IMPLEMENTACIÓN API REST SERVICIO FINANCIERO

MICHAEL MAURICIO GONZÁLEZ MEDINA
ÁNGEL RONALDO PEREZ DIAZ
MARLON JANPIERRE PUENTES GUZMÁN

UNIVERSIDAD PEDAGOGICA Y TECNOLOGICA DE COLOMBIA
SECCIONAL SOGAMOSO
SISTEMAS DISTRIBUIDOS
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
SOGAMOSO BOYACÁ
2021

IMPLEMENTACIÓN API REST SERVICIO FINANCIERO

MICHAEL MAURICIO GONZÁLEZ MEDINA 201711681
ÁNGEL RONALDO PEREZ DIAZ 201722317
MARLON JANPIERRE PUENTES GUZMÁN 201722323

MS(c) CAMILO HARVEY BOHORQUEZ DALLOS
DOCENTE

UNIVERSIDAD PEDAGOGICA Y TECNOLOGICA DE COLOMBIA
SECCIONAL SOGAMOSO
SISTEMAS DISTRIBUIDOS
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
SOGAMOSO BOYACÁ
2021

Tabla De contenido.

INTRODUCCIÓN

Hoy en día compartir información entre dos o más sistemas se ha convertido en un requisito fundamental en el desarrollo de software, en la actualidad existe una gran cantidad de API REST para la creación de servicios profesionales dado que para distribuir su funcionalidad son independientes del hardware, sistema operativo, lenguajes de programación, y así como son independientes para brindar funcionalidad, son independientes cuando se ejecutan y desarrollan su uso. REST es un tipo de arquitectura orientada a recursos para el desarrollo de servicios web reemplazando al SOAP.

El incremento de desarrollo de aplicaciones para empresas, compañías se ha vuelto complicado de realizar una misma aplicación para múltiples plataformas [1] por eso gracias a las API REST esta problemática se ha solucionado y ya no es necesario realizar una programación exclusiva para cada plataforma quitando o eliminando varias limitantes a la hora de consumir el servicio web.

En este articulo se realizara el estudio y implementación de una API REST para el servicio financiero servirá de apoyo para gestionar toda la información suministrada por los usuarios que necesiten crear una cuenta con una entidad bancaria esta API está destinada a proveer información del cliente así como gestionar específicamente sus tarjetas crédito y débito usando una técnica de arquitectura software llamada REST para lo cual se analizaron y estudiaron varios artículos necesarios para el desarrollo de esta. Todo esto tomando en cuenta la confiabilidad y privacidad en la transferencia de datos , ya que al incluir datos personales y privados del cliente los cuales debe suministrar para la correcta ejecución de la API se debe garantizar la protección de estos.

IDEA DE IMPLEMENTACIÓN.

Diseñar, crear e implementar un software que permitirá a cualquier usuario gestionar acciones correspondientes a tarjetas de crédito, de un banco, el software permitirá visualizar los productos que tiene disponibles con la entidad bancaria. Además, podrá solicitar una tarjeta débito, sin importar si ya cuenta con una, teniendo en cuenta que podrá tener varias. Por otro lado, la entidad bancaria suministra tarjetas de crédito, que el usuario podrá solicitar a través del software, el usuario deberá ingresar los datos y podrá obtener varias tarjetas de crédito, que serán validadas por la entidad, y será devuelto un resultado donde se proporcionará la información de la tarjeta que ha solicitado. A nivel lógico el funcionamiento consiste en enviar una petición mediante un protocolo de comunicación HTTP a un servidor, y dependiendo del tipo de petición (GET, POST, DEL, PUT) se introducen los parámetros requeridos por cada método, este, devolverá una respuesta (JSON) y realizará una gestión en la base de datos, esa respuesta finalmente será la que visualice el usuario arrojando el resultado para la búsqueda solicitada. Se tendrá como requisito almacenar cada petición con su respuesta en una base de datos que permitirá llevar a cabo las consultas realizadas al servidor.

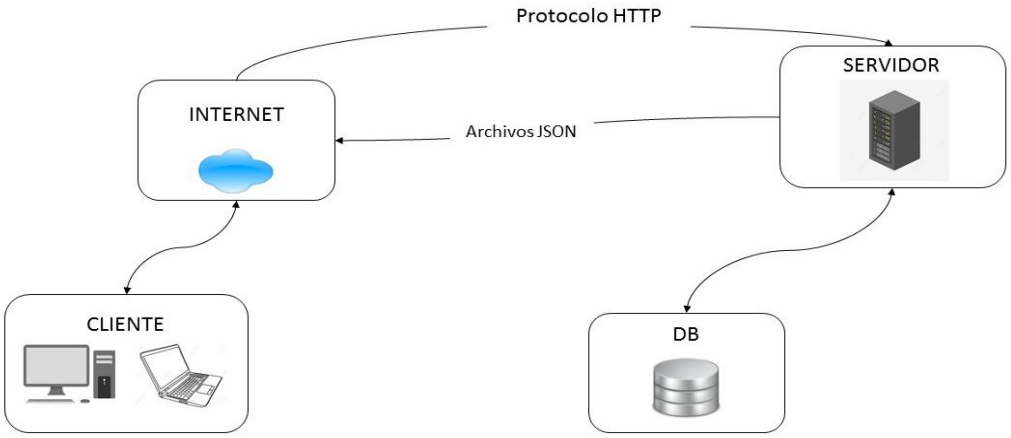

Fig. 1 Arquitectura

DESCRIPCIÓN EJECUCIÓN DE LA IDEA:

Realizamos la correspondiente investigación del desarrollo de un Api Rest, en la cual deducimos cuales eran las herramientas que íbamos para el desarrollo de nuestro proyecto. Como principal característica vamos a desarrollar un Api Rest con Spring Boot, framework que permite compilar nuestras aplicaciones Web como un archivo .jar, que podemos ejecutar como una aplicación Java normal, para el desarrollo del backend del Api Rest financiero, se hizo uso de Spring Tool Suite 4, que es un IDE basado en la versión Java EE de Eclipse, pero altamente personalizado para trabajar con el Framework Spring, y Spring Boot. Seguido a esto se realiza la creación del MVC del proyecto, y especificar en su archivo de configuración el servidor de base de datos, el puerto, el nombre de la tabla, y el servidor de aplicaciones para ser consumido.

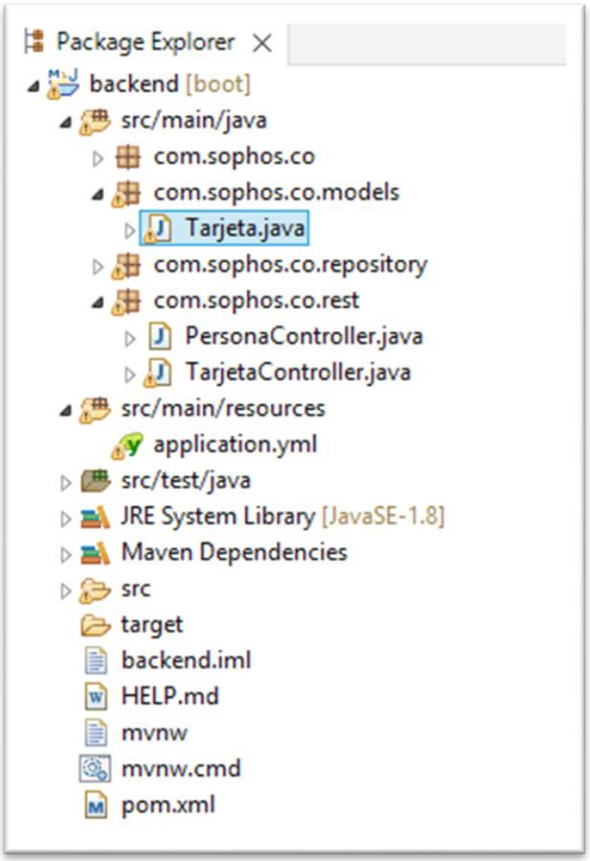

Fig. 2. Carpeta Con Los Paquetes, Clases, Y Librerías Para El Desarrollo Del Backend.

Una vez creados los métodos del modelo Tarjeta, se crea el controlador de esta, llamada *TarjetaController.java*, en la cual se especifican los métodos del Api Rest (GET, POST, DEL, PUT), esto con la ayuda del Framework Spring Boot; que contiene la lógica de proyecto, y es el que recibe las peticiones de los clientes.

Entonces, el flujo de este backend consiste en: recibir la petición por el controlador de la clase Tarjeta, que inmediatamente llama un servicio (GET, POST, DEL, PUT) que ejecuta la lógica de la aplicación, este controlador hace uso de un repositorio de tarjetas *(*del paquete *com.sophos.co.repository),* que ofrece las conexiones con la base de datos y su a vez hace un llamado al modelo, para conocer el tipo de información que va a utilizar. Una vez creado el backend,

se exponen los servicios a consumir para los clientes, esto por medio de un método del framework llamado *@RequestMapping*, que expone una parte de la url del servicio que se va a consumir. Cada función del controlador hace uso del método *Mapping*, pero de una forma mas corta y especificando su servicio (GET, POST, DEL, PUT), a continuación, un ejemplo:

```
//GUARDA UNA TARJETA CON TODOS SUS DATOS
@CrossOrigin(origins = "*")
@PostMapping("/tarjeta")
public ResponseEntity<Tarjeta> findById(@RequestBody Tarjeta tarjeta) {

    Optional<Tarjeta> tarjetaOptional =tarjetaRepository.findById(tarjeta.getId());

    if (!tarjetaOptional.isPresent()) {
        Tarjeta tarjeta2=tarjetaRepository.save(tarjeta);
        return ResponseEntity.ok(tarjeta);
    }else {
        return new ResponseEntity("Esta no existe", HttpStatus.NOT_FOUND);
    }
}

//OBTENER TODOS LOS DATOS DE UNA TARJETA POR EL ID(NUMERO)
@CrossOrigin(origins = "*")
@GetMapping("/tarjeta/{id}")
public ResponseEntity<Tarjeta> findIdTarjeta(@PathVariable("id") Long id) {
    Optional<Tarjeta > optional = tarjetaRepository.findById(id);

    if (optional.isPresent()) {
        return ResponseEntity.ok(optional.get());
    }else {
        return new  ResponseEntity("ESTA TAJETA NO EXISTE...!", HttpStatus.NOT_FOUND);
    }
}
```

Fig. 3. Funciones De La Clase *Tarjetacontroller.Java* Y Donde Se Visualiza El Uso De Los Servicios *Post Y Get* Acompañados Del Método Mapping, Métodos Del Framework Spring Boot.

Para realizar la comprobación de la lógica de estos métodos, se hace uso del cliente Postman, aplicación que nos permite realizar pruebas API, mediante la forma cliente HTTP que nos da la posibilidad de testear 'HTTP requests' a través de una interfaz gráfica de usuario, por medio de la cual obtendremos diferentes tipos de respuesta que posteriormente deberán ser validados. Se debe tener en cuenta que las los testeos se deben realizar por medio de el tipo de archivo JSON, con todos los campos de la tabla *Tarjeta*. Postman nos ofrece unas respuestas según el servicio requerido, y la forma en que la validamos fue por medio de las gestiones a la base de datos de la tabla Tarjeta. Cuando se realizó la validación de la lógica, procedimos a desarrollar la creación de un Front-end.

Como las peticiones se deben realizar en forma de JSON, se hizo uso de Angular que es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. La petición a cada servicio se realizo por medio del protocolo HTTP y la url del servicio requerido, un ejemplo a continuación:

```typescript
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class TarjetaService {

  constructor(private http: HttpClient) { }

  getListTarjetas(): Observable<any> {
    return this.http.get("http://localhost:8080/apirest/tarjeta");
  }

  deleteTarjeta(id: number): Observable<any> {
    return this.http.delete("http://localhost:8080/apirest/tarjeta/tarjeta/" + id);
  }

  saveTarjeta(tarjeta: any): Observable<any> {
    console.log(tarjeta);
    return this.http.post("http://localhost:8080/apirest/tarjeta/tarjeta", tarjeta);
  }

  updateTarjeta(id: number, tarjeta: any): Observable<any> {
    return this.http.put("http://localhost:8080/apirest/tarjeta/tarjeta", tarjeta);
  }
}
```

Fig. 4 Conexión Api-Rest

Y de esta forma poder realizar la conexión con el Back-end. Por otro lado, se hizo uso del servidor MySQL que permitió la conexión a la base de datos, y como cliente de pruebas phpMyAdmin que es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando un navegador web.

De esta forma desarrollamos la implementación de nuestro proyecto Api Rest financiero.

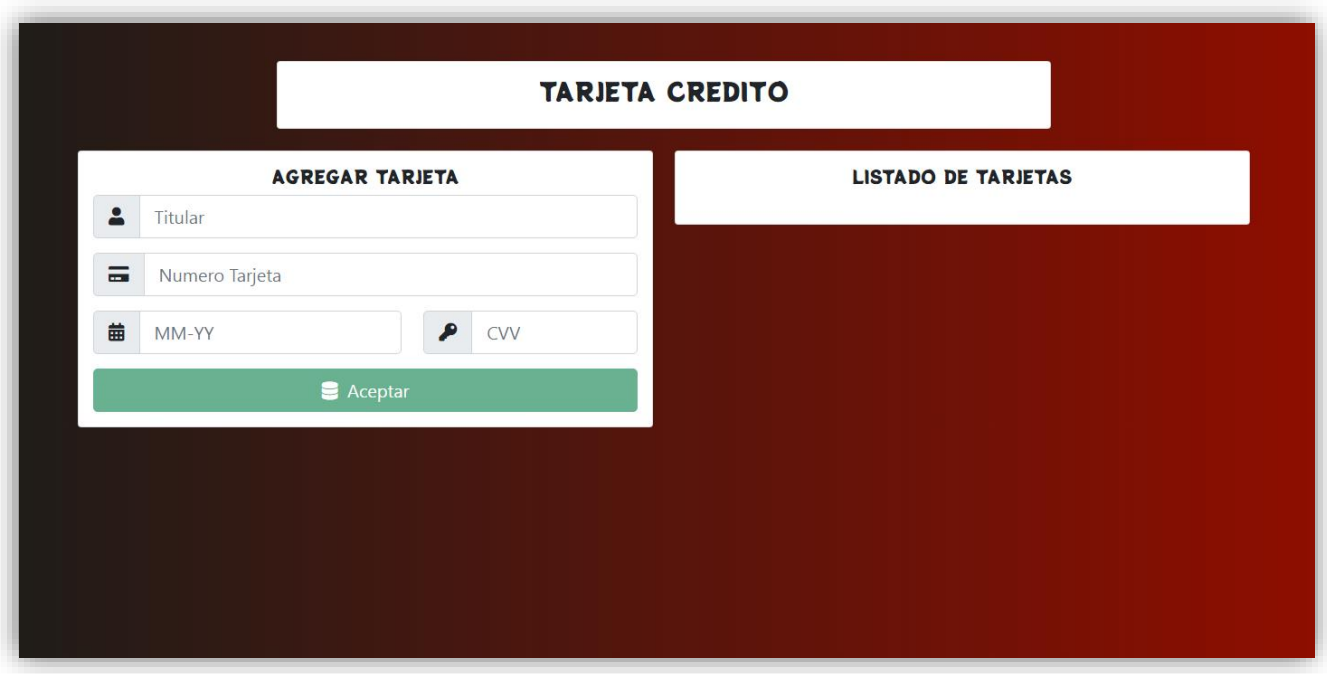

Fig. 5. Front End En Angular Para La Gestión De Las Tarjetas De Crédito.

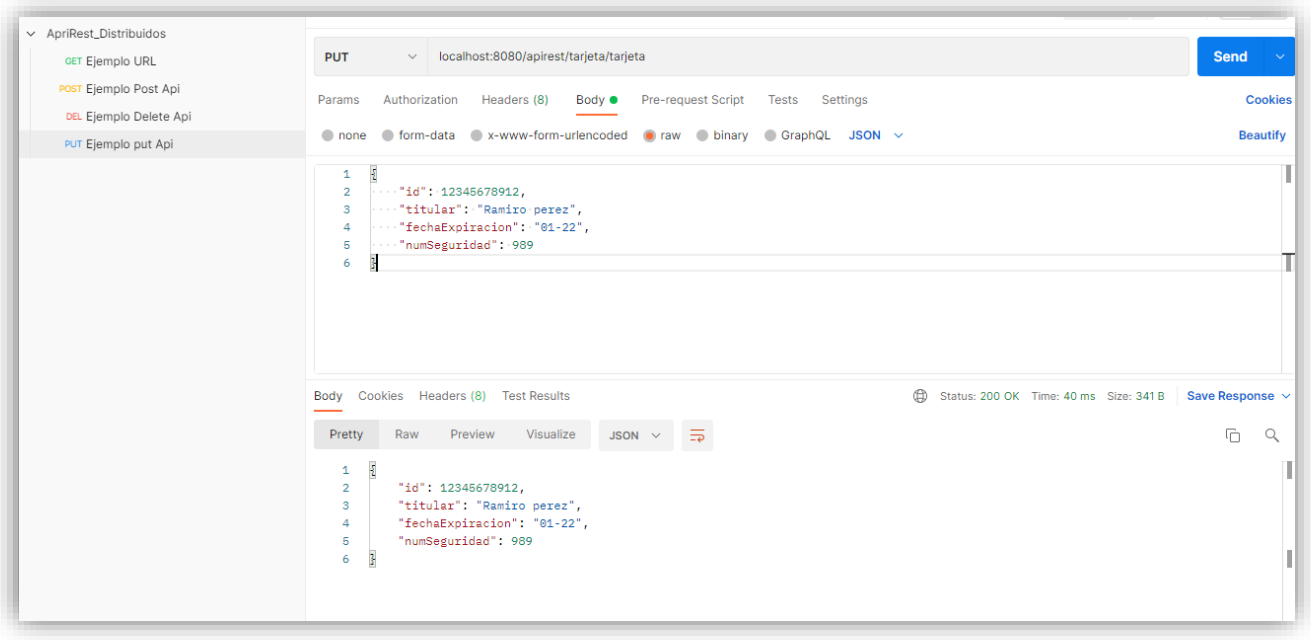

Fig.6. Uso De Postman Como Cliente Para Realizar La Comprobación De La Lógica.

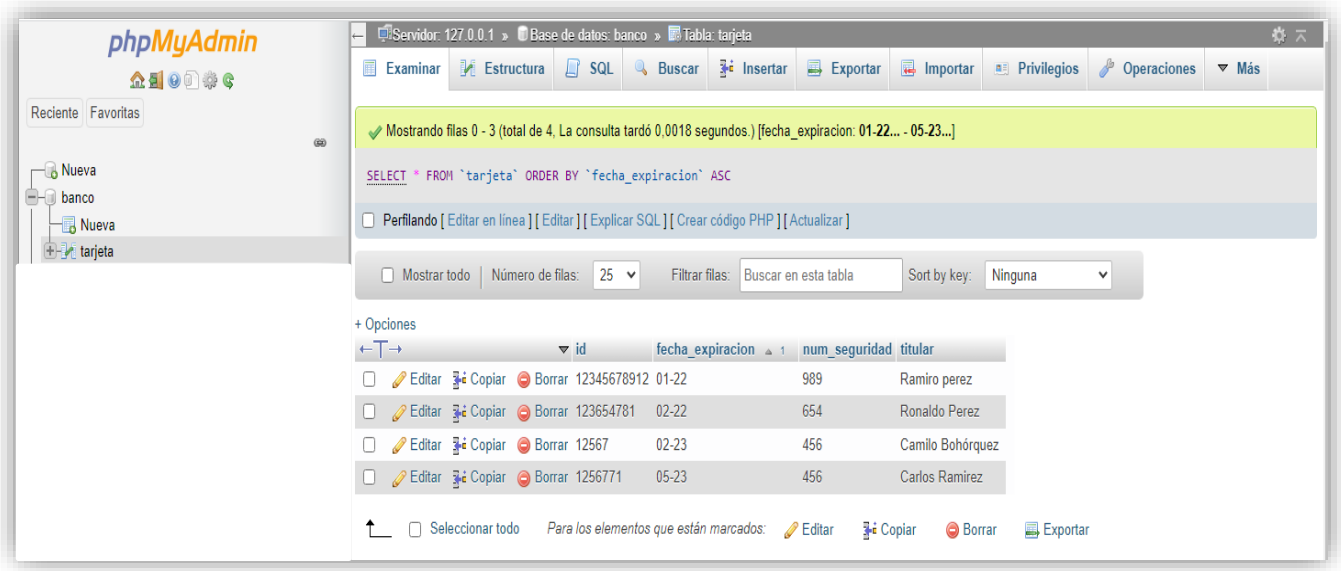

Fig. 7. Cliente De Pruebas De Base De Datos.

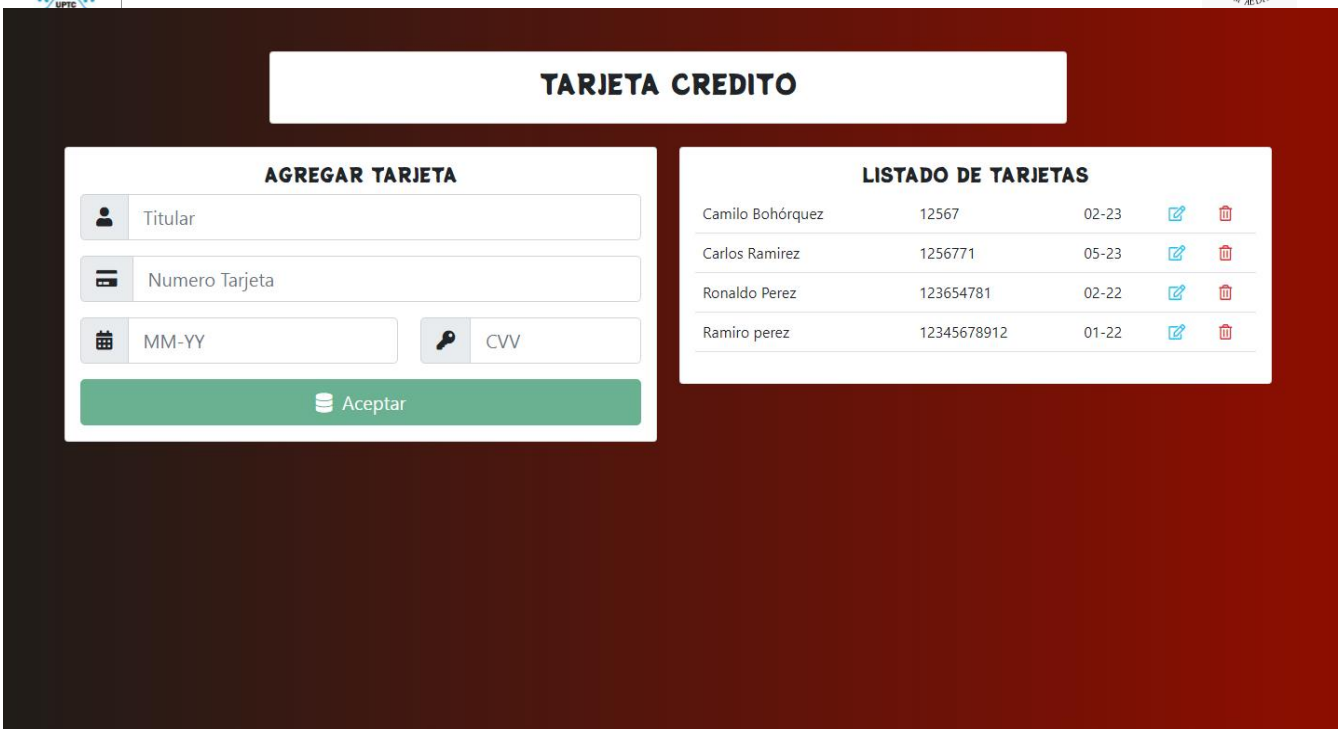

Fig. 8. Front-End Con Los Datos De La Base De Datos.

MARCO CONCEPTUAL.

API

Una API es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita del consumidor (la llamada) y el que requiere el productor (la respuesta). [14]

REST

REpresentational State Transfer, es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP.la mayoría de las aplicaciones que se desarrollan para servicios profesionales disponen de una API REST para el intercambio de información entre el front y el back. Lo que la hace tan potente es precisamente el aislamiento que proporciona entre la lógica del back-end y cualquier cliente consumidor de éste. Esto le permite ser usada por cualquier tipo de cliente: web, móvil, etc. Así, cualquier dispositivo/cliente que entienda de HTTP puede hacer uso de su propia API REST de manera muy simple.[3]

HTTP

HTTP, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos. Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador Web. Así, una página web completa resulta de la unión de distintos sub-documentos recibidos, como, por ejemplo: un documento que especifique el estilo de maquetación de la página web (CSS), el texto, las imágenes, vídeos, scripts, etc…[4]

Códigos de estado HTTP

Los códigos de estado son las respuestas HTTP que un servidor puede devolver cuando recibe una petición HTTP request, y se describen de forma abreviada mediante respuesta HTTP. El primer dígito del código de estado especifica uno de los 5 tipos de respuesta, lo mínimo para que un cliente pueda trabajar con HTTP es que reconozca estas 5 clases, estos son: 1XX Respuestas informativas, 2XX Peticiones correctas, 3XX Redirecciones, 4XX Errores del cliente, 5XX Errores de servidor. [13]

URI

Los identificadores uniformes de recursos (URI) son el elemento más simple de la arquitectura web y el más importante. Los URI han sido conocidos por muchos nombres: direcciones WWW, identificadores universales de documentos, identificadores universales de recursos.[2]
Los nombres de URI no deben implicar una acción, por lo tanto, debe evitarse usar verbos en ellos. Deben ser únicas, no debemos tener más de una URI para identificar un mismo recurso. Deben mantener una jerarquía lógica. Los filtrados de información de un recurso no se hacen en la URI.[2]

POSTMAN

Postman es un entorno de desarrollo de APIs que nos permite diseñar, probar y monitorizar servicios REST, también admite otras funcionalidades que se salen de lo que engloba el testing de este tipo de sistemas. [15]

MONGODB

MongoDB es una base de datos NoSQL orientada a documentos que se utiliza para el almacenamiento de datos de gran volumen. En lugar de usar tablas y filas como en las bases de datos relacionales tradicionales, MongoDB hace uso de colecciones y documentos. Los documentos constan de pares clave-valor que son la unidad básica de datos en MongoDB. Las colecciones contienen conjuntos de documentos y funciones que son equivalentes a las tablas de bases de datos relacionales. [16]

JSON

Su nombre corresponde a las siglas JavaScript Object Notation o Notación de Objetos de JavaScript, es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas. JSON es un formato de texto completamente independiente de lenguaje. Dichas propiedades hacen de JSON un formato de intercambio de datos ideal para usar con API REST o AJAX. El archivo es básicamente una alternativa más simple y liviana al XML (Lenguaje de marcado extenso, por sus siglas en inglés) que cuenta con funciones similares, [8] debido a su estructura ligera y compacta. Según la descripción de Stack Overflow, JSON "define seis tipos de valores: nulo, números, cadenas, booleanos, matrices y objetos", es un estándar basado en texto plano para el intercambio de datos, por lo que se usa en muchos sistemas que requieren mostrar o enviar información para ser interpretada por otros sistemas.[7]

Una de las características de JSON, al ser un formato que es independiente de cualquier lenguaje de programación, es que los servicios que comparten información por este método no necesitan hablar el mismo idioma, es decir, el emisor puede ser Java y el receptor Python, pues cada uno tiene su propia librería para codificar y decodificar cadenas en este formato[7]

Cliente-Servidor

Es uno de los estilos arquitectónicos distribuidos más conocidos, el cual está compuesto por dos componentes, el proveedor y el consumidor. El proveedor es un servidor que brinda una serie de servicios o recursos los cuales son consumidos por el cliente. En una arquitectura Cliente-Servidor existe un servidor y múltiples clientes que se conectan al servidor para recuperar todos los recursos necesarios para funcionar, en este sentido, el cliente solo es una capa para representar los datos y se detonan acciones para modificar el estado del servidor, mientras que el servidor es el que hace todo el trabajo pesado;[10] esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) y este le da respuesta, Los clientes y los servidores pueden estar conectados a una red local o una red amplia, como la que se puede implementar en una empresa o a una red mundial como lo es el Internet. Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla como según le convenga. [9]

Java Spring Boot

Es una de las herramientas principales del ecosistema de desarrollo web backend con Java, principalmente en arquitecturas basadas en servicios web (REST y SOAP) y microservicios. [11] Spring Boot es una de las tecnologías dentro del mundo de Spring, y al momento de construir aplicaciones con Spring Framework, existen una serie de pasos, fundamentalmente existen tres

pasos a realizar; el primero es crear un proyecto Maven/Gradle y descargar las dependencias necesarias, en segundo lugar, desarrollamos la aplicación y en tercer lugar la desplegamos en un servidor, y de esta forma no deberíamos tener que estar eligiendo continuamente las dependencias y el servidor de despliegue.

Con Spring Boot nos olvidamos de tener que desplegar artefactos Jar o War de manera independiente en uno o muchos servidores web diferentes. Porque nos provee una serie de contenedores web servlet para que se despliegue nuestra aplicación automáticamente solo con un "Run". [11]

Así mismo, de estos contenedores web podemos elegir el que nos convenga: Tomcat, Jetty u Undertow porque vienen embebidos como dependencias y simplemente agregamos el que se adapte a nuestras necesidades. Todo esto con el gestor de dependencias que elijamos, bien sea Maven o Gradle, sin tocar un servidor o configurar otro tipo de cosas. [11]

Hoy por hoy la arquitectura REST se utiliza a menudo en aplicaciones móviles y este estilo REST hace énfasis en que las interacciones entre los clientes los servicios se mejoran al tener un número limitado de operaciones, basados en lo anterior existe algunas reglas de la arquitectura REST que se sugieren implementar en nuestro proyecto apoyados en proyectos que en el pasado han incorporado el estilo REST:

- Interfaz uniforme
- Peticiones sin estado
- Separación cliente-servidor
- Sistema de capas
- Código bajo demanda
- Cacheable

Una API REST describe un conjunto de recursos, y un conjunto de operaciones que pueden ser llamadas en esos recursos. Las operaciones de una API REST pueden llamarse desde cualquier cliente HTTP, incluido el código JavaScript del lado del cliente que se ejecuta en un navegador web. La API REST tiene una ruta base, que es similar a la raíz de un contexto. Todos los recursos de una API REST se definen en relación con su ruta base. La ruta base puede utilizarse para proporcionar aislamiento entre diferentes APIs REST, así como aislamiento entre diferentes versiones de la misma API REST. Por ejemplo, se puede construir una API REST para exponer una base de datos de estudiantes a través de HTTP.[18]

REST no se define como un protocolo que se desarrolla sobre HTTP, sino que sigue restricciones mientras desarrollo de APIs. Para la solicitud y la respuesta, REST utiliza HTTP para definir la acción deseada. Para permitir las comunicaciones, existen hay métodos de recursos o tipos de solicitud disponibles:[19]

GET : Recuperar información del recurso.

POST: Se ha desarrollado un nuevo recurso subalterno desarrollado.

PUT: Se actualiza el recurso existente.

DELETE: Eliminar el recurso existente identificado por URI de la solicitud.

Por ejemplo, para la implementación de una API REST para un sistema de recomendación se hizo uso de los métodos POST, GET y DELETE. Con el método GET se obtiene información del servidor. Se recogen los datos que están en memoria en el servidor, o en una base de datos conectada a éste. No importa que para esto se tenga que enviar a través de la petición algún dato en concreto para obtener la respuesta del servidor esperada, como pueda ser en este caso obtener la información de un ítem en concreto, u obtener la información de todos los usuarios que se encuentran en el sistema. Si el sistema no puede recopilar la información solicitada, la respuesta llevará un código de error. El método POST, al contrario que el GET, es para enviar información al servidor. El servidor será el encargado de procesar dicha información y agregarla o actualizarla en memoria o en la base de datos.[20]

En este servicio, cuando hacemos una llamada a POST, el servidor procesa los datos que le llegan a través de la petición, y devuelve información de relevancia en la respuesta, como, por ejemplo, al añadir un usuario, se devuelve el identificador asignado para dicho usuario. El método DELETE, elimina información del servidor. El servidor recibe en la petición un identificador del objeto a eliminar, y elimina dicho elemento del servidor, así como toda su información, y en la respuesta devuelve un código de confirmación. Si el servidor no es capaz de eliminar el elemento, porque, por ejemplo, el elemento a borrar no existe en el servicio, se devuelve un código de error en la respuesta.[20]

```typescript
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class TarjetaService {

  constructor(private http: HttpClient) { }

  getListTarjetas(): Observable<any> {
    return this.http.get("http://localhost:8080/apirest/tarjeta");
  }

  deleteTarjeta(id: number): Observable<any> {
    return this.http.delete("http://localhost:8080/apirest/tarjeta/tarjeta/" + id);
  }

  saveTarjeta(tarjeta: any): Observable<any> {
    console.log(tarjeta);
    return this.http.post("http://localhost:8080/apirest/tarjeta/tarjeta", tarjeta);
  }

  updateTarjeta(id: number, tarjeta: any): Observable<any> {
    return this.http.put("http://localhost:8080/apirest/tarjeta/tarjeta", tarjeta);
  }
}
```

Fig. 9. Conexión Api-Rest Con Los Métodos Especificados.

MARCO REFERENCIAL.

La historia de las API's comienza a escribirse gracias al comercio electrónico y su intención era impulsar el área de soluciones, luego de esto las plataformas sociales experimentaban en el mundo de las APIs,más tarde Google integró sus herramientas en sitios de terceros y actualmente, en el mercado global se encuentran numerosas soluciones de software referentes a arquitecturas REST API y microservicios, todas ellas con diferentes plataformas de desarrollo y diferentes tecnologías.[1] La motivación para desarrollar REST fue crear un modelo arquitectónico de cómo debería funcionar la Web, de modo que pudiera servir como marco de referencia para los estándares del protocolo Web. REST se ha aplicado para describir la arquitectura web deseada, ayudar a identificar problemas existentes, comparar soluciones alternativas y garantizar que las extensiones de protocolo no violen las restricciones centrales que hacen que la web sea exitosa.[2]

Una de las luchas que se han llevado los últimos años es ofrecer transparencia de la información con la entrega de datos para apoyar la toma de decisiones en el mercado. En este sentido se busca seguir avanzando en nuevas plataformas de información en el ámbito financiero ya que solo se cuenta hoy en día con servicios web para visualizar mapas, servicios meteorológicos, etc. Cabe destacar que en el mundo ya existe una implementación realizada por el Banco Mundial a través de su api pública. [5] dado el crecimiento que ha tenido el internet, es de esperarse que todo esté conectado, haciendo que la implementación de los API REST vaya aumentando de manera considerable proponiendo nuevos usos como lo son la agrupación de datos y el almacenamiento esto teniendo como plus que la latencia de la comunicación al componer dos o más servicios web REST en solo uno se reduce.

Además de esto uno de los usos más factibles son las api de busqueda en sistemas convirtiendo la datos en la materia prima para la toma de decisiones con esto los analistas desarrollan proyectos de investigacion e innovacion implementando una api rest que organice, categorize y recupere aquellos datos de interés a trabajar.[6] Por lo tanto, este estilo se centra más en interactuar con recursos con estado, que con mensajes y operaciones. En particular, el lanzamiento de REST como protocolo de intercambio y manipulación de datos en los servicios de internet produjo un gran cambio en el desarrollo de software en los últimos años.[17]

DESARROLLO ACTUAL:

Universidad Pedagógica y Tecnológica de Colombia **Seccional Sogamoso**
Nombre: Generación de ejemplos de API REST usando Javadoc
Parte 1. Ficha de lectura
1. Referencia bibliográfica (APA 2020)
Rantanen, P. (2017). REST API Example Generation Using Javadoc. Pori Department, Tampere University ofTechnology, Pori, Finland, 14, 466-477. https://doi.org/10.2298/CSIS161022009R
1.2 Objetivo del autor / Texto
El objetivo principal es facilitar la generación de ejemplos de llamadas a métodos para servicios web (RESTful)
1.3 Partes / estructura del texto
Dar formato y editar la documentación puede ser un proceso tedioso,independientemente de lo bien que estén hechas sus plantillas de documentación. Especialmente, mantener los ejemplos de código actualizados puede llevar mucho tiempo y ser propenso a errores. La investigación presentada en este artículo describe una extensión de Javadoc que puede usarse para producir datos de ejemplo en combinación con ejemplos de llamadas a métodos API generados automáticamente, y explica cómo las API en nuestra implementación están organizadas para facilitar aún más el proceso de documentación automática
1.4 Hipótesis o idea central del Autor / Texto
Diseño e implementación de interfaces de una API para un servicio de F-E al que acceden los clientes y B-E de análisis decontenido Descripción de como el RESTLet, extensión para la herramienta Javadoc se utiliza para generar documentación,y más específicamente, para generar automáticamente ejemplos de llamadas a la API web
1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)
El método implementado tiene factores sobresalientes en la documentación, se logró evidenciar una desventaja la cual genera una mayor carga de trabajo en la implementación de los componentes desoftware necesaria para el proceso de documentación automática
1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)
El desarrollo de la documentación y actualización de la misma puede resultar una tarea tediosa y que muchas veces puede llevar el mismo tiempo que el tiempo de implementación y desarrollo de un software,es por esto que al tener la capacidad de implementar automatización en una labor tan importante como la documentación puede marcar una brecha importante en tener un muy buen producto

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: Mejora del rendimiento a través de la agrupación de API abierta Rest para redes de sensores inalámbricos

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Hyuk Park, J., Young-Sik, J., & Cho, M. (2013). Improving Performance through REST Open API Grouping forWireless Sensor Network. International Journal of Distributed Sensor Networks, 2013, 1-13. https://doi.org/10.1155/2013/958241

1.2 Objetivo del autor / Texto

agrupación de API REST y almacenamientoen caché de API REST.

1.3 Partes / estructura del texto

Con el crecimiento que está teniendo Internet, se espera que todos los dispositivos, incluidas las computadoras, estén conectados a Internet, como se llama IoT.En esta investigación, Nos enfocamos en diseñar el concepto de Map Reduce como un enfoque a través de la agrupación de servicios web de varios servicios web en uno.

En primer lugar, la composición del servicio web reduce el consumo de energía y la latencia de la comunicación al componer dos o más servicios web REST en uno. En segundo lugar, la técnica de almacenamiento en caché del servicio web proporciona unacceso rápido al que se accede recientemente o al que se accede con frecuencia. Realizamos experimentos con el servidor de servicios web REST de Jersey. El resultado experimental muestra que nuestro enfoque supera a los enfoques convencionales

1.4 Hipótesis o idea central del Autor / Texto

la técnica de almacenamiento en caché del servicio web proporciona unacceso rápido al que se accede recientemente o al que se accede con frecuencia.

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

- Se encontró un número óptimo de arquitecturas de servidores web REST en paralelo bajo ciertas tarifas de
  llegada de solicitud. Se demostró el rendimiento de laarquitectura propuesta
- Se propuso un método y un sistema de composición de servicios web RESTful que permita a los desarrolladores Implementar su servicio web a través del protocolo HTTP
- Se tiene un beneficio en este enfoque dado al hecho de que la interfaz de trabajo de la presentación es muy familiar para

los usuarios porque es REST Open API

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

EL artículo expone un eje a tener en cuenta en la evolución de la IoT y es la versatilidad que puede propiciar la API REST y su eje arquitectónico en la solución de más de un problema

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: Desarrollo de una API REST para un sistema académico de terceros

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Ferreira, A. S., Nicacio, J. M., Ferreira, G. V., Santos Filho, G. M., & Rodrigues, V. S. (2018). Desenvolvimentode uma API REST para um Sistema Acadêmico de Terceiros. Revista de Sistemas e Computação, Salvador, 8(2), 536-546. http://biblio.uptc.edu.co:2304/ehost/detail/detail?vid=0&sid=d6a10824-9759-4af2-a589-6db039d1d973%40sdc-v-sessmgr03&bdata=Jmxhbmc9ZXMmc2l0ZT1laG9zdC1saXZl#db=iih&AN=134691613

1.2 Objetivo del autor / Texto

Presentar la hoja de ruta en el desarrollo de una API para buscar un Sistema Académico (AS)

1.3 Partes / estructura del texto

Parte del problema recae en que la infraestructura actual es compleja costosa e improductiva de mantener, ya que los equipos decada uno de los proyectos necesitan comprender, dominar y acceder directamente el modelo de datos relaciones del sistema el desarrollo y uso de servicios web, fue una de las maneras de contrarrestar los inconvenientes actuales, a través de una API REST,donde se buscó hacer posible la consulta de datos de actividad gestionado por la SA.
La API facilita la el acceso a la información haciendo posible que varios proyectos de desplazamiento interno se vean reducida lacomplejidad para el acceso a la información

1.4 Hipótesis o idea central del Autor / Texto

Permitir a los estudiantes, profesores y analistas de sistemas de TI desarrollar software que utilice estos datos académicos y luego realizar proyectos de investigación, desarrollo e innovación, promoviendo la producción deconocimiento científico y tecnológico innovador

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

Una funcionalidad transversal que necesariamente debe incorporarse al API antes de su disponibilidad real para la comunidad académica esun mecanismo de autorización, ya que es inconcebible desde el puntode vista de la seguridad, en cuanto a la seguridad de la información por parte de los desarrolladores que utilizan la API, está destinado a proporcionar unentorno de aprobación con una base de datos de prueba antes de implementarla en producción

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

La data se ha convertido en una materia prima para la toma de decisiones, la data está ahí y es mediante la implementación de una API REST que se puede organizar, categorizar y recuperar aquella dataque nos genere interés a fin de poder trabajar, investigar o tomar mejores decisiones en torno a tareas que estemos realizando.

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: Demostración de la API myCBR Rest

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Kerstin, B., Bjørn Magnus, M., & Jaiswal, A. (2019). Demonstrating the myCBR Rest API (13.ª ed., Vol. 5).http://ceur-ws.org/Vol-2567/paper13.pdf

1.2 Objetivo del autor / Texto

Permitir a los usuarios modelar un sistema CBR usando el banco detrabajo de myCBR y luego poder llevarlo a la implementación de la aplicación como un servicio web

1.3 Partes / estructura del texto

Las herramientas de razonamiento basado en casos (CBR) son importantes para reducir el esfuerzo de desarrollar sistemas CBR.
myCBR ha sido una herramienta para investigadores y profesionales durante los últimos diez años proporcionando un sistema CBR bloques de construcción y funcionalidad a través de myCBR-SDK y medios para desarrollar modelos CBR en myCBR-workbench. En estepapel nosotros presentamos la API myCBR Rest que expone la funcionalidad de ambos myCBR-SDK y myCBR-workbench a través deuna API RESTful. Incluye la funcionalidad myCBR-SDK para permitir a los investigadores el desarrollo rápido y la experimentación de aplicaciones CBR no solo de Java, sino del lenguaje de programación elegido por los desarrolladores. La mayoría de
la funcionalidad myCBR-workbench también se ha expuesto en la misma moda que permite a los usuarios crear, modificar y eliminarmediante programación

1.4 Hipótesis o idea central del Autor / Texto

La API Rest se ha implementado utilizando Spring Boot Framework y configurado para exponer su documentación a través de una herramienta llamada swagger. Desde swagger, un desarrollador puede usar la documentación interactiva para probar solicitudes y desarrollo de aplicaciones.
Demostración de cómo usar la API Rest para obtener informaciónsobre la representación del caso, llevar a cabo la recuperación y cómo evaluar los resultados obtenidos

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

La API myCBR Rest logra crear servicios CBR que se puedenimplementar y utilizar en cualquier lugar y permite una evaluación sistemática de los sistemas CBR.
MyCBR Rest API proporciona un marco flexible para crearsistemas CBR y desarrollar nuevos componentes.

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

El desarrollo en la creación de servicio de la API myCBR proporciona un eje más preciso hacia nuevos cambios que puedan ser introducidos, esto proporciona una flexibilidad y mejoras significativas en los sistemas nuevos o que ya se encuentrandesarrollados.

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: Aplicación móvil que accede a la salud de los pacientes registrados a través de una API REST

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Francois, A., Wan, L., & Nicholson, D. (2011). A MOBILE APPLICATION ACCESSING PATIENTS' HEALTH RECORDS THROUGH A REST API. scitepres, 27-32. https://doi.org/10.5220/0003129600270032

1.2 Objetivo del autor / Texto

Acceder a la salud de los pacientes registrados a través de una API REST

1.3 Partes / estructura del texto

Los dispositivos móviles ofrecen nuevas formas para que los usuariosaccedan a los datos y servicios de atención médica de una manera segura y fácil de usar. Estas nuevas aplicaciones deben ser fáciles de crear, implementar, probar y mantener, y deben confiar en una infraestructura escalable y de fácil integración. En este trabajo presentamos las motivaciones y técnicas, opciones para crear una APIREST integrada con una aplicación móvil (iPhone / iPad) que ofrecen alos médicos, acceso a los registros médicos de sus pacientes a través de un intercambio de información de salud comunitario, regional o estatal Describimos la arquitectura del sistema, incluida la forma en queabordamos las preocupaciones de seguridad y privacidad,
las operaciones de la API REST y el formato de datos del subconjuntoHL7 utilizado para los resultados de laboratorio y las observaciones. También explicamos por qué el uso temprano de pruebas unitarias y pruebas de integración fue esencial para el éxito del proyecto.

1.4 Hipótesis o idea central del Autor / Texto

Con REST poder construir URLs dinámicamente para representar objetos de registros sanitarios remotos como sea necesario. La aplicación móvil envía solicitudes HTTP a través de Secure Sockets Layer (SSL) para obtener una notación de objetos JavaScript (JSON) del paciente, información demográfica (fecha de nacimiento,

nombre,etc.) o registros de salud.

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

- Es diseño y desarrollo al implementar la API REST fue unéxito total, si bien se enfatiza bastante en las pruebas de integración y unitarias será la retroalimentación que nos faciliten los usuarios durante las próximas semanas crucialpara seguir mejorando y escalando la aplicación
- Las API REST son especialmente adecuadas para acoplamientos rápidos e integración de soluciones como aplicaciones móviles, pero también se puede utilizar en el cuidado de la salud para aplicaciones de portal y mashup.

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

La incursión de la tecnología en el eje de salud ha dado pasos gigantescos en las últimas décadas, siendo un motor de ayuda para los pacientes y médicos, la idea de poder acceder a la historia clínica y información de un paciente desde un dispositivo móvil no solo ayudaa tomar mejores decisiones si no que ayuda a mejorar el trato que tienen los pacientes todos los días y permite una contextualización más general del caso que se está tratando

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: A Springboard to Machine Learning for All Ages

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Tanmay Bakshi. Tanmay Teaches Julia for Beginners: A Springboard to Machine Learning for All Ages. PackageManagement, Chapter (McGraw-Hill Education: New York, Chicago, San Francisco, Athens, London, Madrid, Mexico City, Milan, New Delhi, Singapore, Sydney, Toronto, 2020). https://biblio.uptc.edu.co:2164/content/book/9781260456639/chapter/chapter7

1.2 Objetivo del autor / Texto

ilustra cómo instalar y usar paquetes para entender aprofundidad los procesos API REST

1.3 Partes / estructura del texto

El artículo centra su interés en el significado de API (interfaz de programación de aplicaciones) y REST (Transferencia de estado representacional) y mediante un ejemplo de trivialidad que usa unpaquete que permite llamar una API REST.

1.4 Hipótesis o idea central del Autor / Texto

Precisar el significado de API REST y cómo funciona viéndolo desde lasimplicidad de mostrar número aleatorio.

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)
Una API es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software. Un servicio REST no posee estado, lo que implica que, entre dos llamadas, el servicio pierde todos sus datos. REST es considerada una técnica de arquitectura de software.
1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)
La noción de API REST aplicada al campo de Machine Learning a través deflask RESTful resultó ser un modelo multinomial, los ejercicios como el dela trivialidad permite entender cómo funcionan los micro servicios y como mantienen el núcleo simple pero extensible.

Universidad Pedagógica y Tecnológica de Colombia **Seccional Sogamoso**
Nombre: Implementación de patrones de diseño en arquitectura Api Rest
Parte 1. Ficha de lectura
1. Referencia bibliográfica (APA 2020)
Rodrigo Sánchez Peregrino, Miguel Pérez Vasconcelos, Rosa Gómez Domínguez, Eutimio Sosa Silva, FidelioCastillo Romero, Implementación de patrones de diseño en arquitectura Api Rest (Tecnológico nacional de México, Villahermosa, 2007).https://iydt.files.wordpress.com/2020/06/2_7_implementacic3b3n-de-patrones-de-disec3b1o-en-arquitectura-api-rest.pdf
1.2 Objetivo del autor / Texto
Implementar patrones de diseño podremos tener una mejor organización de los componentes en un desarrollo
1.3 Partes / estructura del texto
En la actualidad arquitectura api rest es una de las más utilizadas en el mundo del desarrollo de software es por eso por lo que se debe implementar un patrón de diseño para poder manipular de manera óptima cada uno de los componentes que este contiene. Este documento presenta un ejemplo de implementación de un patrón de diseño personalizado en arquitectura api rest.

1.4 Hipótesis o idea central del Autor / Texto
Implementación de patrones de diseño en arquitectura Api Rest
1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)
• La primera capa de la arquitectura API REST es la base de datos, es ahí donde nosotros guardamos nuestros datos de una aplicación. • La segunda capa dentro de la arquitectura es donde se construye toda la lógica de las Api 's, para ello se debe implementar un patrón de diseño personalizado que se ajuste a las necesidades del proyecto. • Como capa final está el desarrollo de la aplicación web que es donde se consumirá las Api´s desarrolladas y se podrá manipularcada uno de los recursos.
1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)
Se puede deducir que al implementar patrones de diseño podremos tener una mejor organización de los componentes en un desarrollo, así como identificar de manera más puntual los recursos y disminuir los costos de mantenimiento.

Universidad Pedagógica y Tecnológica de Colombia **Seccional Sogamoso**
Nombre: Patrones de diseño de la API REST para la API SDN Northbound
Parte 1. Ficha de lectura
1. Referencia bibliográfica (APA 2020)
Wei Zhou, Li Apis, Min Luo, Wu Chou, Patrones de diseño de la API REST para la API SDN Northbound, (2014).https://ieeexplore.ieee.org/abstract/document/6844664/keywords#keywords
1.2 Objetivo del autor / Texto
Cómo se puede diseñar un protocolo de red de una manera verdaderamente RESTful, convirtiéndolo en una redde datos orientada a servicios.
1.3 Partes / estructura del texto
El estilo arquitectónico REST gana cada vez más popularidad en el diseño de protocolos de red y se ha convertido en una opción predominante parala API de redes definidas por software (SDN) en dirección norte. El artículo aborda muchos problemas críticos en el diseño de protocolos de red

RESTful y presenta un marco sobre cómo se puede diseñar un protocolo de red de una manera verdaderamente RESTful, convirtiéndolo en una redde datos orientada a servicios.

1.4 Hipótesis o idea central del Autor / Texto

REST muestra cómo aplicar protocolos de red para solucionar problemas de diseño REST en algunas API. También demuestran cómo diseñar un API RESTful de SDN en dirección norte en el contexto de OpenStack

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

- La red definida por software (SDN) desacopla los planos de datos y de control, en los que un controlador lógicamente centralizado controla los comportamientos de la red en función de la información de la red global a través de varios elementos de red.
- Para que una API se considere RESTful su arquitectura cliente-servidor debe estar compuesta de clientes, servidores y recursos con la gestión de solicitudes a través de HTTP.
- Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro; POST (crear), GET (leer y consultar), PUT (editar) y DELETE (borrar).
- La URI es el identificador único de cada recurso de un sistema REST. Esta, nos facilita el acceso a la información, para poder modificarla o borrarla.

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

Este artículo es muy interesante puesto que expone cómo diseñar un API RESTful de SDN en el contexto de OpenStack haciendo uso de un controlador SDN generalizado y al final se evidencia los beneficios de diseñar un protocolo de red convirtiéndola en una red de datos orientada a servicios

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: Buenas prácticas para el diseño de una API RESTful pragmática

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Armentano, L. (2017). *Buenas prácticas para el diseño de una API RESTful pragmática*. elbaul. https://elbauldelprogramador.com/buenas-practicas-para-el-diseno-de-una-api-restful-pragmatica/

1.2 Objetivo del autor / Texto

Describir las mejores prácticas para una API pragmática diseñada para aplicaciones web basándose en requisitos como: estándares web, responsive design, consistencia, flexibilidad, eficiencia

1.3 Partes / estructura del texto

Tu modelo de datos ha empezado a estabilizarse y es el momento de crear una API pública para tu aplicación web. Te das cuenta de que es difícil hacer cambios significativos a tu API una vez que fue liberada, quieres lo mejor y lo antes posible. Ahora, en internet no escasean opiniones sobre diseño de APIs. Pero, debido a que no hay un standard adoptado popularmente que funcione en todos los casos, te quedas con un manojo de opciones: ¿Qué formatos deberías aceptar? ¿Cómo deberías autenticar? ¿Debería tu API ser versionada? Diseñando una API para SupportFu (una alternativa para Zendesk), intenté encontrar respuestas pragmáticas a estas preguntas. Mi objetivo para la API de SupportFu es que sea fácil de usar, fácil de adoptar y lo suficientemente flexible para implementarla en nuestras propias interfaces de usuario.

1.4 Hipótesis o idea central del Autor / Texto

Enfocar el esfuerzo en implementar un API REST no solamente funcional sino en cambio amigable y con todas las virtudes y buenas prácticas.

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

- Usa acciones y URLs RESTful
- SSL todo el tiempo
- Documentación de acceso público para integración
- Versionado
- Filtrado, ordenación y búsqueda en los resultados
- El consumidor de la API no siempre necesita la representacióncompleta de un recurso
- Para prevenir abusos, una buena práctica es limitar el tráfico dela API mediante el protocolo HTTP 429

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

Es un artículo muy detallado, que expone cada limitación y las consecuencias de las malas prácticas en la implementación y despliegue de un API RESTful.

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: Arquitectura De micro-servicios habilitando APIs

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Rocha, Rafael, Arquitectura de Microservicios habilitando APIs, 3 de enero de 2018. https://es.sensedia.com/es-posts-archive/arquitectura-de-microservicios-habilitando-apis

1.2 Objetivo del autor / Texto

Utilizar arquitectura de microservicios permite el desarrollo de interfaces RESTful

1.3 Partes / estructura del texto

Este modelo de arquitectura es un enfoque que determina el despliegue de aplicaciones en una gama de servicios. La arquitectura de micro servicios aumenta también la escalabilidad por permitir el uso de enfoques como auto-scaling y micro-container. APIs y micro servicios están muy relacionados, pues el estándar es utilizar una interfaz de API´s totalmente dirigida a RESTful.

1.4 Hipótesis o idea central del Autor / Texto

Utilizar arquitectura de microservicios permite el desarrollo de interfaces RESTful que expondrán su legado que nativamente no cuenta con interfaz HTTP, pero el primer desafío es elegir las herramientas correctas para esa implementación.

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

- Aplicaciones monolíticas para web usando una base de datos única
- La mayoría de los sistemas no tienen protocolos como WebServices o interfaces RESTful
- La iniciación de la aplicación debe ser rápida para crear rápidamente nuevas instancias de contenedores o servidores.
- Cuando implementamos API's usando micro servicios la capacidad de integración con sistemas legados es crucial

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

su principal objetivo es la exposición de API´s RESTful de forma consistente cumpliendo los requisitos funcionales y no funcionales, es congruente la información y al final la intención es encontrar plataformas integradas que logren permitir la gestión total de sus microservicios y de las API's

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: API REST PARA EL RECONOCIMIENTO FACIAL DE EMOCIONES (FER REST API)

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

López Mollinedo, D. (2019). API REST PARA EL RECONOCIMIENTO FACIAL DE EMOCIONES (FER REST API). Santa Clara, 1-154. https://dspace.uclv.edu.cu/handle/123456789/12159

1.2 Objetivo del autor / Texto

Reconocimiento de emociones.

1.3 Partes / estructura del texto

En la Universidad Central “Marta Abreu” de Las Villas se creó una biblioteca para el reconocimiento facial de emociones llamada Feel UCLV. Para acceder desde otras aplicaciones a dicha biblioteca se necesita desarrollar un entorno de comunicación. Con el fin de simplificar la programación de métodos y funciones que permitan reconocer las emociones expresadas en el rostro humano para ser empleados en diferentes aplicaciones se decidió la creación de una API REST. Esta permite la detección automática de emociones facialesen tiempo real en aplicaciones de Computación Afectiva. Reconoce las seis emociones universales: alegría, tristeza, miedo, ira, sorpresa yasco. Facilita el uso sin que el desarrollador tenga dominio de este tema. Está implementado en el lenguaje de programación Python mediante el framework Fast API. Facial Emotions Recognition REST API incluye para la clasificación Red Neuronal Perceptrón Multicapa, Maquina de Soporte Vectorial, Vecinos más Cercanos, Bosque Aleatorio, Árbol de Decisión, y Naive Bayes. Para evaluar los clasificadores utiliza los conjuntos de datos Cohn-Kanade y KDEF alcanzando resultados satisfactorios.

1.4 Hipótesis o idea central del Autor / Texto

Buscar que se pueda acceder con facilidad a la biblioteca UCLV que está creada en Python pero que aún no ha tenido implementación por lo que busca implementar una API con un framework de Python para comunicarse con esta biblioteca y otrasaplicaciones diseñadas en cualquier lenguaje de programación

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

Para el diseño e implementación de la API REST se utilizaron una seriede recursos de softwares necesarios en cada una de las etapas, entre estos programas están, Visual Paradigm esta ayuda a gestionar códigoy documentación por lo que permite
diseñar especificaciones Swagger. PyCharm (es una IDE) para programar en Python, Postman permite crear peticiones a las API de manera sencilla Fer Rest Api en la que logra independencia para la comunicación entre diferentes aplicaciones. Con lo investigadoplantea su diseño y diagramas como el UML y el de componentes del software Weka 3.9.3. Se extrajeron los vectores de características de todas las imágenes del conjunto de datos Cohn Kanade (Lucey et al., 2010) y se guardan en un archivo ARFF (Attribute Relation File Format). Este archivo es cargado por Weka para realizar las pruebas. Desde Weka son ajustados los parámetros de los clasificadores, para optimizar los algoritmos implementados con Scikit-learn en la implementación de la API. Y se realiza una descripción de los resultados obtenidos conFERREST API.

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

Gracias a la investigación y comportamiento de Fast Api usado como Framework puede servir de referencia como framework si necesitamos una implementación útil, ágil y sencilla de usar realizada en Python. En este cómo en muchos trabajos las garantías de los Api Rest son superiores a las opciones brindadas por SOAP por lo que los beneficios modulares y toda la integración del software en la parte de comunicación va a estar mejor implementada con una Api Rest.

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: Características de API Individualización de servicios web: REST y SOAP

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Ranga, Virender & Soni, Anshu. (2019). API Features Individualizing of Web Services: REST and SOAP. International Journal of Innovative Technology and Exploring Engineering. 8. 10.35940/ijitee. I1107.0789S19.

1.2 Objetivo del autor / Texto

Comparación entre REST y SOAP

1.3 Partes / estructura del texto

La Transferencia de estado representacional (REST) y el Protocolo simple de acceso a objetos (SOAP) son los dos principales servicios webs más utilizados hoy en días. REST se basa en un estilo arquitectónico, mientras que SOAP es un protocolo subyacente. Ambos servicios se utilizan para gestionar la comunicación en la World Wide Web (www). Ambos servicios tienen algunas ventajas einconvenientes y es decisión del desarrollador web decidir qué servicio es mejor utilizar de acuerdo con sus requisitos. El objetivo deeste trabajo de investigación es diseñar una API REST y una API SOAPmediante JAX-RS y JAX-WS, respectivamente, y brinda un análisis comparativo de las características de la Interfaz de programación de aplicaciones (API) (en términos de tiempo de respuesta, uso de memoria, velocidad de ejecución, etc.) de estos servicios mediante eluso de una herramienta de prueba API como Postman. Esto proporciona una visión de qué servicio es mejor utilizar según los requisitos.

1.4 Hipótesis o idea central del Autor / Texto

Buscar eficiencia una solución desarrollada en REST será mucho mejor que una frente a SOAP

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

La comparación del diagrama de flujo de REST y SOAP, nos indica que REST es más fácil de desarrollar que SOAP, ya que REST utiliza un estilo arquitectónico con métodos HTTP que son fáciles de entender y que los archivos SOAP WSDL. Al enviar datos con el servicio SOAP, seadjunta con el estándar SOAP siguiendo un estilo de sobre que hace que su carga útil sea más pesada. Por el contrario, REST transfiere datos tal cual sin ninguna carga útil sobre la base de la interfaz URI.

Por lo tanto, REST es liviano y requiere menos ancho de banda en comparación con SOAP, que usa más ancho de banda. En nuestro artículo de investigación, la comparación del tiempo de respuesta serealizó durante las pruebas de REST y SOAP en Postman mientras se creaban, eliminaban y realizaban consultas sobre los mensajesA partir de estos resultados, concluimos que REST tiene menos tiempo de respuesta que SOAP al realizar diferentes operaciones con servicios web. Con el análisis del tiempo de respuesta, podemos concluir que REST permite un menor uso de ancho de banda y SOAP permite un alto uso de ancho de banda, ya que tiene una carga útil deestilo envolvente y consume más recursos. Sobre la base del análisis de rendimiento, claramente que la velocidad de ejecución de REST es mucho más rápida que SOAP. El uso de memoria para los servicios web REST y SOAP y encuentra que el requisito de memoria de SOAP es mayor en comparación con REST, ya que REST usa JSON y es fácil de analizar que XML, pero SOAP usa XML y es difícil de analizar, también SOAP tiene una gran carga útil que REST. Por lo tanto, SOAP requiere más memoria y tiempo de CPU.
la misma señal SOAP en el mismo formato. SOAP admite solicitudes asincrónicas, por lo tanto, se puede utilizar en computación distribuida, mientras que REST admite el servicio punto a punto, porlo que no permite la computación distribuida. Al considerar la seguridad, el protocolo SOAP tiene WS-Security para el cifrado que hace que los datos sean más seguros.

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

Para buscar eficiencia una solución desarrollada en REST será mucho
mejor que una frente a SOAP, ya que el tiempo de respuesta, el usode memoria, su aplicación con el servidor, su forma de desarrollo, portabilidad y facilidad de uso con cualquier formato hace que realizar el proceso con la API sea más viable con Rest, aunque una característica a favor de SOAP es que con la funcionalidad de ws-security permite agregar métodos para mejorar la seguridad, depende de la necesidad del cliente la implementación de un servicio u otro.

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: Servicio Web para el control de los procesos de la Jornada Universitaria de Desarrollo Científico (JUDC) basada en la arquitectura Api Rest en el Departamento de Computación de la UNAN-Managua, en el segundo semestre del 2016.

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Aguilera Trujillo, C. A., Miranda Alvarado, G. A., & Varela Mercado, M. O. (todavía no publicado). Servicio Webpara el control de los procesos de la Jornada Universitaria de Desarrollo Científico (JUDC) basada en la arquitectura Api Rest en el Departamento de Computación de la UNAN-Managua, en el segundo semestre del2016. UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA, MANAGUA.

1.2 Objetivo del autor / Texto

control de los procesos de la Jornada Universitaria de Desarrollo Científico (JUDC) basada en la arquitectura Api Rest

1.3 Partes / estructura del texto

En la universidad Nacional Autónoma de Nicaragua, Managua UNAN-Managua se realiza cada año la celebración de la JornadaUniversitaria de Desarrollo Científico (JUDC) en sus distintas facultades. La facultad de ciencias e ingenierías posee varios departamentos en los cuales el Departamento de Computación cubre dos carreras importantes como son Ingeniería en Sistemas de Información e Ingeniería en Ciencias de la Computación. Para llevar acabo la JUDC en el Departamento de Computación, se necesita inscribir a los estudiantes antes de la fecha de realización, y la evaluación de los proyectos durante la actividad, pero estos procesosse han realizado de manera manual y se requiere de mucho tiempo. Dada esta problemática y con el aprovechamiento de recursos existentes, se presenta el desarrollo de una aplicación web para control de los procesos referente a la Jornada Universitaria de Desarrollo Científico (JUDC) en el Departamento de Computación de la UNAN-Managua, con el cual se consume y brinda servicios, y de igual manera aligera adecuadamente las actividades involucradas en
la JUDC.

1.4 Hipótesis o idea central del Autor / Texto

La aplicación se adapta a las necesidades de los usuarios, la cual es capaz de realizar de forma automatizada las actividades requeridas para la JUDC, donde se le da respuesta inmediata a los usuarios y se les presenta una interfaz interactiva.

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

Los resultados de las encuestas buscan el consenso de los estudiadospara la realizar la aplicación web para llevar a cabo la jornada universitaria de desarrollo científico JUDC En la que los encuestados conocen de los beneficios que traería estaherramienta al desarrollo exitoso de esta actividad, por lo que después realizar un análisis de presupuesto que les tomaría a los 3 líderes del proyecto llevar a cabo la orquestación del proyecto.

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

Se constató mediante el análisis de la situación actual realizada en elDepartamento de Computación de la UNAN-Managua la urgencia de automatizar y agilizar los procesos críticos de la Jornada Universitariade Desarrollo Científico (JUDC), debido a la gran relevancia que la misma representa para nuestra institución, en la cual los estudiantesdemuestran sus habilidades en diferentes campos de aplicación, aportando siempre al desarrollo del pensamiento científico.

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: Arquitectura de IoT para habilitar la intercomunicación a través de REST API y UPnPUsando IP, ZigBee y Arduino.

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

H. G. C. Ferreira, E. Dias Canedo and R. T. de Sousa, "IoT architecture to enable intercommunication through REST API and UPnP using IP, ZigBee and arduino," 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Lyon, France, 2013, pp. 53-60, doi: 10.1109/WiMOB.2013.6673340.

1.2 Objetivo del autor / Texto

permitir el fácil desarrollo de aplicaciones que puedan mejorar eficazmente la vida mediante transparencias. Interacción con personas normales. Además, se propondrán dos aplicaciones directas de estas API.

1.3 Partes / estructura del texto

La tecnología Universal Plug and Play (UPnP) define la arquitectura
para la interconexión de dispositivos. UPnP habilita una red con características omnipresentes y conectividad de igual a igual (P2P)[Utiliza protocolos estándar de Internet como IP, TCP, UDP, SOAP,XML, HTTP y otros.
Un servidor de eventos, que publica cambios en las variables de estado para los suscriptores. Los puntos de control son entidades quepueden descubrir y controlar otros dispositivos. Son entidades capaces de obtener la descripción de los dispositivos y servicios, invocar acciones para controlar los servicios y suscribirte a la fuente de eventos del servicio. Este controlador, integrado en un dispositivonormal, permite la construcción de una red UPnP omnipresente y sensible al contexto que puede intercambiar datos pertinentes para brindar un mejor servicio a los usuarios. El uso de ZigBee para interconectar dispositivos accesibles hace que la propagación y el protocolo de datos sean transparentes, seguros y garantizados.
Arduino Para traducir las solicitudes del controlador y aplicarlas a los dispositivos, se puede ampliar intuitivamente mediante la colocación de escudos, incluida la comunicación ZigBee (escudo XBee), y puede recibir y enviar señales digitales y analógicas Y una nueva arquitecturapara permitir que los dispositivos se controlan a través de IP, utilizando UPnP o RESTful API.
Presentaremos un middleware en capas que puede resolver el controly los eventos para dispositivos distribuidos. Para ser controlados por software, los dispositivos distribuidos deben poder comunicarse físicamente con otros y, para solucionarlo, 2 tipos de controladores componen esta arquitectura, el controlador maestro y los controladores esclavos. El controlador maestro es una PC Linux con un SDK UPnP y un servidor LAMP con la API responsable de la comunicación con lasaplicaciones y con los controladores esclavos. Una vez que recibe datos útiles de las aplicaciones, envía comandos de lectura y escritura

1.4 Hipótesis o idea central del Autor / Texto

Brindar una mejor calidad de vida a la humanidad a través del uso de dispositivos eléctricos y electrónicos que rodean al mundo, el control y la comunicación entre estos equipos debe ocurrir y debe ocurrir de manera automática y transparente. Este trabajo propone un modelo basado en tecnología habitual para permitir la lectura deeventos y el control de dispositivos eléctricos, ya existentes y posteriormente desarrollados, a través de una API RESTful o tecnología UPnP con el fin de permitir el fácil desarrollo de aplicaciones que puedan mejorar eficazmente la vida mediante transparencias

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

El control de dispositivos de software y la intercomunicación entre dispositivos han sido el foco de varias investigaciones recientes, que demuestran la importancia y la necesidad de una arquitectura común, simple, completa, ampliable y unificada. Los vastos circuitos de lectura y escritura permiten abarcar todos los escenarios y también permite la fácil creación de circuitos personalizados para extenderlos.La integración entre los dispositivos finales y las aplicaciones finales es transparente y usos bien conocidos de protocolos y tecnologías.

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

Está más enfocado al hardware y su aplicaciónnecesaria para la interconexión con otros SW demuestra que mediante la aplicación de APIs. Las APIS nos ayudan a conectar diferentes arquitecturas y a mejorar escenarios para la interacciónconsistemas de HW y SW.

Universidad Pedagógica y Tecnológica de Colombia
Seccional Sogamoso

Nombre: La API REST de Ensembl: datos de Ensembl para cualquier idioma

Parte 1. Ficha de lectura

1. Referencia bibliográfica (APA 2020)

Andrew Yates, Kathryn Beal, Stephen Keenan, William McLaren, Miguel Pignatelli, Graham R. S. Ritchie, MagaliRuffier, Kieron Taylor, Alessandro Vullo, Paul Flicek, The Ensembl REST API: Ensembl Data for Any Language, Bioinformatics, Volume 31, Issue 1, 1 January 2015, Pages 143–145, https://doi.org/10.1093/bioinformatics/btu613 .

1.2 Objetivo del autor / Texto

un servicio web para acceder a datos de Ensembl mediante Representational State Transfer (REST)

1.3 Partes / estructura del texto

Las llamadas a la API REST de Ensembl se basan en URL simples queespecifican tanto los datos requeridos como el formato devuelto. Por ejemplo, para solicitar la secuencia de proteínas para BRCA2 (ENSP00000439902) como documento JSON, solo es necesario ingresar la siguiente URL en un navegador web: http://rest.ensembl.org/sequence/id/ ENSP00000439902.json
Los componentes de la URL definen los datos y / o la acción deseados.Con comandos similares, los usuarios pueden recuperar características como genes, ortólogos, variantes, alineamientos genómicos y árboles genéticos o realizar acciones como convertir coordenadas entre ensamblajes, entre otras acciones. La incorporación de datos de Ensembl en cualquier análisis requiere una biblioteca HTTP y un analizador JSON. HTTPS es compatible para el acceso seguro del cliente.
A continuación, se muestra un ejemplo de una solicitud de Pythonpara imprimir el número de variantes que se superponen con BRAF (ENSG00000157764): alojar el código VEP o los archivos de caché. HTTP también ha demostrado ser un protocolo de datos más robusto en comparación con MySQL, lo que mejora la experiencia del usuario para usuarios detodo el mundo.
Se han desarrollado varias API nativas de terceros para ayudar a acceder a la API REST en lenguajes como Python, R y JavaScript, lo que demuestra la utilidad de nuestra API REST para estos lenguajes bioinformáticos cada vez más populares. Las aplicaciones JavaScript como Wasabi importan árboles genéticos Ensembl y alineaciones de secuencias múltiples del genoma a través de REST, creando un vínculoperfecto entre la herramienta y los datos.
REST ha demostrado ser un modelo sostenible para la distribución dedatos genómicos a múltiples lenguajes de programación. Planeamos ampliar la cobertura de los datos y herramientas de Ensembl alojados en él. También planeamosproporcionar más formatos desde el servicio, como la salida VCF de nuestro punto final VEP.

1.4 Hipótesis o idea central del Autor / Texto

La API REST de Ensembl se puede utilizar para consultar las herramientas y los recursos de datos de Ensembl desde una variedadde lenguajes de programación y permite un acceso programático flexible que antes solo era compatible con nuestra APIde Perl. Los costos de configuración reducidos para un cliente significan que los usuarios pueden interactuar con los últimos datos de Ensembl sin la necesidad de seguir nuestras versiones de API regulares.
El soporte de solicitudes POST para VEP permite la anotación de conjuntos de datos de variación a gran escala sin la necesidad dedescargarlos.

1.5 ¿Cuáles son las ideas que sustentan la hipótesis? (Hipótesis secundarias)

Utilizando formatos estándar como JSON y FASTA y minimizando el trabajo del cliente. También presentamos enlaces a lapopular herramienta Ensembl Variant Effect Predictor que permite el análisis de variantes programáticas a gran escala independientemente de cualquier lenguaje de programación específico.

1.6 Conclusiones centrales del Autor / Texto (Derivadas de las hipótesis)

La comunicación y el uso de las herramientas de APIRest son mayormente usadas en lenguajes cómo Python, pero pueden ser evaluadas en otros lenguajes y en favor de esta, puede encontrar másdocumentación a comparación de SOAP. Y así como los APIREST han demostrado ser sostenibles para la distribución de datos genómicosen múltiples lenguajes, así podría adaptarse a cualquier otro ámbito.

REFERENCIAS

- [1] Xanthopoulos, S. y Xinogalos, S. (2013, 1 de septiembre). *(PDF) A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications*. ResearchGate. https://www.researchgate.net/publication/258010031_A_Comparative_Analysis_of_Cross-platform_Development_Approaches_for_Mobile_Applications

- [2] Fielding, R. (s. f.). *Fielding Dissertation: CHAPTER 6: Experience and Evaluation*. Donald Bren School of Information and Computer Sciences @ University of California, Irvine. https://www.ics.uci.edu/~fielding/pubs/dissertation/evaluation.htm

- [3] Maria. (s. f.). *Características y usos de las APIs REST*. Tribalyte Technologies. https://tech.tribalyte.eu/blog-que-es-una-api-rest

- [4] *Generalidades del protocolo HTTP - HTTP | MDN*. (s. f.). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/HTTP/Overview

- [5] Bengura, J. y Angulo, A. (2010, 12 de enero). Repositorio Académico - Universidad de Chile. http://repositorio.uchile.cl/bitstream/handle/2250/104108/cf-gonzalez_ro.pdf?sequence=3&isAllowed=y

- [6] *Bases de Datos*. (2018, 1 de julio). Bases de Datos. https://biblio.uptc.edu.co:3534/eds/pdfviewer/pdfviewer?vid=3&sid=18136f44-db79-41ef-badb-21b081594a91@sessionmgr103

- [7] Arturo Barrera. (s.f.) JSON: ¿Qué es y para qué sirve? https://www.nextu.com/blog/que-es-json/

- [8]Deyimar A. (Jul 27, 2021). ¿Qué es JSON? https://www.hostinger.co/tutoriales/que-es-json

- [9]JOSE GUILLERMO VALLE (2005). Arquitectura Cliente Servidor. https://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml

- [10]Oscar J. Iturralde (octubre 18, 2016). Introducción a Los Patrones de Diseño: Un Enfoque Práctico. https://books.google.com.co/books/about/Introducci%C3%B3n_a_Los_Patrones_de_Dise%C3%B1o.html?id=OSI7MQAACAAJ&source=kp_book_description&redir_esc=y - https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor.

- [11] Maikol Garcia (junio, 2021). ¿Qué es y para qué sirve Java Spring Boot? https://platzi.com/blog/que-es-spring-boot/?utm_source=google&utm_medium=cpc&utm_campaign=12915366154&utm_adgr

oup=&utm_content=&gclid=Cj0KCQjw1dGJBhD4ARIsANb6OdnYhxHgcUxDKi1VICJYynnNPlplIzkcPBlo9QZp7JfLc9ENuW7fw6kaAsPnEALw_wcB&gclsrc=aw.ds

- [12]Cecilio Álvarez Caules (marzo 8, 2016). ¿Qué es Spring Boot?
- https://www.arquitecturajava.com/que-es-spring-boot/

- [13]Diego Lázaro. (2018). Códigos de estado HTTP. https://diego.com.es/codigos-de-estado-http.

- [14] Red Hat. (S.F). Qué son las API y para qué sirven. https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

- [15] VÍCTOR CUERVO (FEBRERO 14, 2019). ¿Qué es Postman? https://www.arquitectoit.com/postman/que-es-postman/

- [16] MongoDB. (S.F.). ¿Qué es una base de datos NoSQL? https://www.mongodb.com/es/nosql-explained

- [17] *Generación automática de API REST a partir de API Java, basada en transformación de Modelos (MDD)*. (s. f.). SEDICI - Repositorio de la Universidad Nacional de La Plata. http://sedici.unlp.edu.ar/bitstream/handle/10915/67777/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

- [18] Vijay, S. (s. f.). *REST API Modeling Languages -A Developer ' s Perspective*. https://d1wqtxts1xzle7.cloudfront.net/47318910/IJSTEV2I10199-with-cover-page-v2.pdf?Expires=1630892977&Signature=ZZkUN25F3fFmP6MYSRALtd6tuaOw-TyhrTT0vyRYRBwDXrSoO6HdZNEf4Ew94GoCZIdZ1i-frRKwxebvlfxKrb1uINI1oLuU4yXuIgk0TK-7ZNol3KMv1bOFWs0V~TctQxyI3stuFNsvbkgu50ydPsbDdQ0b8k3CHR5SpFjM4inHF3ifCsMRgtFoM9srx3XbakEiHEhFNqWhpBVR4l9rw22t9y1FqwWdNdwHdqiuk5B3eqisp2BPRvNcRhA9pGB8wCPVM2zV2ifTZzdPBhSU1wrax7CRyhisCa36RHpJCyrK7nQ-d3tDOBcIoSsGufcQcmnOmJV7Z8xS401i9VOXSw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

- [19] *API features individualizing of web services: REST and SOAP*. (s. f.). ResearchGate | Find and share research. https://www.researchgate.net/profile/Virender-Ranga/publication/335419384_API_Features_Individualizing_of_Web_Services_REST_and_SOAP/links/5d64960ea6fdccc32cd31171/API-Features-Individualizing-of-Web-Services-REST-and-SOAP.pdf

- [20] Garcia Rodriguez. (s. f.). *DEFINICIÓN E IMPLEMENTACIÓN DE UNA API REST PARA SISTEMAS DE RECOMENDACIÓN*. Biblos-e Archivo. https://repositorio.uam.es/bitstream/handle/10486/688292/garcía_rodríguez_iván_tfg.pdf?sequence=1&isAllowed=y

- [21]quichimbo, J. (s. f.). *Bases de Datos*. Bases de Datos. https://biblio.uptc.edu.co:3442/eds/pdfviewer/pdfviewer?vid=3&sid=15dbb552-fd08-4b65-8c52-ab313da2781b@sessionmgr101

- Rantanen, P. (2017). REST API Example Generation Using Javadoc. Pori Department, Tampere University of Technology, Pori, Finland, 14, 466-477. https://doi.org/10.2298/CSIS161022009R

- Hyuk Park, J., Young-Sik, J., & Cho, M. (2013). Improving Performance through REST Open API Grouping for Wireless Sensor Network. International Jornal o Distributor Sensor Networks, 2013, 1-13. https://doi.org/10.1155/2013/958241

- Ferreira, A. S., Nicacio, J. M., Ferreira, G. V., Santos Filho, G. M., & Rodrigues, V. S. (2018). Desenvolvimento de uma API REST para um Sistema Acadêmico de Terceiros. Revista de Sistemas e Computação, Salvador, 8(2), 536-546. http://biblio.uptc.edu.co:2304/ehost/detail/detail?vid=0&sid=d6a10824-9759-4af2-a589-6db039d1d973%40sdc-v-sessmgr03&bdata=Jmxhbmc9ZXMmc2l0ZT1laG9zdC1saXZl#db=iih&AN=134691613

- Kerstin, B., Bjørn Magnus, M., & Jaiswal, A. (2019). Demonstrating the myCBR Rest API (13.a ed., Vol. 5). http://ceur-ws.org/Vol-2567/paper13.pdf

- Francois, A., Wan, L., & Nicholson, D. (2011). A MOBILE APPLICATION ACCESSING PATIENTS' HEALTH RECORDS THROUGH A REST API. scitepres, 27-32. https://doi.org/10.5220/0003129600270032