Visualización e Interfaces
-
Interfaces de usuario e Interacción

Visualización de datos via API

[API = Application Programming Interface]

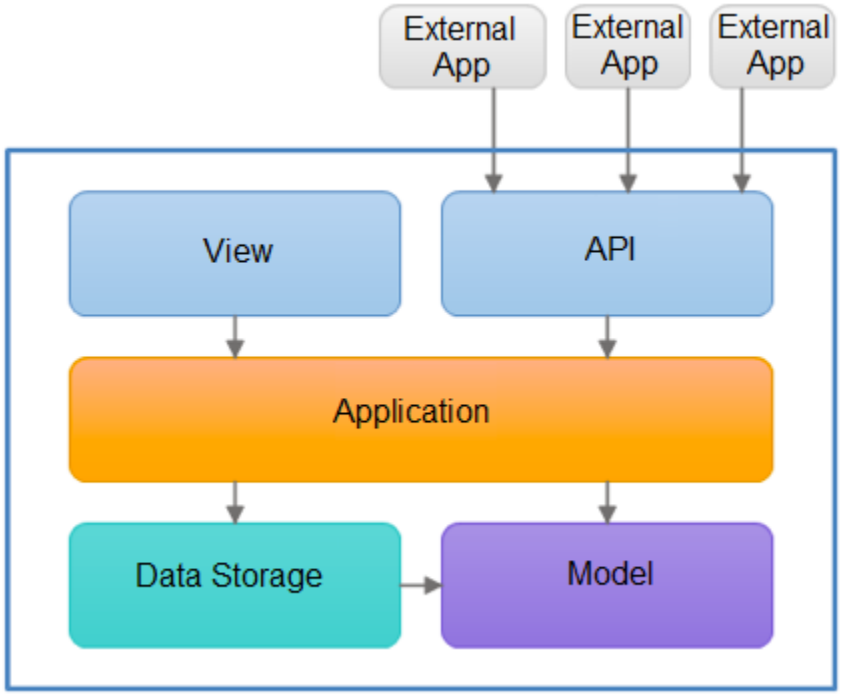
¿Qué es una API?

Beneficios de su uso

REPASO

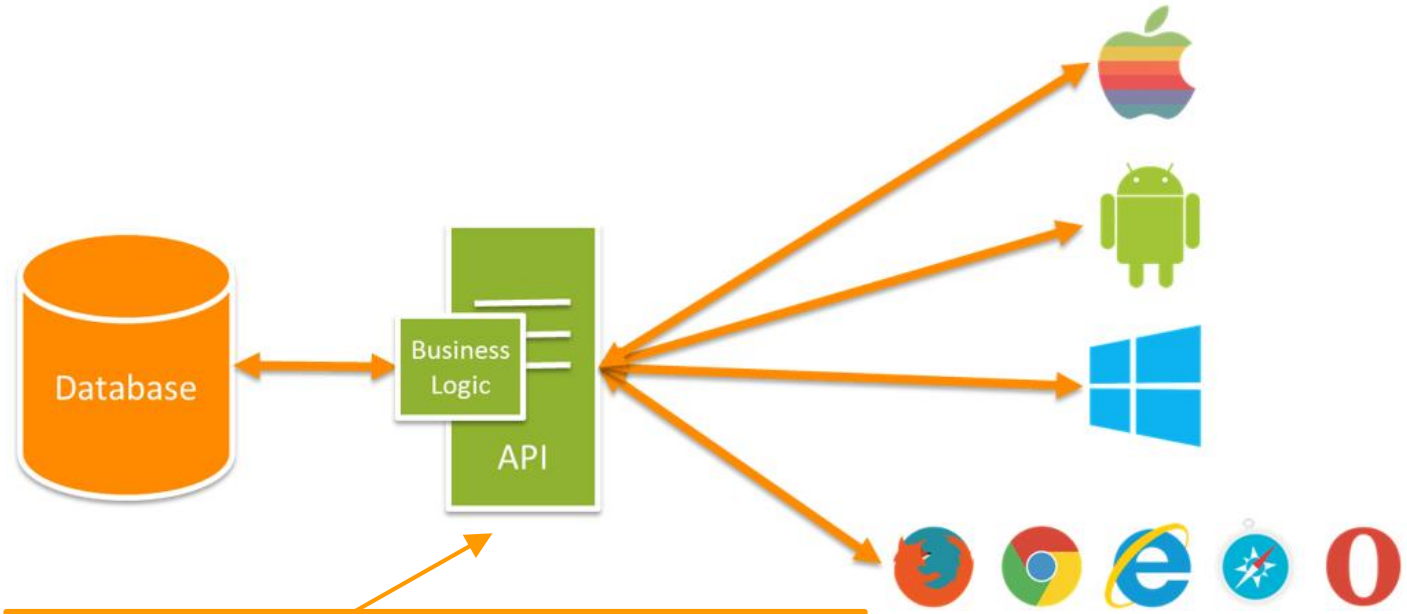
API: Application Programming Interface

- Es una interfaz.
- Permiten la utilización desde el exterior del sistema.
- Definen datos y cómo acceder a ellos.
- Implementada como:
  - Procedimientos
  - Funciones
  - Objetos y Métodos
  - Servicios web



REPASO

Un acercamiento a la arquitectura



Application Programming Interface

REPASO

Servicios web

- Ya usaron uno en Web 1 (AJAX)
- Hay diferentes protocolos de servicios, los dos principales son:
  - SOAP: muy usado en sistemas corporativos
  - REST: muy usado en la internet

REST

- REST, REpresentational State Transfer, es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP.
- Es el tipo de arquitectura más natural y estándar para crear APIs para servicios orientados a Internet.
- La mayoría de las APIs REST usan JSON para comunicarse.
- **Una URI representa un recurso al que se puede acceder o modificar mediante los métodos del protocolo HTTP (POST, GET, PUT, DELETE)**.

API REST - EJEMPLO

- GET /factura (en genérico /factura)
  - Nos permite acceder al listado de facturas
- POST /factura (en genérico /factura)
  - Nos permite crear una factura nueva
- GET /factura/123 (en genérico /factura/:id_fact)
  - Nos permite acceder al detalle de una factura
- PUT /factura/123 (en genérico /factura/:id_fact)
  - Nos permite editar la factura, sustituyendo la totalidad de la información anterior por la nueva.
- DELETE /factura/123 (en genérico /factura/:id_fact)
  - Nos permite eliminar la factura

Respuestas

Indican el resultado de una solicitud

- 1xx: Informational
- 2xx: Success
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

¿Quiénes usan servicios web?

- Casi todos
- Incluso proveen APIs públicas (que podes usar vos desde tu aplicación)

Características

Ventajas:

- Cada servicio puede actualizarse independientemente mientras respete su interfaz
- Pueden programarse en diferentes lenguajes
- Facilita la reutilización
- Desacopla partes no relacionadas del sistema
- Se pueden usar servicios existentes

Desventajas:

- Comunicación y coordinación
- Costo de cambiar la división de servicios
- Complejidad de cambiar las API y de planificar cambios a futuro

REST

REST == **RE**presentational **S**tate **T**ransfer

- Basado en acciones y recursos
- Principios fundamentales
  - Interfaz uniforme
  - Stateless
  - Client-server
  - Cacheable
- Fuertemente soportado por HTTP

Endpoints

Son los puntos de entrada, URLs con las que el cliente accede a un servicio

http://example.com/api/recurso/

http://example.com/api/recurso/123

Un mismo servicio puede tener múltiples endpoints, por ejemplo, para que esté disponible con protocolos diferentes

[APIs] Ejemplos

- Twitter
- GoogleMaps
- Facebook
- Linkedin
- Instagram
- GoogleMail
- …

- https://apigee.com/providers

Ejemplo API: Google Maps

[Iteración 1]

¿Cómo vinculo la API con mi aplicación?

¿Cómo inicializo y muestro un mapa en una posición determinada?

[Iteración 1]

HTML

```
<!-- se incluye la api de GoogleMaps -->
<script type="text/javascript" src="https://maps.google.com/maps/api/js?sensor=true"></script>
<script type="text/javascript" src="./script/main.js"></script>


 <body>
    <!-- DIV que contendrá el mapa -->
    <div id="map_canvas" style="width:100%; height:100%"></div>
 </body>
```

Javascript

```
function initialize() {
    //coordenadas Lat y Long para la ciudad de Tandil
    var latlng = new google.maps.LatLng(-37.328611, -59.136944);

    var mapOptions = {
        zoom: 8,
        center: latlng,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    }
    //se crea el elemento mapa con las opciones especificadas. Se asocia con el elemento HTML donde se va dibujar
    var map = new google.maps.Map(document.getElementById('map_canvas'), mapOptions);

}

google.maps.event.addDomListener(window, 'load', initialize);
```