Clase 1 - DA2 Tecnología

@March 6, 2023

Profes:

Matias Salles

Joselen Cecilia

Link al repo de clase

https://github.com/ORT-DA2/2023.1-Tecnologia-N6A

Cronograma clase 1:

- Introduccion al curso
- Presentacion de los alumnos
- Introduccion a las API

Introducción del curso

https://docs.google.com/presentation/d/1MPDcN3HKNQna0859pDuTuDH9hc0F4vRT2PcWxL3yLAk/edit?usp=sharing

Durante el curso se construirá una aplicación completa vista desde la perspectiva de sus componentes, es decir, se va desarrollar un **back-end** con funcionalidad y base

de datos, y también, se va generar una **SPA** (Single Page Application) para que el usuario pueda utilizar las funcionalidades provistas por el servidor.

Vamos a dividir esta construcción en dos partes:

- **La primera** es la construcción de la API (Application Programming Interface) REST la cual se creará utilizando .Net Core y WebApi. Cabe hacer énfasis que en esta parte no vamos a tener una GUI de usuario, sino que haremos el servicio y lo probaremos utilizando una aplicación para ello llamada Postman.
- **La segunda** es la construcción de una SPA (aplicación web) con la que el usuario podrá utilizar el sistema. La misma se realizará en Angular.

Introduccion a las API

Que son las API y para que sirven?

Una API (Application Programming Interface) es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones.

Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber como estan implementados. Una API es el mecanismo más util para conectar dos softwares entre si para el intercambio de mensajes o datos en un formato estándar.

Las APIs otorgan:

- flexibilidad
- simplifican el diseño, la administracion y el uso de las aplicaciones
- proporcionan oportunidades de innovacion, lo cual es ideal al momento de diseñar herramientas y productos nuevos (o de gestionar los actuales).

En términos mas simples, una API es un tipo de interfaz la cual expone un conjunto de funciones que permite a desarrolladores independientes poder acceder a cierta informacion expuesta por estas funciones.

Que es una WebApi?

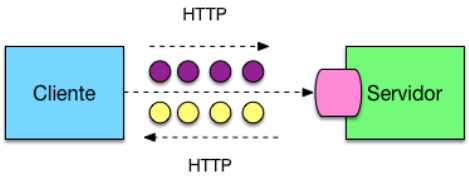
Una WebAPI es una API especifica utilizada a traves de la web.

Las ApisWeb funcionan con las limitaciones del estilo arquitectonico REST, las cuales se denominan RESTful API.

Que es REST?

REST o Representational State Transfer es un estilo de arquitectura, a la hora de realizar una comunicacion entre cliente y servidor.

esta comunicacion se puede realizar de varias formas posibles, pero hoy por hoy una de las necesidades mas claras es que esa comunicacion sea abierta y podamos acceder desde cualquier sitio sin importar el tipo de cliente que seamos. Estamos hablando de una comunicacion utilizando el protocolo HTTP (Hyper Text Transfer Protocol). Como logramos esto? En nuestro servidor vamos a tener un puerto reservado para nuestra API (ejemplo el 80) el cual cualquiera del mundo tendra acceso a la informacion que la API exponga desde cualquier lado.

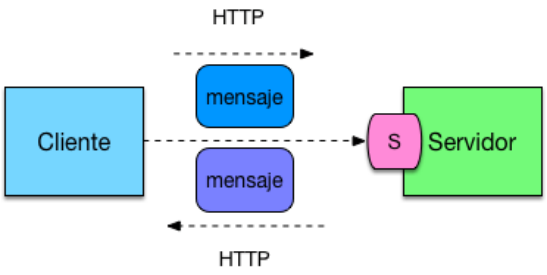


Una vez que establecimos el protocolo de comunicacion, se tiene que definir la tipologia de mensajes que se va a enviar.

En que formato van a viajar los mensajes? XML o JSON.

Una vez establecido este formato estaremos listo para el intercambio de mensajes. Una api es un servicio que provee interoperabilidad con otros servicios ya que permite el intercambio significante de informacion mediante interfaces en un contexto particular. No solo se considera la habilidad de intercambiar data (sintaxis) sino que tambien la habilidad de interpretar correctamente la data que esta siendo intercambiada (semantica).

Por defecto la serializacion de los mensajes de una API es con formato JSON.



Conceptos clave

Endpoint/Recurso

Un recurso es una URI la cual es conocida por el mundo y es el identificador para realizar diferentes acciones en la API, aplicar diferentes verbos HTTP.

Cuando una API interactúa con otro sistema, los puntos de contacto de esta comunicación se consideran **endpoints**.

Cada **endpoint** es la ubicación desde la cual las API pueden acceder a los recursos que necesitan para llevar a cabo su accion.

Las API funcionan mediante "request" y "response". Cuando una API solicita información de una aplicación web o un servidor web, recibirá una respuesta. El lugar donde las API envían solicitudes y donde vive el recurso se denomina **endpoint**.

La creacion de un endpoint deberia respetar las siguientes caracteristicas:

1. **Sustantivos ante verbos**.

```
.../dogs → URI para manipular perros
.../users → URI para manipular usuarios
.../sessions → URI para manipular sesiones
```

2. **Intuitiva y simple.** Esto es fundamental para que el uso de la API sea intuitiva y simple. Si mediante la URI se puede entender lo que hace la API sin necesitar ningun tipo de documentacion extra, sera mas simple de utilizarla.
3. **Plural ante singular**. Esto facilita la interpretacion de nuestra API, ya que queda mas intuitiva. Es importante recalcar que no se tienen que mezclar.
4. **En minuscula**
5. **Nombres concretos ante abstractos**. Los desarrolladores siempre buscan niveles de abstracciones altos, pero esto es contraproduciente en la creacion de

recursos porque tenderan a ocultar implementaciones concretas y a no saber cual es la respuesta exacta a ese recurso.

   a. Bien:

```
../dogs
../admins
```

   b. Evitar:

```
../things
../someone
```

6. **Recursos con relaciones**. Vamos a encontrar muchos recursos con relaciones hacia otro recurso y probablemente queramos obtener algo en particular de ese recurso.

   a. Bien:

```
../owners/1/dogs
```

   b. Evitar:

```
../users/1/dogs/2
```

7. **Hacer uso de ? para ocultar complejidad**.
8. **Verbos fuera de la URI**. El uso de los verbos tiene que estar explicito en la request y no en la URI. El evitar los verbos en la URI evitamos crear URIs especificas a acciones especificas las cuales para el momento sirven pero no son mantenibles.

   a. Bien:

```
../dogs?leashed=true
../dogs
```

   b. Evitar:

```
../getAllLeashedDogs
../getHungerLevel
```

Verbos HTTP

Durante el curso vamos a utilizar 4 verbos, GET, POST, PUT y DELETE, para obtener, agregar, modificar y eliminar elementos. La combinación de estas operaciones hace que REST sea CRUD.

Un mismo verbo en distintos endpoints puede significar una respuesta diferente. Ejemplo:

Resource	POST create	GET read	PUT update	DELETE delete
/dogs	Create a new dog	List dogs	Bulk update dogs	Delete all dogs
/dogs/1234	Error	Show Bo	If exists update Bo If not error	Delete Bo

El verbo Get es Idempotente no importa las veces que lo aplique, el servidor siempre me va a responder lo mismo.

En cambio los verbos POST y DELETE no lo son.

Manejo de errores

Desde la perspectiva del desarrollador que consume nuestra Web API, todo del otro lado de la interfaz es desconocido, es una caja negra. Por lo tanto errores se convierten en una herramienta para proveer un contexto y visibilidad en como usar la API.

Para el maejo de errores vamos a usar

HTTP status code

1. Todo salio bien success - 2xx

2. La aplicacion hizo algo mal - client error - 4xx
3. La API hizo algo mal - server error 5xx

Los que vamos a ver durante el curso:

```
200 - Ok
400 - Bad Request
500 - Internal Server Error
201 - Created
304 - Not Modified
404 - Not Found
401 - Unauthorized
403 - Forbidden
```

Mensajes de error

Debemos ademas de indicar el codgio de error ser explicativos en el body de la response del origen del error y como solucionarlo.

Estrucutra de la respuesta

El formato por defecto de las APIs es JSON.

Seguir la convencion de JavaScript para el nombramiento de los atributos, usar camelCase.