



Tablas de Investigación conceptos básicos de la API web

Autor: Jimmis J. Simanca

Tecnología en Desarrollo de Software, Corporación Universitaria Uniremington

Asignatura: Lenguaje de Programación 3

Director: Milton Javier Mateus Hernández

22 de octubre de 2024




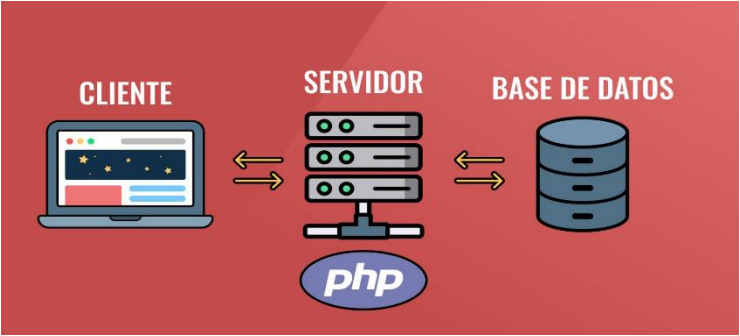
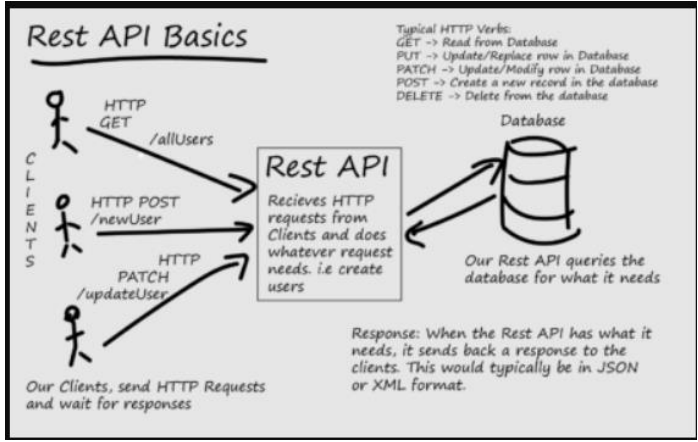
UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996


Tabla de contenido


Tabla 1	3
2. Tabla 2	7
Referencias.....	11

Tabla 1

Concepto	Descripción	Ejemplos o usos
HTML (HyperText Markup Language)	(HTML)Es un lenguaje de marcado de Hipertexto que sirve para darle la estructura y desplegar una página web, el contenido puede ser párrafos, viñetas, imágenes, texto, tablas etc. (mdn web docs_, s.f.)	<pre><h1>Mi primera página</h1> <p>Mi primera publicación en la página</p></pre>
Frontend	Es la parte visual que interactúa con el usuario mostrándole botones tablas imágenes entre otros brindando una interfaz de usuario hermosa y atractiva.	<p>El Frontend sirve para realizar la interfaz de un sitio web, desde su estructura hasta los estilos, como pueden ser la definición de los colores, texturas, tipografías, secciones, entre otros. (Copola, s.f., párr 6)</p>  <p>Imagen tomada de: frontendbackend.jpeg (509x339)</p>

Backend	<p>Es la parte lógica del software este principalmente se encarga de la lógica de negocio, recibe y envía datos a las app y sitios web entre otros.</p>	<p>El backend son todos los códigos ocultos que sirven para que una página web o aplicación funcione correctamente. (Copola, s.f., párr 24)</p>  <p>Imagen tomada de: a04ca961-f61d-4dc3-837c-55c06df42ce7.png (1921x1080)</p>
REST	<p>La arquitectura de software REST es esencial para desarrollar aplicaciones eficientes y escalables. Se basa en la representación de recursos y utiliza solicitudes HTTP para la comunicación, lo que facilita la interoperabilidad entre sistemas distribuidos. Esto permite a las aplicaciones funcionar de manera más efectiva y conectarse fácilmente con otros sistemas. (BYRON VARGAS, 2024)</p>	<p>REST (Representational State Transfer) es una arquitectura de software que utiliza el protocolo HTTP para la creación, modificación y eliminación de datos. (MPDajedrez, s.f.)</p>  <p>Imagen tomada de: api-rest62.png (450x282)</p>

Framework	Es una herramienta de que facilita y agiliza el desarrollo de software a los programadores ya que trae toda lógica incluida para muchos problemas que ya han sido resueltos y testeados por la comunidad este puede incluir patrones de diseño, módulos, librerías y código estándar (programacion & data, 2023)	Spring, Django, laravel, Angular, vue.js, electron, Flutter etc.
Interfaz	En el mundo IT una interfaz es la manera como interconectamos recursos de software facilitando el intercambio de información y existen diferentes tipos como las interfaces lógicas, interfaz de usuario o interfaces físicas (concepto, s.f.)	<p>Una interfaz gráfica brinda un entorno de trabajo mucho más</p> <p>Interfaz gráfica</p>  <p>amigable</p> <p>Fuente: https://concepto.de/interfaz/#ixzz8pL6m2M3c</p>
API (Application Programming Interface)	Es un conjunto de reglas predefinidas que permiten a varias apps que se comuniquen entre sí ya sea una app web, móvil, de escritorio e incluso para el internet de las cosas(iot)	El api de mercado pago que sirve para que puedan hacer pagos desde cualquier banco. www.mercadopago.com.mx
Servidor Web	Es un pc con una gran capacidad que sirve para almacenar, procesar y distribuir información o archivos mediante protocolos de internet como (http, smtp) entre otros. (V, 2024)	Apache. Lighttpd, nginx

Base de Datos	Es un conjunto de información que se almacena para después poder usarlo y manipularlos dese app móviles o web etc (concepto, s.f.)	MySQL, MongoDB, SQL server, PostgreSQL
Bug	Bug(bicho)es el lenguaje que se utiliza para nombrar los errores que se presentan en un software	Según (definicion de, s.f.) algunos bugs habituales son la inclusión de variables que no fueron inicializadas en el momento preciso, la mala indexación de las tablas en una base de datos, la creación de un bucle infinito ,
Cookies	Pequeños archivos de texto que guardan datos como búsqueda en internet un nombre de usuario etc (kaspersky, s.f.)	Inicio de sesión: Las cookies pueden almacenar tokens de sesión para que los usuarios no tengan que volver a ingresar sus credenciales.
Suite de pruebas	Es la representación de un grupo de pruebas que se pueden repetir en varios entornos o el mismo (IBM, 2021)	Ejemplo de pruebas de api, prueba de interfaz de usuario, prueba de seguridad, pruebas de bases de datos, entre otros
Tipos de Pruebas	Existen una gran variedad de tipos de prueba para validar correcto funcionamiento del software estos son algunos: pruebas de humo, pruebas de rendimientos, pruebas unitarias, pruebas funcionales, pruebas de casos de uso entre otros	<p>Ejemplos de tipos de Pruebas No Funcionales</p>  <p>Imagen tomada de: maxresdefault.jpg (1280x720)</p>

2. Tabla 2

Componente	Función	Métodos HTTP	Descripción y Ejemplo
Endpoint	se refiere a una URL específica a la que los clientes pueden enviar solicitudes HTTP para realizar operaciones sobre recursos específicos.	Se utiliza para realizar operación CRUD con los métodos http como GET, POST, PUT y DELETE,	<p>Es como los usuarios pueden interactuar con el software enviando solicitudes HTTP utilizar los recursos específicos del programa</p> <pre> @GetMapping // @PreAuthorize("isAuthenticated()") public ResponseEntity<Page<DatosListadoTopico>> listadoTopicos(@PageableDefault(size = 5) Pageable paginacion) { return ResponseEntity.ok(topicoRepository.findBy(paginacion).map(DatosListadoTopico::new)); } @PostMapping // @PreAuthorize("hasRole('ADMIN')") public ResponseEntity<DatosRespostaTopico> guardarTopico(@RequestBody @Valid DatosRegistroTopicos datosRegistroTopicos, UriComponentsBuilder uriComponentsBuilder) { Curso curso = cursoRepository.findById(datosRegistroTopicos.idCurso()).orElseThrow(); Usuario usuario = usuarioRepository.findById(datosRegistroTopicos.idUsuario()).orElseThrow(); // Asignar el curso al DTO datosRegistroTopicos = new DatosRegistroTopicos(datosRegistroTopicos.titulo(), datosRegistroTopicos.mensaje(), datosRegistroTopicos.status(), datosRegistroTopicos.idUsuario(), usuario, datosRegistroTopicos.idCurso(), curso); Topico topico = topicoRepository.save(new Topico(datosRegistroTopicos)); DatosRespostaTopico datosRespostaTopico = new DatosRespostaTopico(topico.getId(), topico.getTitulo(), topico.getMensaje(), topico.getFechaCreacion(), topico.getStatus(), topico.getCurso() != null ? new DatosCurso(topico.getCurso().getId(), topico.getCurso().getNombre(), topico.getCurso().getCategoria()) : null // valido si no tiene curso asignado devuelve null , topico.getAutor() != null ? new DatosUsuario(topico.getAutor().getId(), topico.getAutor().getNombre()) : null); URI url = uriComponentsBuilder.path("/{topico/{id}}").buildAndExpand(topico.getId()).toUri(); return ResponseEntity.created(url).body(datosRespostaTopico); } </pre>
Métodos HTTP	GET	Se utiliza para realizar peticiones al servidor y darnos los recursos solicitados	<pre> @GetMapping no usages 2023/09/07 10:00:00 // @PreAuthorize("isAuthenticated()") public ResponseEntity<Page<DatosListadoTopico>> listadoTopicos(@PageableDefault(size = 5) Pageable paginacion) { return ResponseEntity.ok(topicoRepository.findBy(paginacion).map(DatosListadoTopico::new)); } </pre>

	POST	Se utiliza para enviar recursos al servidor	<pre> @PostMapping("no usages") // @PreAuthorize("hasRole('ADMIN')") public ResponseEntity<DatosRespostaTopico> guardarTopico(@RequestBody @Valid DatosRegistroTopicos datosRegistroTopicos, UriComponentsBuilder uriComponentsBuilder) { Curso curso = cursoRepository.findById(datosRegistroTopicos.idCurso()).orElseThrow(); Usuario usuario = usuarioRepository.findById(datosRegistroTopicos.idUsuario()).orElseThrow(); // Asignar el curso al DTO datosRegistroTopicos = new DatosRegistroTopicos(datosRegistroTopicos.titulo(), datosRegistroTopicos.mensaje(), datosRegistroTopicos.status(), datosRegistroTopicos.idUsuario(), usuario, datosRegistroTopicos.idCurso(), curso); Topico topico = topicoRepository.save(new Topico(datosRegistroTopicos)); </pre>
	PUTT	Se utiliza para modificar recursos del servidor	<pre> @PutMapping("no usages") @Transactional public ResponseEntity actualizarTopico(@RequestBody @Valid DatosActualizarTopico datosActualizarTopico) { Topico topico = topicoRepository.getReferenceById(datosActualizarTopico.id()); topico.actualizarTopico(datosActualizarTopico); return ResponseEntity.ok(new DatosRespostaTopico(topico.getId(), topico.getTitulo(), topico.getMensaje(), topico.getFechaCreacion(), topico.getStatus(), topico.getCurso() != null ? new DatosCurso(topico.getCurso().getId(), topico.getCurso().getNombre(), topico.getCurso().getCategoria()) : null // valida si no tiene curso asignado devuelve null)); topico.getAutor() != null ? new DatosUsuario(topico.getAutor().getId(), topico.getAutor().getNombre()) : null); } </pre>
	DELETE	Se utiliza eliminar recursos del servidor	<pre> @DeleteMapping("/{id}") no usages public ResponseEntity eliminarTopico(@PathVariable Long id) { topicoRepository.getReferenceById(id); topicoRepository.deleteById(id); return ResponseEntity.ok().build(); } </pre>
Headers	Permite enviar información adicional junto con las peticiones HTTP	N/A	De acuerdo con (mdn web docs, s.f.) Una cabecera de petición está compuesta por su nombre (no sensible a las mayúsculas) seguido de dos puntos ':', y están divididas o agrupadas según el contexto a utilizar

Cuerpo (Body)	En el api existe un un parámetro que se puede pasa a las peticiones HTTP por lo general se usan en el método post y el putt	N/A	<p>Este se utiliza para enviar los datos en json, en java con spring se utiliza la anotación @RequestBody este recibe los datos en json desde el Frontend y los envía al servidor</p> <pre>{ "mensaje": "Usar expresiones regulare", "idTopico": "19", "idUsuario": "1", "solucion": "debese usar algo as {/7cd//sd}" }</pre> <pre>@PostMapping no usages JimsimroDev // @PreAuthorize("hasRole('ADMIN')") public ResponseEntity<DatosRespuestaTopico> guardarTopico(@RequestBody @Valid DatosRegistroTopicos datosRegistroTopicos, UriComponentsBuilder uriComponentsBuilder) {</pre>
Códigos de Estado	Los códigos de estados no indica si la petición fallo o fue exitoso algunos códigos son 200 ok, 201 create, 202 accepted, 404 not found, 403 foriben entre otros	N/A	<ol style="list-style-type: none"> 1. Respuestas informativas (100–199), 2. Respuestas satisfactorias (200–299), 3. Redirecciones (300–399), 4. Errores de los clientes (400–499), 5. y errores de los servidores (500–599).
Pruebas Unitarias	En las pruebas de software, las pruebas unitarias son un método para probar porciones aisladas más pequeñas (o unidades) de código. Las pruebas unitarias generalmente, se realizan con scripts de automatización de pruebas en la parte más	N/A	<pre>@Test JimsimroDev public void testEliminarTopico() { Topico topico = new Topico(); when(topicoRepository.getReferenceById(1L)).thenReturn(topico); ResponseEntity response = topicosController.eliminarTopico(id: 1L); assertEquals("expected: 200, response.getStatusCodeValue()"); verify(topicoRepository, times(wantedNumberOfInvocations: 1)).getReferenceById(1L); verify(topicoRepository, times(wantedNumberOfInvocations: 1)).deleteById(1L); }</pre>

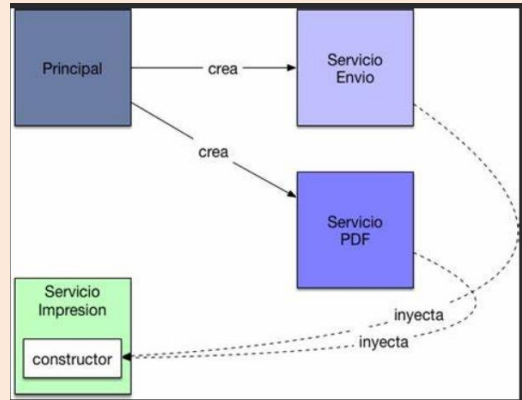
	pequeña del software que se puede probar (Chicazawa, 2021).		
Inyección de Dependencias	Es uno de los principios solid mas populares, este nos permite reemplazar fácilmente las dependencias de los objetos en tiempo de ejecución	N/A	 <p>Diagrama de inyección de dependencias:</p> <ul style="list-style-type: none">Un objeto Principal (azul) crea (crea) dos servicios: Servicio Envio (púrpura) y Servicio PDF (púrpura).El Servicio Impresion (verde) contiene un constructor (verde).Los servicios Servicio Envio y Servicio PDF inyectan (inyecta) sus dependencias al constructor del Servicio Impresion.

Imagen toma

de: https://th.bing.com/th/id/OIP.2_QyJHNBUCpHVGvdrMg2-QHaFs?rs=1&pid=ImgDetMain

Referencias

- BYRON VARGAS. (23 de 05 de 2024). Obtenido de <https://www.byronvargas.com/web/que-es-la-arquitectura-de-software-rest/#:~:text=La%20arquitectura%20de%20software%20REST%20es%20un%20estilo,utilizando%20verbos%20como%20GET%2C%20POST%2C%20PUT%20y%20DELETE.>
- concepto. (s.f.). *concepto*. Recuperado el 21 de 10 de 2024, de concepto: <https://concepto.de/base-de-datos/>
- Copola, M. (s.f.). *hubspot*. Recuperado el 21 de 10 de 2024, de hubspot: <https://blog.hubspot.es/website/frontend-y-backend#:~:text=Algunos%20ejemplos%20de%20frontend%20son%20los%20siguientes%3A%20Optimizaci%C3%B3n,Velocidad%20%28cuanto%20m%C3%A1s%20r%C3%A1pido%20cargue%20el%20sitio%2C%20mejor%29.>
- definicion de. (s.f.). Obtenido de <https://definicion.de/bug/>
- IBM. (06 de 03 de 2021). Obtenido de <https://www.ibm.com/docs/es/rtw/9.0.0?topic=perspective-test-suites-overview-tasks>
- kaspersky. (s.f.). *kaspersky*. Recuperado el 21 de 10 de 2024, de kaspersky: <https://www.kaspersky.es/resource-center/definitions/cookies>
- mdn web docs. (s.f.). Obtenido de <https://developer.mozilla.org/es/docs/Web/HTTP/Headers>
- mdn web docs_. (s.f.). *M mdn web docs*. Recuperado el 20 de 10 de 2024, de M mdn web docs: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/CSS_basics
- MPDajedrez. (s.f.). Obtenido de <https://mdpajedrez.com.ar/api-rest-definicion-usos-y-ejemplos/>
- programacion & data. (21 de 09 de 2023). *programacion & data*. Obtenido de programacion & data: <https://ebac.mx/blog/frameworks>