la organización WHATWG (Web Hypertext Application Technology Working Group) y crean HTML5. En 2009, la W3C abandona el proyecto XHTML y más adelante, en 2019, cede el control de la especificación HTML a WHATWG.

HTML, desde su primera versión de 18 etiquetas en Octubre de 1991 ha tomado muchas formas hasta su versión actual HTML5 que ha sido desarrollada desde enero de 2008. HTML5 como tal es un estándar en tiempo real que en Octubre de 2014 es lanzado como una recomendación estable del W3C. Una de las diferencias con las versiones anteriores de HTML, es su soporte para video y audio. HTML5 ofrece etiquetas como: <video>, <audio>, <canvas>, <header>, <footer>.

HTML5 tiene varios aspectos que lo hacen diferente de sus versiones anteriores. Por ejemplo, en lugar de usar cookies, HTML5 usa la base de datos conectada con la página actual para almacenar datos temporales. HTML5 es amigable con navegadores móviles. HTML5 soporta multimedia directamente, no necesita de plugins como Adobe Flash Reader. HTML5 permite que el browser corra directamente código JavaScript. Se puede obtener de manera fácil, la geolocalización del usuario. Dos de las mayores ventajas de HTML5 son: 1) su capacidad de usar una base de datos del lado del usuario como SQL en lugar de usar cookies, y 2) con HTML5 no es necesario escribir código para múltiples plataformas, por lo tanto HTML5 provee compatibilidad multi-plataforma.



4.3. El futuro del desarrollo web

Otra tecnología Web que ha evolucionado con el tiempo es JavaScript, un lenguaje de programación creado para desarrollo Front-End, en la actualidad se lo puede utilizar para desarrollar aplicaciones Back-End. El lenguaje en si es robusto, pero lo que más aporta al desarrollo Web es la creación de Frameworks y Bibliotecas basadas en JavaScript, tal es el caso de React (Front End), Angular (Front End), NodeJS (Back End). Desde la perspectiva del desarrollador, entonces el futuro está relacionado al uso de Frameworks o Librerías que permitan acelerar y mejorar el proceso de desarrollo.

Tecnológicamente hablando, hay muchos aspectos de la computación que afectan directamente al Web, y por consecuencia a sus usuarios. Por ejemplo, los datos biométricos de un usuario pueden integrarse a una aplicación y permitir realizar acciones de manera remota, por otro lado está la pérdida de la privacidad de cada individuo que se encuentra en una red social, por lo tanto la seguridad toma un papel primordial para mejorar esa pérdida de la privacidad. Una solución que se usa para implementar sistemas financieros seguros, se podría usar para proveer sistemas web más seguros, la ciber seguridad.



4.3. El futuro del desarrollo web

En conclusión, el desarrollo web sigue evolucionando constantemente, tanto desde el punto de vista de desarrollo (Bibliotecas y Frameworks) como de las tecnologías utilizadas (servidores web, servidores de aplicaciones, máquinas virtuales, dockers), tecnologías básicas (para el lado del cliente) como HTML, CSS y JavaScript han evolucionado durante estas tres décadas y seguirán evolucionando en el futuro. HTML actualmente se encuentra en su versión 5, CSS está en la especificación 4, JavaScript está en la 12th Edición ECMAScript 2021 y se proyecta a ES.Next. Java Script actualmente se utiliza también para desarrollar aplicaciones del lado del servidor. Las aplicaciones Web, que al inicio se publicaban en las máquinas de las empresas con conexión dedicada a Internet e instalando un servidor Web como httpd1.0, Apache, Apache Tomcat, Glassfish, Payara, etc., pasaron a publicarse en servicios de hosting. Estos proveedores, se encargan de ofrecer la infraestructura y la plataforma necesaria para publicar las aplicaciones Web. Actualmente, las aplicaciones web, las páginas web, las bases de datos, entre otras aplicaciones se instalan en máquinas virtuales publicadas en proveedores de servicios en la Nube de Computación.



4.3. El futuro del desarrollo web

Luego de su creación en 1989 por su autor Tim Berners-Lee mientras trabajaba en la Organización Europea para la Investigación Nuclear -CERN (European Organization for Nuclear Research) y de su posterior introducción el público en 1991, han pasado más de tres décadas. Sus bases: el HTTP, el HTML, el browser y el servidor web han evolucionado, pero se mantienen como el corazón del World Wide Web. Tim Berners-Lee funda el World Wide Consortium (W3C), quienes crean XML y en 1996 proponen la adopción de una HTML más estricto, el XHTML, al mismo tiempo da inicio la revolución de la Web 2.0, la Web colaborativa. Los tres grandes de la época, Mozilla, Opera y Apple rechazan el uso de XHTML y crean la organización WHATWG (Web Hypertext Application Technology Working Group) y crean HTML5. En 2009, la W3C abandona el proyecto XHTML y más adelante, en 2019, cede el control de la especificación HTML a WHATWG.

HTML, desde su primera versión de 18 etiquetas en Octubre de 1991 ha tomado muchas formas hasta su versión actual HTML5 que ha sido desarrollada desde enero de 2008. HTML5 como tal es un estándar en tiempo real que en Octubre de 2014 es lanzado como una recomendación estable del W3C. Una de las diferencias con las versiones anteriores de HTML, es su soporte para video y audio. HTML5 ofrece etiquetas como: <video>, <audio>, <canvas>, <header>, <footer>.

HTML5 tiene varios aspectos que lo hacen diferente de sus versiones anteriores. Por ejemplo, en lugar de usar cookies, HTML5 usa la base de datos conectada con la página actual para almacenar datos temporales. HTML5 es amigable con navegadores móviles. HTML5 soporta multimedia directamente, no necesita de plugins como Adobe Flash Reader. HTML5 permite que el browser corra directamente código JavaScript. Se puede obtener de manera fácil, la geolocalización del usuario. Dos de las mayores ventajas de HTML5 son: 1) su capacidad de usar una base de datos del lado del usuario como SQL en lugar de usar cookies, y 2) con HTML5 no es necesario escribir código para múltiples plataformas, por lo tanto HTML5 provee compatibilidad multi-plataforma.

4.3.1. XaaS (Everything as a service)

La era de la computación en la nube viene de la mano con la virtualización. Se puede tener varios sistemas computacionales, varios sistemas operativos ejecutándose en una misma computadora física, a estos equipos se los conoce como máquinas virtuales.

En la era del Web (90s en adelante), las empresas publican sus páginas en servidores web que se encuentran en sus instalaciones, más adelante, las páginas de las empresas pasan a ser aplicaciones web y se publican en las instalaciones de las empresas o en proveedores de hosting, de esta manera, la empresa no tiene la necesidad de administrar sus servidores físicos o de mantener una conexión activa en Internet 24x7. En el primer caso los costos de hardware, de mantenimiento, de administración los cubre la empresa que desea estar publica en el Web, en el segundo caso, le empresa paga una cuota mensual por tener sus aplicaciones activas en Internet.

La computación en la nube, si bien es un concepto que viene de hace varias décadas atrás, toma forma en las década de los 2010. Basada en el concepto de utility computing (servicio computacional público), ofrece sus servicios cobrando un costo con medidor, de esta forma el usuario paga de acuerdo al uso (tiempo o recurso usado), nace el concepto pay-per-use pay-as-you-go, la comercialización de los servicios es similar a un servicio público como la eléctrica, el agua potable, el servicio telefónico. Servicios como máquinas virtuales, bases de datos, discos duros virtuales, correo electrónico, repositorios de código, procesadores de palabras, en otras palabras, todo servicio o recurso se puede ofrecer como Servicio en la Nube.

4.3.1. XaaS (Everything as a service)

Los principales modelos de servicio en la Nube son IaaS (Infrastructure as a Service), PaaS (Platform as a Service) y SaaS (Software as a Service) (Al-Ghatrifi, 2015). IaaS es un servicio que ofrece recursos virtuales en la nube tales como Almacenamiento a manera de discos duros, memoria RAM, procesadores, computadoras + Sistemas Operativos, redes. Estos servicios pueden reemplazar por completo a un centro de datos de una empresa o a un servidor web local. IaaS es un modelo en el que se ofrece una o más máquinas virtuales al usuario, el cual debe proveer datos como Sistema Operativo (Linux, MacOS, MS Windows), cantidad de memoria RAM (GB), almacenamiento (GB/TB), procesadores virtuales (VPCs). Una vez contratado el servicio, el cliente debe instalar el software necesario orientado a sus necesidades. Un ejemplo de este servicio es AWS/EC2 Amazon Web Services Elastic Computing.

Platform as a Service es un modelo en el cual el proveedor ofrece al cliente, un conjunto formado por infraestructura virtual y software básico de desarrollo o para ejecución de programas. Por ejemplo, si el cliente desea desarrollar aplicaciones web PHP, podría contratar un servidor virtual con sistema operativo Linux y LAMP (Linux Apache MySQL Php/Perl/Python), el proveedor del servicio entrega la máquina virtual lista, configurada y con los servicios levantados para que el cliente dé inicio inmediatamente en su proceso de desarrollo o de publicación de su aplicación web. Si el cliente es un desarrollador Java, podría buscar una máquina virtual que tenga instalado servidores web tales como Glassfish, Apache Tomcat, Payara; un motor de bases de datos como PostgreSQL, MariaDB, MongoDB; y una herramienta de desarrollo como NetBeans, Visual Studio Code, Eclipse. En caso de ser necesario contratar PaaS para el cliente final que contrató la construcción de la aplicación web y desea poner en marcha su proyecto desarrollado en Java, únicamente se debería contratar el servicio que incluya un sistema operativo cualesquiera, el intérprete de Java (JRE) y un servidor Web como Payara. El cliente no administra nada más que su aplicación desarrollada. Un ejemplo típico es AWS MarketPlace

4.3.1. XaaS (Everything as a service)

SaaS es un modelo, en el cual el proveedor del servicio solicita y paga por el servicio sin importar la plataforma o el sistema operativo utilizado para proveer el servicio. En otras palabras, el cliente paga por el servicio y recibe el servicio, no máquinas virtuales o plataformas. Por ejemplo, si se necesita almacenamiento en la nube, se puede contratar DropBox, Google Drive, One Drive, en esos servicios, el cliente obtiene el servicio de almacenamiento remoto de archivos sin importarle el sistema operativo o la administración de los dispositivos de almacenamiento.

Al hablar de XaaS, se tiene que en la nube de computación se puede, prácticamente, ofrecer todo como servicio, por ejemplo (“XaaS,” 2019):

- DBaaS, DataBase as a Service, MongoDB Atlas
- SECaaS, Security as a Service, Oracle Cloud Access Security Broker (CASB)
- DaaS, Desktop as a Service, Horizon Cloud -VMware
- CaaS, Communication as a Service, VoIP
- AaaS, API as a Service, Google API Discovery Service
- AlaaS, Artificial Intelligence as a Service, IBM Watson
- CaaS, Content as a Service, WordPress
- ESaaS, Energy Storage as a Service, Constant Power
- MaaS, Malware as a Service, VMWare App Defense
- NaaS, Network as a Service, Cisco Digital Network Architecture
- SaaS (Storage as a Service) Azure Storage

4.3.1. XaaS (Everything as a service)

La era de la computación en la nube viene de la mano con la virtualización. Se puede tener varios sistemas computacionales, varios sistemas operativos ejecutándose en una misma computadora física, a estos equipos se los conoce como máquinas virtuales.

En la era del Web (90s en adelante), las empresas publican sus páginas en servidores web que se encuentran en sus instalaciones, más adelante, las páginas de las empresas pasan a ser aplicaciones web y se publican en las instalaciones de las empresas o en proveedores de hosting, de esta manera, la empresa no tiene la necesidad de administrar sus servidores físicos o de mantener una conexión activa en Internet 24x7. En el primer caso los costos de hardware, de mantenimiento, de administración los cubre la empresa que desea estar publica en el Web, en el segundo caso, le empresa paga una cuota mensual por tener sus aplicaciones activas en Internet.

La computación en la nube, si bien es un concepto que viene de hace varias décadas atrás, toma forma en las década de los 2010. Basada en el concepto de utility computing (servicio computacional público), ofrece sus servicios cobrando un costo con medidor, de esta forma el usuario paga de acuerdo al uso (tiempo o recurso usado), nace el concepto pay-per-use pay-as-you-go, la comercialización de los servicios es similar a un servicio público como la eléctrica, el agua potable, el servicio telefónico. Servicios como máquinas virtuales, bases de datos, discos duros virtuales, correo electrónico, repositorios de código, procesadores de palabras, en otras palabras, todo servicio o recurso se puede ofrecer como Servicio en la Nube.



4.3.2. Web 3.0



Luego del web participativo y social, Web 2.0 (1999...), el cual se centra en contenido generado de acuerdo al usuario, facilidad de uso, cultura participativa e interoperabilidad, surge la idea del Web 3.0 (2014...), basado en la tecnología blockchain, que incorpora conceptos como la economía descentralizada y basada en tokens. Esta idea gana interés, sin embargo, en 2021 con el surgimiento de las cripto-monedas. Sus principales características incluyen: seguridad de datos, escalabilidad, y privacidad para los usuarios y de cierta manera, combate la influencia de las grandes compañías tecnológicas. No obstante, en la actualidad, empezando desde 2004 se podría considerar que aun vivimos la era del Web 2.0

4.3.3. Inteligencia Artificial

El Web va evolucionando de páginas estáticas que muestra contenido informativo a páginas de comercio electrónico, a redes sociales e interactividad, a cualquier dispositivo, a pensar por sí misma.

Las últimas y las futuras versiones Web, llámese Web 4.0 o Web 5.0 hacen referencia a una Web de IoT, en la cual los dispositivos y la Web aprenden. La inteligencia artificial y el aprendizaje de maquina (machine learning). Si por ejemplo, hablamos de un edificio inteligente como por ejemplo el Edge de Ámsterdam (Web 4.0, 2018, p. 0), tiene una plataforma IP en la cual se pueden integrar diferentes edificios usando el web, nace el concepto de WebOS. Tenemos un web simbiótico, en el cual las maquinas se benefician de la interacción con el humano, y el humano se beneficia de los datos y servicios proporcionados por las máquinas. Inclusive, la interacción se la hace a través de la voz humana, por ejemplo Siri, Alexa. En este momento (2022), el IoT es una realidad y la inteligencia artificial esta implementada en muchos de los dispositivos de uso diario.



4.3.4. Minería de Datos



Otro aspecto importante en el Web es la minería de datos, a través de la cual, un programa busca información en la información para mejorar inversiones, decisiones, comercio, propaganda, salud. En el Internet, este proceso ya se lo realiza desde la creación de uno de sus primeros servicios, el correo electrónico. El correo electrónico entrega a través de sus usuarios una cantidad importante de información, la cual es minada con fines comerciales en la mayoría de casos, pero sus fines podrían ser varios, por ejemplo, militares, de salud, de educación. Lo cual no era posible al inicio del Web porque era estático o de procesamiento reducido a ventas en Internet, en la actualidad, y con la llegada de las redes sociales (Web 2.0) y la interactividad incrementada de los usuarios del Web y actualmente de las aplicaciones Web, se ha visto incrementada en cantidades astronómicas. Esta cantidad de datos permite realizar una minería de datos con diferentes propósitos. El nuevo repositorio de datos para la minería es el Web.

4.3.5. Otros Temas

Existen varios temas que se podrían considerar como el futuro del Web en cuanto al desarrollo y a las tecnologías web, al momento de escribir este contenido (2022), se podrían mencionar algunos que ya se detalló con anterioridad. Por ejemplo en la página de Global Media Insight una empresa de digital marketing con más de 21 años de experiencia GMI, se menciona en su artículo “Web Development Trends” (25 Latest Web Development Trends to Follow in 2022 | GMI, n.d.) de Abril de 2022, los siguientes 25 tópicos a considerar como el futuro del Web:

1. Blockchain Technology
2. Progressive Web Apps (PWA)
3. Internet of Things (IoT)
4. Accelerated Mobile Pages (AMP)
5. Voice Search Optimization
6. API-first Development
7. AI-Powered Chatbots
8. Push Notifications
9. Content Personalization with Machine Learning
10. Motion UI
11. Data Security
12. Multi-experience
13. Cybersecurity



4.3.5. Otros Temas

14. Micro Frontends
15. Virtual Reality
16. Serverless Architecture
17. Cloud Computing
18. Single-Page Applications (SPA)
19. JavaScript Frameworks
20. Automation Testing
21. Responsive Websites
22. Dark Mode
23. WebAssembly
24. No-code/Low code development
25. Augmented Reality

Para cuando el lector haya llegado a este punto, posiblemente algunos de esos temas seguramente, serán el pasado o el presente del Web, y nuevas tendencias en cuanto a tecnologías y a desarrollo se considerarán más importantes.



4.3.5. Otros Temas

En conclusión, el Web, y en general las áreas de tecnologías de la información y Ciencias de la Computación se encuentran en constante evolución, debido en gran medida a los avances electrónicos y afines. El experto en Tecnologías de la Información, en Ciencias de la Computación, en Ingeriría de Software, debe por lo tanto estar predisposto al cambio , a adaptarse y a aprender cada día de su profesión, en especial en el área del desarrollo web, y en general en todo aspecto tecnológico y científico.

A continuación, se presenta un taller (hands-on lab) que guiará al estudiante en el desarrollo de una aplicación web 3 capas.



4.3.5. Otros Temas

Existen varios temas que se podrían considerar como el futuro del Web en cuanto al desarrollo y a las tecnologías web, al momento de escribir este contenido (2022), se podrían mencionar algunos que ya se detalló con anterioridad. Por ejemplo en la página de Global Media Insight una empresa de digital marketing con más de 21 años de experiencia GMI, se menciona en su artículo “Web Development Trends” (25 Latest Web Development Trends to Follow in 2022 | GMI, n.d.) de Abril de 2022, los siguientes 25 tópicos a considerar como el futuro del Web:

1. Blockchain Technology
2. Progressive Web Apps (PWA)
3. Internet of Things (IoT)
4. Accelerated Mobile Pages (AMP)
5. Voice Search Optimization
6. API-first Development
7. AI-Powered Chatbots
8. Push Notifications
9. Content Personalization with Machine Learning
10. Motion UI
11. Data Security
12. Multi-experience
13. Cybersecurity

4.3.6. Hands-on lab Desarrollo de una Aplicación Web basada en API

El siguiente tutorial fue desarrollado por Steeven Vargas bajo la supervisión del docente autor Edison Lascano. El código ha sido publicado en un repositorio GitHub y puede ser libremente utilizado y replicado por el lector.

i. Tema:

Implementación de una interfaz web (Front End) responsiva, en el consumo de un servicio web RESTful (GET, PUT, POST, DELETE) con acceso a una base de datos en la nube usando React y otras librerías que permitan optimizar y mejorar la calidad del Web App.

ii. Objetivo:

- Desarrollar una aplicación web front end que permita el consumo de un servicio REST con acceso a una base de datos en la nube computacional aplicando principios de usabilidad en el manejo de las interfaces para varios dispositivos.
- Usar librerías para mejorar la implementación de la interfaz de usuario.
- Conocer cómo crear un cliente RESTful con acceso a una API previamente creada.

iii. Descripción:

La aplicación web a desarrollar consumirá un servicio previamente desarrollado en un el Hands-on lab de servicios REST. Se va a implementar un cliente REST que permita ejecutar servicios tipo GET, POST, PUT y DELETE, usando la Biblioteca React. Los datos obtenidos y cambios sobre ellos se encontrarán reflejados en una base de datos NOSQL en la nube -MongoDB Atlas. Los datos recuperados serán presentados en pantalla en una tabla, cada registro con sus respectivos botones de acciones (editar, eliminar, crear). Se prioriza la creación de una interfaz con colores representativos a la acción a realizar, también es prioridad el uso de un diseño responsivo que se adapte a browsers de diferentes dispositivos.

La aplicación web usara las siguientes tecnologías y conceptos:

El siguiente tutorial fue desarrollado por Steeven Vargas bajo la supervisión del docente autor Edison Lascano. El código ha sido publicado en un repositorio GitHub y puede ser libremente utilizado y replicado por el lector.

i. Tema:

Implementación de una interfaz web (Front End) responsiva, en el consumo de un servicio web RESTful (GET, PUT, POST, DELETE) con acceso a una base de datos en la nube usando React y otras librerías que permitan optimizar y mejorar la calidad del Web App.

ii. Objetivo:

- Desarrollar una aplicación web front end que permita el consumo de un servicio REST con acceso a una base de datos en la nube computacional aplicando principios de usabilidad en el manejo de las interfaces para varios dispositivos.
- Usar librerías para mejorar la implementación de la interfaz de usuario.
- Conocer cómo crear un cliente RESTful con acceso a una API previamente creada.

iii. Descripción:

La aplicación web a desarrollar consumirá un servicio previamente desarrollado en el Hands-on lab de servicios REST. Se va a implementar un cliente REST que permita ejecutar servicios tipo GET, POST, PUT y DELETE, usando la Biblioteca React. Los datos obtenidos y cambios sobre ellos se encontrarán reflejados en una base de datos NOSQL en la nube -MongoDB Atlas. Los datos recuperados serán presentados en pantalla en una tabla, cada registro con sus respectivos botones de acciones (editar, eliminar, crear). Se prioriza la creación de una interfaz con colores representativos a la acción a realizar, también es prioridad el uso de un diseño responsivo que se adapte a browsers de diferentes dispositivos.

La aplicación web usara las siguientes tecnologías y conceptos:

- *React*: Biblioteca JavaScript diseñada para facilitar el desarrollo de interfaces de usuario Web.
- *Node.js*: Entorno en tiempo de ejecución multiplataforma para la capa de servidor orientado a eventos.
- *Axios*: Biblioteca basada en promesas que se usa en navegadores y node.js
- *URI*: Cadena de caracteres que identifica los recursos de una red de forma única.
- *Material UI*: Biblioteca que provee de material design enfocado al diseño y visualización de páginas web.
- *Fontawesome*: Biblioteca que provee de íconos y fuentes basadas en CSS y LESS.

iv. Análisis

Se desarrollará una aplicación web que permita obtener y modificar los datos de clientes de una empresa X:

4.1. Requisitos Funcionales.

- a. La aplicación web deberá consumir un servicio REST que contenga los datos de los clientes.
- b. La aplicación web mostrará los datos de los clientes en una tabla.
- c. La aplicación web permitirá realizar las operaciones CRUD en la base de datos en la nube (MongoDB Atlas).

4.2. Requisitos No Funcionales.

- a. La aplicación web podrá adaptarse a la pantalla de cualquier dispositivo que posea un navegador Web.

v. Diseño

5.1. Diagrama de caso de uso

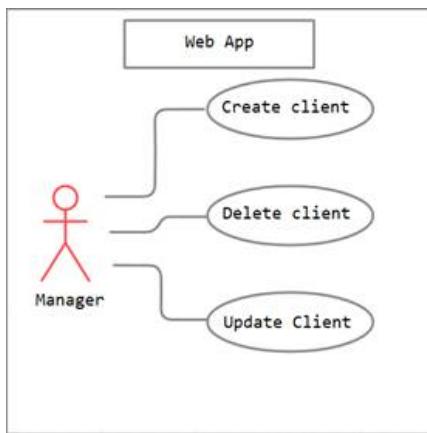
En la Figura 1 se muestra el diagrama de casos de uso que representa a la aplicación web a desarrollar donde tenemos:

- Un usuario de tipo “Manager” que realiza las siguientes acciones:
 - Crear un nuevo cliente (Create client)
 - Eliminar un cliente existente (Delete client)
 - Modificar un cliente existente (Update client)

En este caso se tendrá un solo tipo de usuario por el tamaño de la aplicación, pero se podrían agregar más tipos de usuarios con específicas acciones a realizar.

Figura 1

Diagrama de casos de uso

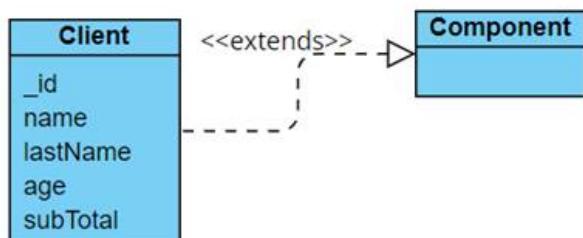


5.2. Diagrama de clases

En la Figura 2 se muestra el diagrama de clase que contará con métodos get, put, post, delete, select y modalInsert que permitirán manejar los atributos _id, name, lastName, age y subTotal consumidos del servicio REST.

Figura 2

Diagrama de clases



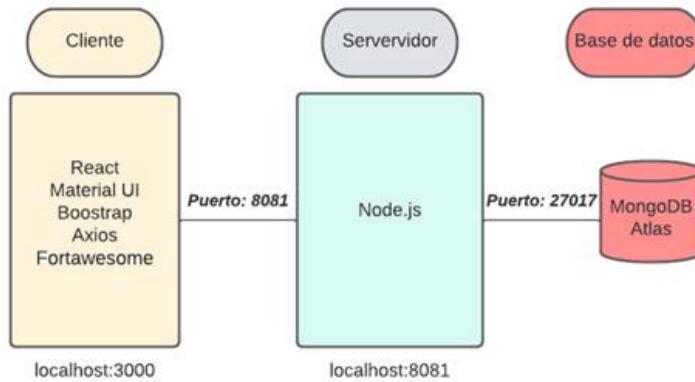
5.3. Arquitectura del sistema

En la Figura 3, se muestra la arquitectura que representa la aplicación web, se compone de: cliente desarrollado usando React, Servidor que provee servicios REST y servidor de

datos MongoDB Atlas.

Figura 3

Arquitectura



5.4. Diseño de la aplicación

Como se mencionó, se usará una aplicación back-end creada en talleres anteriores. La URI (<http://localhost:8081/clientes/clients>) permite recuperar la siguiente información.

Figura 4

Listado de clientes en formato JSON

La captura de pantalla muestra una lista de clientes en formato JSON. Los datos son los siguientes:

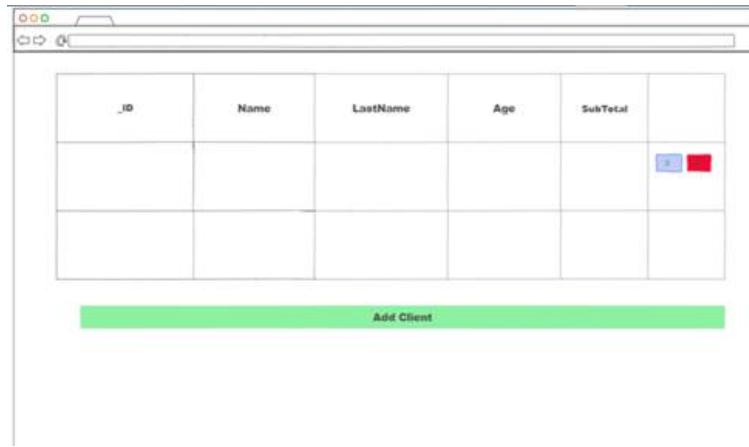
```
[{"_id": "623bee2388810c286e5c9f12", "name": "Andres", "lastName": "Vargas", "age": "21", "subTotal": "300", "__v": 0}, {"_id": "623bee8488810c286e5c9f15", "name": "Adrian", "lastName": "Burgos", "age": "23", "subTotal": "200", "__v": 0}, {"_id": "623dee288d5f68a03fe95ca6", "name": "Wilson", "lastName": "Iza", "age": "20", "subTotal": "211", "__v": 0}]
```

La información obtenida está en formato JSON, esto facilita el manejo de datos en el front end.

Al tener la información en un formato específico y conocer sus componentes se realiza un maquetado ubicando las claves Json que son: `_id`, `name`, `lastName`, `age` y `subtotal` en la cabecera de columna que le corresponde, y sus valores en los espacios designados, además se agrega una columna extra para los iconos de las acciones a realizar (modificar y eliminar) junto con un botón para crear un cliente en la parte inferior, el resultado se muestra se muestra en la Figura 5.

Figura 5

Maquetado



vi. Herramientas y entorno de desarrollo

Nota. Es posible que cuando usted realice este ejercicio, las versiones de las aplicaciones de software que instalen se hayan actualizado, sin embargo este hands-on lab considera que el software y sus versiones son compatibles y estarán estables por un período de tiempo razonable. Se le aconseja al lector, analizar las versiones más actuales. Además, el estudiante debe considerar que la instalación dependerá del Sistema Operativo de su elección, Windows, MacOS, Linux, etc.

Se recomienda usar Visual Studio Code para el desarrollo.

Además de la instalación previa de node.js y React en nuestro equipo los cuales fueron parte de talleres anteriores.

6.1. Visual Studio Code

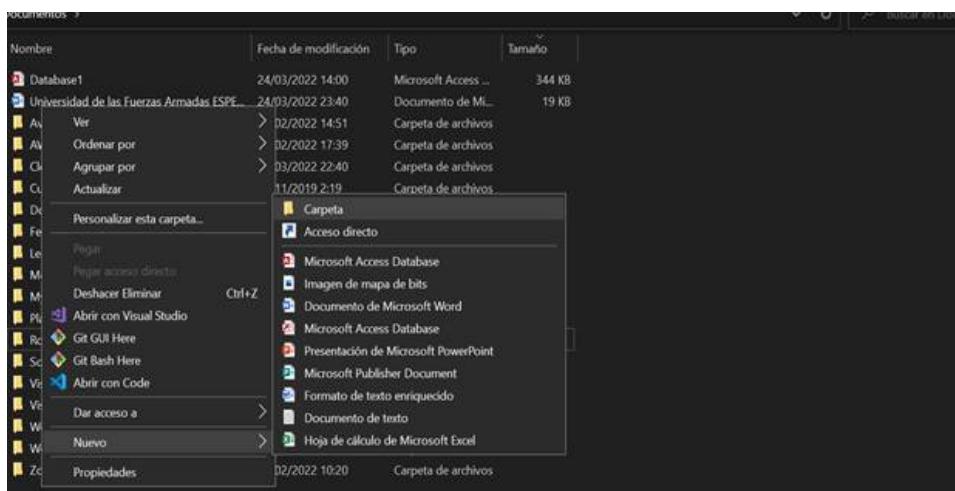
Link de descarga: <https://code.visualstudio.com/download>

6.2. Creación del proyecto React

6.2.1. Seleccionar el destino del proyecto o crearlo en una carpeta en el destino de preferencia personal, ver Figura 6.

Figura 6

Carpetas del proyecto React

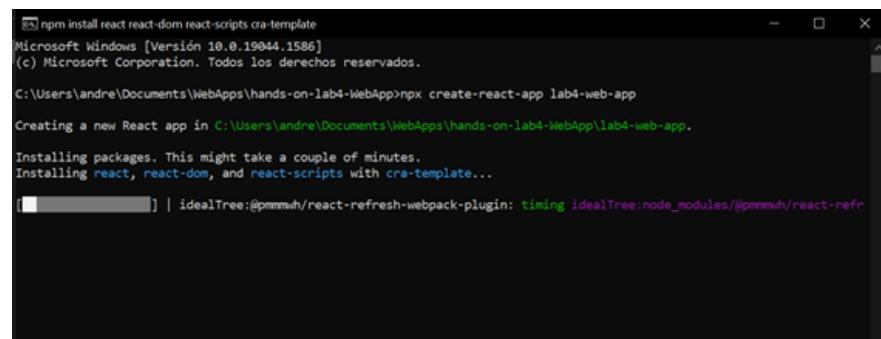


6.2.2. Una vez dentro de la carpeta seleccionada o creada para el proyecto en la barra de búsqueda escribir *cmd* y presionar la tecla *enter*.

6.2.3. Automáticamente se abrirá una ventada de línea de comandos (cmd) con la ruta destino del proyecto, aquí ejecutar: **npx create-React-app lab4-web-app** para iniciar el proceso de creación del proyecto con el nombre *lab4-web-app* (*este nombre depende del desarrollador*) Figura 8 y Figura 9.

Figura 7

Creación proyecto React

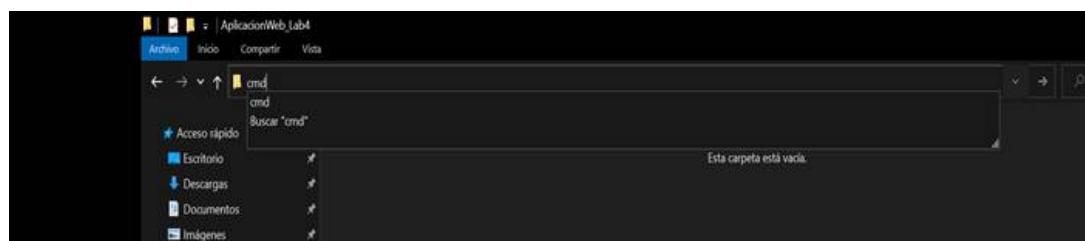


```
npm install react react-dom react-scripts cra-template
Microsoft Windows [Versión 10.0.19044.1586]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\andre\Documents\WebApps\hands-on-lab4-WebApp>npx create-react-app lab4-web-app

Creating a new React app in C:\Users\andre\Documents\WebApps\hands-on-lab4-WebApp\lab4-web-app.

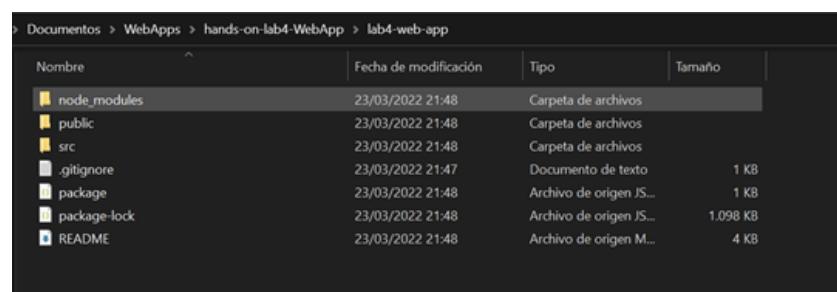
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
[██████████] | idealTree:@pmmwh/react-refresh-webpack-plugin: timing idealTree:node_modules/@pmmwh/react-rethr
```



6.2.4. Finalizado el proceso anterior se obtendrá la siguiente estructura de carpetas necesarias para el desarrollo del proyecto.

Figura 9

Estructura de carpetas



Nombre	Fecha de modificación	Tipo	Tamaño
node_modules	23/03/2022 21:48	Carpeta de archivos	
public	23/03/2022 21:48	Carpeta de archivos	
src	23/03/2022 21:48	Carpeta de archivos	
.gitignore	23/03/2022 21:47	Documento de texto	1 KB
package	23/03/2022 21:48	Archivo de origen JS...	1 KB
package-lock	23/03/2022 21:48	Archivo de origen JS...	1.098 KB
README	23/03/2022 21:48	Archivo de origen M...	4 KB

Figura 10

Finalización de la creación del proyecto

```

C:\Windows\System32\cmd.exe
npm audit fix --force
Run `npm audit` for details.

Created git commit.

Success! Created lab4-web-app at C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd lab4-web-app
  npm start

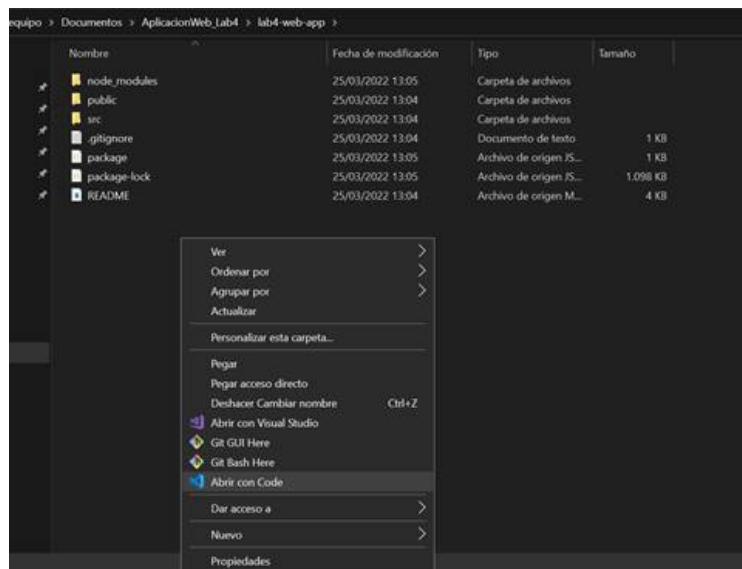
Happy hacking!

C:\Users\andre\Documents\AplicacionWeb_Lab4>

```

6.2.5. Abrir Visual Studio Code en la ruta del proyecto creado

Figura 11

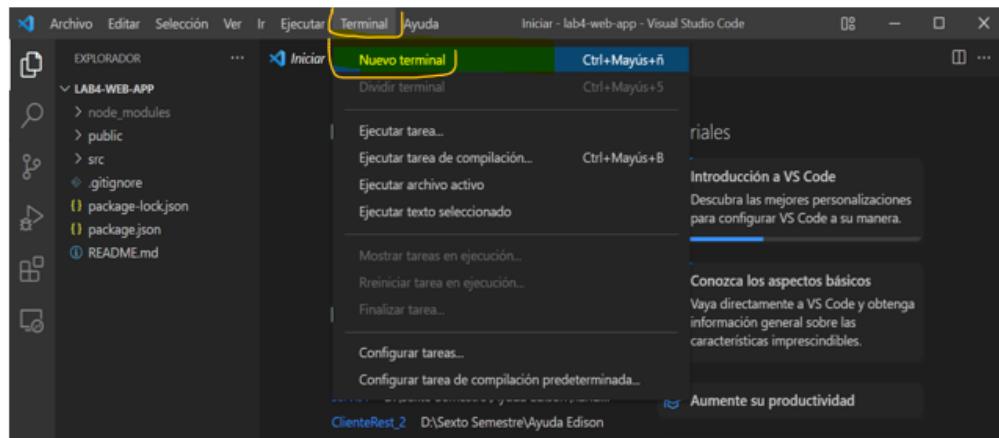


Instalación de bibliotecas necesarias para el proyecto:

6.2.6. En Visual Studio Code, abrir una nueva terminal para la instalación de las bibliotecas necesarias para el desarrollo del proyecto.

Figura 12

Terminal en Visual Studio Code



6.2.7. En la terminal escribir el comando `npm install axios`, seguido presionar intro.

Figura 13

Instalación de Axios

```

Reciente Vaya directamente a VS Code y obtenga
información general sobre las
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL powershell + × ☰ ^ ×
PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app> npm install axios
added 1 package, and audited 1410 packages in 6s
169 packages are looking for funding
  run 'npm fund' for details
6 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.
PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app>

```

Comando: `npm install react-bootstrap bootstrap@5.1.3`

Figura 14

Instalación de Bootstrap

```

Reciente Vaya directamente a VS Code y obtenga
información general sobre las
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL powershell + × ☰ ^ ×
PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app> npm install react-bootstrap bootstrap@5.1.3
added 19 packages, and audited 1429 packages in 11s
171 packages are looking for funding
  run 'npm fund' for details
6 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.
PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app>

```

Comando: `npm install --save @fortawesome/react-fontawesome`

Figura 15

Instalación de @fortawesome

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL powershell + × ☰ ^ ×

PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app> npm install --save @fortawesome/react-fontawesome

added 3 packages, and audited 1432 packages in 7s

171 packages are looking for funding
  run `npm fund` for details

6 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app>
```

Comando: npm install --save @fortawesome/react-fontawesome

Figura 16

Instalación de @fortawesome/react-fontawesome

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL powershell + × ☰ ^ ×

PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app> npm install --save @fortawesome/free-solid-svg-icons

added 1 package, and audited 1433 packages in 14s

171 packages are looking for funding
  run `npm fund` for details

6 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app>
```

Comando: npm i reactstrap react react-dom

Figura 17

Instalación de reactstrap

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL powershell + × ☰ ^ ×

PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app> npm i reactstrap react react-dom

added 3 packages, and audited 1436 packages in 7s

171 packages are looking for funding
  run `npm fund` for details

6 moderate severity vulnerabilities

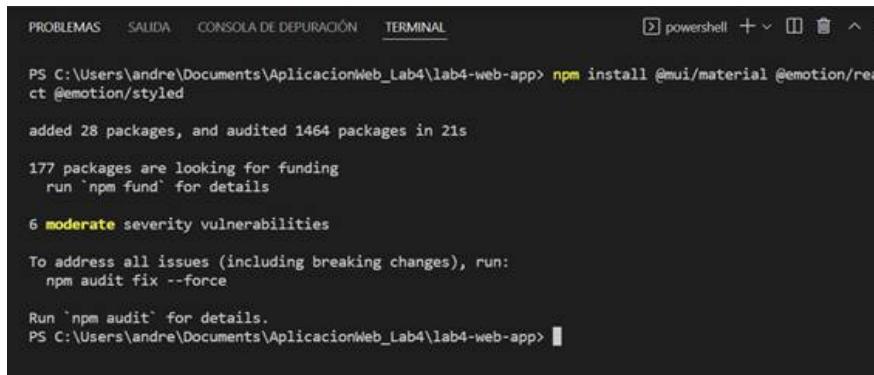
To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app>
```

Comando: npm install @m

Figura 18

Instalación de mui/material



```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL powershell + × ☰ × ×

PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app> npm install @mui/material @emotion/react @emotion/styled
added 28 packages, and audited 1464 packages in 21s
177 packages are looking for funding
  run `npm fund` for details
6 moderate severity vulnerabilities

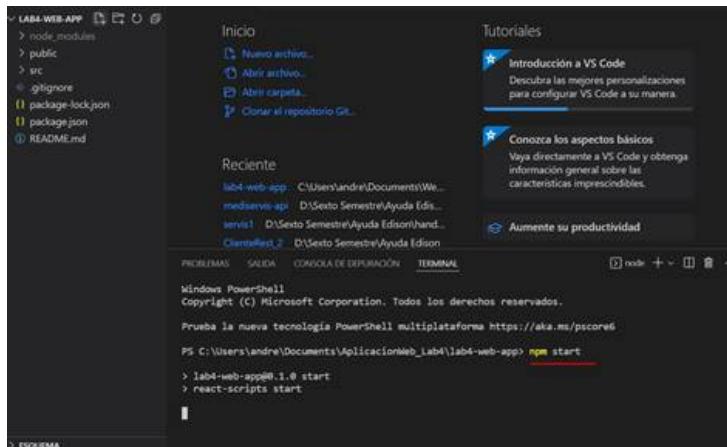
To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\andre\Documents\AplicacionWeb_Lab4\lab4-web-app>
```

6.2.8. En la terminal se ejecuta: **npm start**, para iniciar la ejecución del proyecto.

Figura 19

Ejecución de proyecto React bajo el comando **npm start**



Seguido se visualizara Figura 20 la que indica la creacion correcta del proyecto a trabajar.

Figura 20

Proyecto corriendo en el navegador port:3000

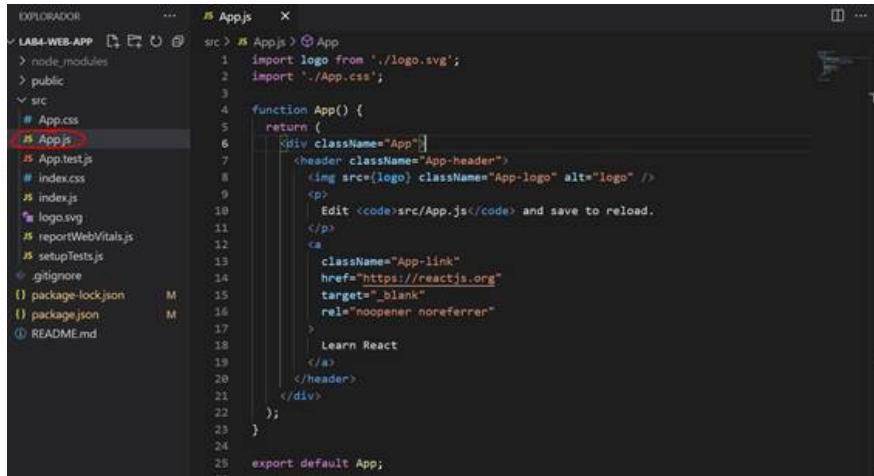


vii. Desarrollo de la aplicación

En Visual Studio Code se debe abrir el archivo App.js (Figura 21) que se encuentra en la carpeta /src de la estructura de carpetas del proyecto.

Figura 21

Archivo App.js



```
EXPLORADOR ... JS App.js x
LAB4-WEB-APP ...
> node_modules
> public
src
  # App.css
  # App.js
  # App.test.js
  # index.css
  # index.js
  # logo.svg
  # reportWebVitals.js
  # setupTests.js
  .gitignore
  package-lock.json M
  package.json M
  README.md

src > JS App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           Edit <code>src/App.js</code> and save to reload.
11         </p>
12         <a
13           className="App-link"
14           href="https://reactjs.org"
15           target="_blank"
16           rel="noopener noreferrer"
17         >
18           Learn React
19         </a>
20       </header>
21     </div>
22   );
23 }
24
25 export default App;
```

A continuación, se Borra todas las líneas de código y seguido se importan las bibliotecas previamente instaladas.

```
import React, { Component } from 'react';
import axios from "axios";
import "bootstrap/dist/css/bootstrap.min.css";
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faEdit, faTrashAlt } from '@fortawesome/free-solid-svg-icons';
import { Modal, ModalBody, ModalFooter, ModalHeader } from 'reactstrap';
import Button from '@mui/material/Button';
```

A continuación, se debe crear una constante que tendrá la URI del servicio RESTful al cual se conectará, se debe crear una clase Client para permitir la creación de un estado donde se encontrarán los datos y estados de ventanas flotantes a usar.

```
const URI="http://localhost:8081/clientes/clients/";
```

```
class Client extends Component {
state={
  data:[],
  modalInsert: false,
  modalDelete: false,
  form:{
    _id: "",
    name: "",
    lastName: "",
    age: "",
    subTotal: ""
  }
}
export default Client;
```

Después de crear los métodos GET, PUT, POST y DELETE necesarios para la manipulación de los datos, el método post es asíncrono (`async`), se ejecuta en segundo plano cuando se termine de llenar los datos y se cierre la ventana flotante por parte del usuario del web-app.

```
callGet=()=>{
  axios.get(URI).then(response=>{
```

```

    this.setState({data: response.data});
}).catch(error=>{
  console.log(error.message);
})
}

callPost=async()=>{
  delete this.state.form._id;
  await axios.post(URI,this.state.form).then(response=>{
    this.modallInsert();
    this.callGet();
  }).catch(error=>{
    console.log(error.message);
  })
}

callPut=()=>{
  axios.put(URI+this.state.form._id, this.state.form).then(response=>{
    this.modallInsert();
    this.callGet();
  })
}

callDelete=()=>{
  axios.delete(URI+this.state.form._id).then(response=>{
    this.setState({modalDelete: false});
    this.callGet();
  })
}

```

A continuación, se crea la ventana flotante de insertar (modallInsert) la que cambiará su estado a *true* cuando se la invoque, caso contrario tendrá un estado *false*, la función captureData ayudará a obtener los datos y mostrarlos al usuario en los campos input ejecutándose en segundo plano cuando se desee editarlos, esto mejora la usabilidad evitando problemas al momento que el usuario edite información.

```

modallInsert=()=>{
  this.setState({modallInsert: !this.state.modallInsert});
}
captureData=async data=>{
  data.persist();
  await this.setState({
    form:{
      ...this.state.form,
      [data.target.name]: data.target.value
    }
  });
  console.log(this.state.form);
}

```

A continuación, se debe crear una tabla con los datos correspondiente al proyecto, destacando la clase “table-responsive-md” la misma que permite hacer responsive a una tabla de forma rápida

Nota. Es posible modificar la tabla para un tamaño particular “table-responsive{-sm|-md|-lg|-xl}”

Los botones son creados y se les asigna un evento específico editar y eliminar representados por los iconos `faEdit` y `faTrashAlt` respectivamente de la biblioteca `FontAwesomelcon`.

```

render(){
  const {form}=this.state;
  return (
    <div className="App">
      <div class="table-responsive-md">
        <table className="table ">
          <thead>
            <tr>
              <th>ID</th>
              <th>Name</th>
              <th>LastName</th>
              <th>Age</th>
              <th>Sub Total</th>
            </tr>
          </thead>

          <tbody>
            {this.state.data.map(client=>{
              return(
                <tr>
                  <td>{client._id}</td>
                  <td>{client.name}</td>
                  <td>{client.lastName}</td>
                  <td>{client.age}</td>
                  <td>{client.subTotal}</td>

                  <td>
                    <button className="btn btn-primary" onClick={()=>{this.selectClient(client);
                      this.modallInsert()}}><FontAwesomelcon icon={faEdit}/></button>
                    {" "}
                    <button className="btn btn-danger" onClick={()=>{this.selectClient(client);
                      this.setState({modalDelete: true})}}><FontAwesomelcon icon={faTrashAlt}/></button>
                  </td>
                </tr>
              )
            })
          </tbody>
        </table>
      </div>
    )
}

```

A continuación, se crean las ventanas flotantes que permitirán el ingreso de datos cuando el usuario dese crea un nuevo cliente o cuando desee editar uno ya existente, se debe llamar la método captureData en cada input, porque este permitirá la vista de los datos extraídos por _id, cuando se desee editar y cuando se cree uno nuevo no mostrará datos ya que no existe un _id creado.

```

<Modal isOpen={this.state.modallInsert}>
  <ModalHeader style={{display: 'block'}}>
    <span style={{float: 'right'}} onClick={()=>this.modallInsert()}>x</span>
  </ModalHeader>
  <ModalBody>

    <div className="form-group">
      <label htmlFor="_id">ID</label>
      <input className="form-control" type="text" name="id" id="id" readOnly onChange={this.captureData} value={form?form._id: this.state.data.length+1}/>
      <br />
    </div>
  </ModalBody>
</Modal>

```

```

<label htmlFor="Name">Name</label>
    <input className="form-control" type="text" name="name" id="name" onChange={this.captureData} value={form?form.name: ""}/>
    <br />
    <label htmlFor="LastName">LastName</label>
        <input className="form-control" type="text" name="lastName" id="lastName" onChange={this.captureData } value={form?form.lastName: ""}/>
        <br />
    <label htmlFor="age">Age</label>
        <input className="form-control" type="text" name="age" id="age" onChange={this.captureData } value={form?form.age:""} />
        <br />
    <label htmlFor="subTotal">Sub Total</label>
        <input className="form-control" type="text" name="subTotal" id="subTotal" onChange={this.captureData } value={form?form.subTotal:""} />
    </div>
</ModalBody>

```

Se crea los botones correspondientes a cada acción, cada botón tendrá un evento asignado que llama al método POST o PUT.

```

<ModalFooter>
    {this.state.tipoModal=='insertar'?
        <button className="btn btn-success" onClick={()=>this.callPost()}>
            Insertar
        </button>: <button className="btn btn-primary" onClick={()=>this.callPut()}>
            Actualizar
        </button>
    }
    <button className="btn btn-danger" onClick={()=>this.modalInsert()}>Cancelar</button>
</ModalFooter>

```

A continuación, en App.js, se crea una alerta para confirmar la eliminación de un registro y al final de la tabla, se tiene la creación de un botón para crear un nuevo cliente con su respectivo evento.

```

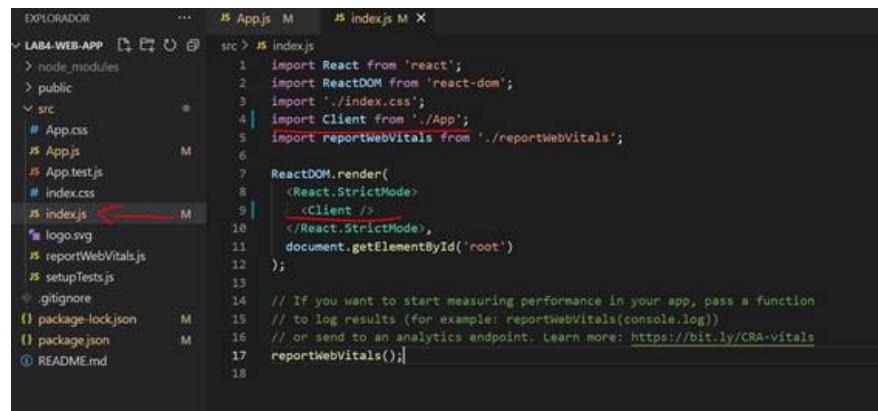
<ModalBody>
    Estás seguro que deseas eliminar el cliente: {form && form.name}
</ModalBody>
<ModalFooter>
    <button className="btn btn-danger" onClick={()=>this.callDelete()}>Sí</button>
    <button className="btn btn-secundary" onClick={()=>this.setState({modalDelete: false})}>No</button>
</ModalFooter>
</Modal>
<Button
    className='login'
    fullWidth
    variant="contained"
    style={{ color: 'white', backgroundColor: "#5DC1B9" }}
    onClick={()=>{this.setState({form: null, tipoModal: 'insertar'}); this.modalInsert()}}
>
    <strong><p>Add client</p></strong>
</Button>

```

Para finalizar con el desarrollo, se abre el archivo index.js (Figura 22) que se encuentra en la carpeta raíz del proyecto, se importa la clase Client y se la utiliza para ser mostrada.

Figura 22

Archivo index.js



```

EXPLORADOR          ...      JS App.js M      JS index.js M X
LABA-WEB-APP        <-->  src > JS index.js
> node_modules
> public
<--> src
  # App.css
  # App.js M
  JS App.test.js
  # index.css
  JS index.js M ←----- Red arrow points here
  logo.svg
  JS reportWebVitals.js
  JS setupTests.js
  .gitignore
  package-lock.json M
  package.json M
  README.md
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import Client from './App';
5  import reportWebVitals from './reportWebVitals';
6
7  ReactDOM.render(
8    <React.StrictMode>
9      <Client />
10     </React.StrictMode>,
11     document.getElementById('root')
12   );
13
14  // If you want to start measuring performance in your app, pass a function
15  // to log results (for example: reportWebVitals(console.log))
16  // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17  reportWebVitals();
18

```

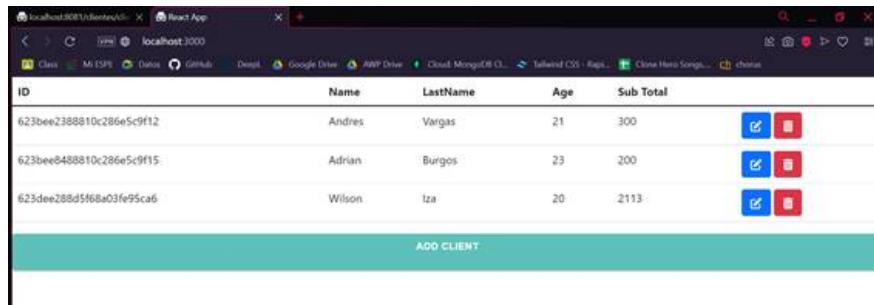
viii. Resultados

Se obtuvo una tabla con los datos de los clientes (Figura 23), donde se realiza un nuevo ingreso de datos presionando el botón “Add Client”, este botón muestra un formulario (Figura 24) donde el usuario podrá ingresar la información requerida a ser guardada, que será mostrada en la tabla principal (Figura 26). Se modificó un registro haciendo clic en el botón azul (ícono editar), obteniendo el formulario con los datos actuales a editar (Figura 25).

Una vez concluido este proceso, los datos se verán reflejados en la pantalla principal (Figura 27), al eliminar un registro de la base de datos (Figura 28), después de confirmar la acción, este registro habrá sido, también eliminado (visualmente) de la tabla (Figura 29). Los cambios realizados sobre los registros se verán reflejados en la base de datos en tiempo real (Figura 30).

Figura 23

Obtención de datos (Método Get)

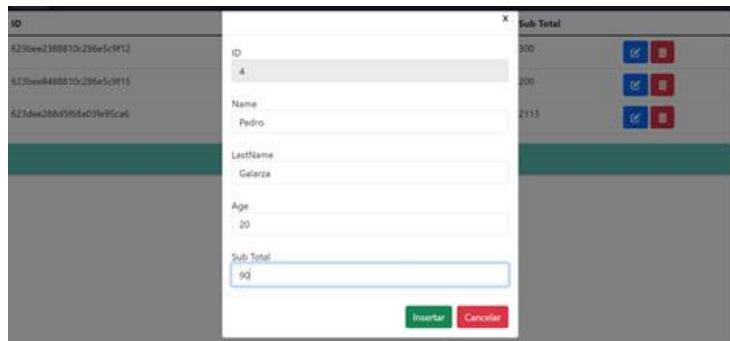


ID	Name	Lastname	Age	Sub Total	Action
623bee2388810c286e5c9f12	Andres	Vargas	21	300	
623bee8488810c286e5c9f15	Adrian	Burgos	23	200	
623dee288d5f68a03fe95ca6	Wilson	Iza	20	2113	

ADD CLIENT

Figura 24

Agregar un nuevo cliente (Método Post)



The form contains the following data:

ID	4	Sub Total	90
Name	Pedro		
Lastname	Galarza		
Age	20		
Sub Total	90		

Buttons at the bottom: Insertar (blue) and Cancelar (red).

Figura 25

Comprobación del funcionamiento del Método Post

ID	Name	LastName	Age	Sub Total	
623bee2388810c286e5c9f12	Andres	Vargas	21	300	
623bee8488810c286e5c9f15	Adrian	Burgos	23	200	
623dee288d5f68a03fe95ca6	Wilson	Iza	20	2113	
623e1be58d5f68a03fe95cb3	Pedro	Galarza	20	90	

ADD CLIENT

Figura 26

Modificar un cliente existente (Método Put)

ID	623e1be58d5f68a03fe95cb3
Name	Pedro A
LastName	Morales
Age	20
Sub Total	90

Actualizar **Cancelar**

Figura 27

Comprobación del funcionamiento del método Put

ID	Name	LastName	Age	Sub Total	
623bee2388810c286e5c9f12	Andres	Vargas	21	300	
623bee8488810c286e5c9f15	Adrian	Burgos	23	200	
623dee288d5f68a03fe95ca6	Wilson	Iza	20	2113	
623e1be58d5f68a03fe95cb3	Pedro A	Morales	20	90	

ADD CLIENT

Figura 28

Eliminar un cliente (Método Delete)

ID	Name	LastName	Age	Sub Total	
623bee2388810c286e5c9f12	Andres	Vargas	21	300	
623bee8488810c286e5c9f15	Adrian	Burgos	23	200	
623dee288d5f68a03fe95ca6	Wilson	Iza	20	2113	
623e1be58d5f68a03fe95cb3	Pedro A	Morales	20	90	

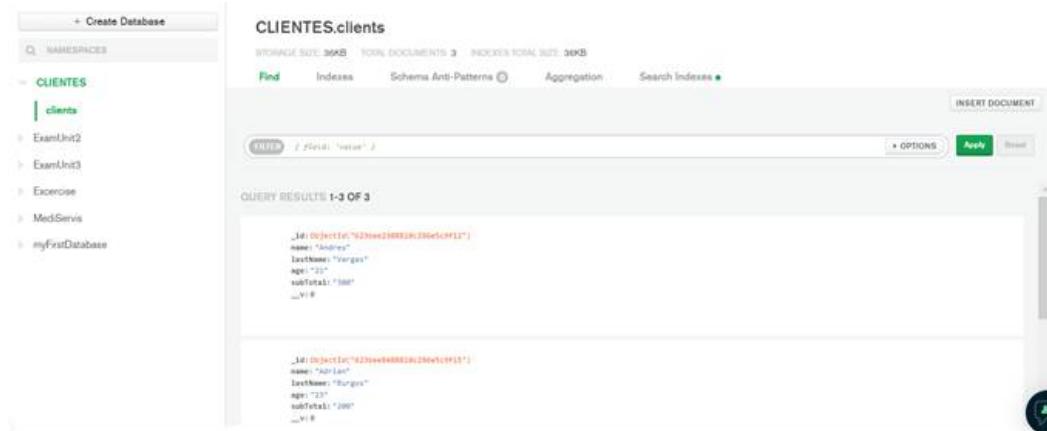
Figura 29

Comprobación del funcionamiento del método Delete

ID	Name	LastName	Age	Sub Total	
623bee2388810c286e5c9f12	Andres	Vargas	21	300	 
623bee8488810c286e5c9f15	Adrian	Burgos	23	200	 
623dee288d5f68a03fe95ca6	Wilson	Iza	20	2113	 
ADD CLIENT					

Figura 30

Persistencia de información en la base de datos en la nube (MongoDB Atlas)



The screenshot shows the MongoDB Atlas interface for the `CLIENTES.clients` collection. The left sidebar lists databases like `ExamUnit2`, `ExamUnit3`, `Excercise`, `MedServis`, and `myFirstDatabase`. The main area shows the `clients` document with the following data:

```
_id: ObjectId("623bee2388810c286e5c9f12")
name: "Andres"
lastName: "Vargas"
age: 21
subTotal: 300
__v: 0

_id: ObjectId("623bee8488810c286e5c9f15")
name: "Adrian"
lastName: "Burgos"
age: 23
subTotal: 200
__v: 0
```

ix. Recursos

9.1. Servicio RESTful utilizado en este proyecto:

<https://youtu.be/CQN1ZQ5jVyE>

x. Enlaces

10.1. GitHub

<https://github.com/elascano/ESPE-DESARROLLO-WEB-2022>

10.2. Video tutorial del desarrollo de esta WebApp

https://www.youtube.com/watch?v=DgJ_0jIDU3s

Recursos complementarios

Creación de un servicio REST

Crear servicio REST con nodejs



<https://www.youtube.com/watch?v=CQN1ZQ5jVyE>

Crear servicio REST con nodejs



Bibliografía



25 Latest Web Development Trends to Follow in 2022 | GMI. (n.d.). Retrieved June 1, 2022, from <https://www.globalmediainsight.com/blog/web-development-trends>

2022 Stats on Top JS Frameworks: React, Vue, Svelte & Angular. (n.d.). Retrieved May 24, 2022, from <https://www.tecla.io/blog/top-js-frameworks>

Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement: Aaron Gustafson, Jeffrey Zeldman: 9780983589501: Books. (n.d.). Retrieved June 7, 2022, from <https://www.amazon.com/Adaptive-Web-Design-Experiences-Progressive/dp/098358950X>

Al-Ghatrifi, I. N. S. (2015). Cloud computing: A key enabler for higher education in Sultanate of Oman. 2015 International Conference on Computer, Communications, and Control Technology (I4CT), 70–72. <https://doi.org/10.1109/I4CT.2015.7219539>

Angular vs react vs vue | npm trends. (n.d.). Retrieved May 24, 2022, from <https://www.npmtrends.com/angular-vs-react-vs-vue>

Brandon, L. (2014). Website Usability: Virtual Elephants of the Internet Room. Bosmol Social Media News. <http://bosmol.com/2014/02/website-usability-virtual-elephants-of-the-internet-room.html#.U3qrqc8RC4Q>

Responsive Web Design (Brief Books for People Who Make Websites, No. 4): Ethan Marcotte: 9781937557188: Books: Amazon.com. (n.d.). Retrieved June 7, 2022, from <https://www.amazon.com/Responsive-Design-Brief-People-Websites/dp/098444257X>

Web 4.0: The Internet of Things and AI. (2018, November 12). IT Strategy.

25 Latest Web Development Trends to Follow in 2022 | GMI. (n.d.). Retrieved June 1, 2022, from <https://www.globalmediainsight.com/blog/web-development-trends>

2022 Stats on Top JS Frameworks: React, Vue, Svelte & Angular. (n.d.). Retrieved May 24, 2022, from <https://www.tecla.io/blog/top-js-frameworks>

Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement: Aaron Gustafson, Jeffrey Zeldman: 9780983589501: Books. (n.d.). Retrieved June 7, 2022, from <https://www.amazon.com/Adaptive-Web-Design-Experiences-Progressive/dp/098358950X>

Al-Ghatrifi, I. N. S. (2015). Cloud computing: A key enabler for higher education in Sultanate of Oman. 2015 International Conference on Computer, Communications, and Control Technology (I4CT), 70–72.

<https://doi.org/10.1109/I4CT.2015.7219539>

Angular vs react vs vue | npm trends. (n.d.). Retrieved May 24, 2022, from <https://www.npmtrends.com/angular-vs-react-vs-vue>

Brandon, L. (2014). Website Usability: Virtual Elephants of the Internet Room. Bosmol Social Media News. <http://bosmol.com/2014/02/website-usability-virtual-elephants-of-the-internet-room.html#.U3qrqc8RC4Q>

Responsive Web Design (Brief Books for People Who Make Websites, No. 4): Ethan Marcotte: 9781937557188: Books: Amazon.com. (n.d.). Retrieved June 7, 2022, from <https://www.amazon.com/Responsive-Design-Brief-People-Sites/dp/098444257X>

Web 4.0: The Internet of Things and AI. (2018, November 12). IT Strategy. <https://en.itpedia.nl/2018/11/12/web-4-0-the-internet-of-things-en-ai/>

XaaS: ¿cuáles son todos los servicios que puedes usar en la nube? - ADN Cloud. (2019, July 2). #ADNCLOUD. <https://blog.mdcloud.es/xaas-todos-servicios-nube/>

Autoevaluación

1. It provides Several implementations of the same web page, but in different screen resolutions

- Adaptive design.
- Progressive design.
- Responsive design.

Pregunta 1 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

2. It was one of the first markup languages for mobil web pages

- HTML5
- WML.
- XHTML.

Pregunta 2 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

3. Library that allows to implement mobile apps using HTML, CSS and JavaScript

- React Native.
- React.
- Android.

Pregunta 3 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

4. Back-End web development tool.

- ReactJS.
- AngularJS.
- NodeJS.

Pregunta 4 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

5. Its main intent is to let users interact with others through the web

- Web 1.0
- Web 4.0
- Web 2.0

Pregunta 5 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

6. Type of Cloud Computing service that provides: development tools, Operating Systems, Data Bases, Programming languages as a whole

- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)
- Software as a Service (SaaS)

Pregunta 6 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

7. First Internet service that was used to do some kind of data mining

- Web
- e-mail
- telnet

Pregunta 7 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

8. Dropbox offers storage services, it is an example of

- SaaS
- PaaS
- IaaS

Pregunta 8 de 10

Atras

Siguiente

Enviar todo

Autoevaluación

9. Web is born during:

- 80s
- 2000s
- 90s

Pregunta 9 de 10

Atras

Siguiente

Enviar todo