

# **DESARROLLO WEB DEL LADO DEL SERVIDOR**

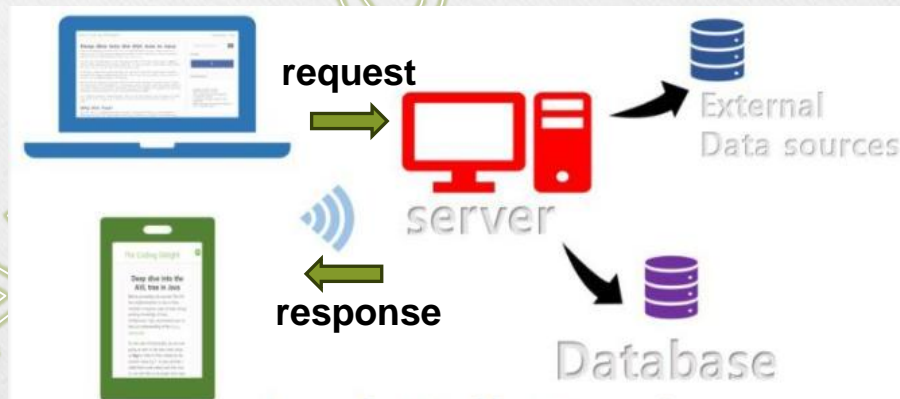
**SESION 2  
PROGRAMACION WEB III**

**INF - 133**

**Lic. Marcelo Aruquipa**

# Desarrollo web lado servidor

- Conocido como Backend
- Parte de la aplicación que maneja la lógica de negocio
- Proceso de datos y comunicación con la BD



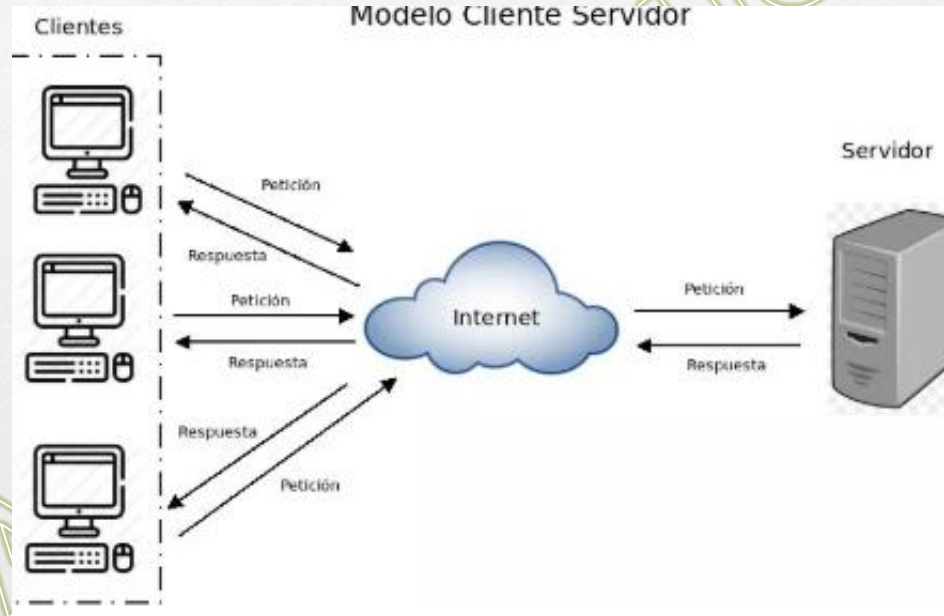


# Arquitectura Cliente Servidor

---

- El frontend y el backend se comunican bajo el modelo cliente-servidor, que consiste:
  1. El cliente (navegador o aplicación móvil) envía una solicitud HTTP al servidor.
  2. El servidor procesa la solicitud, consulta la base de datos si es necesario y genera una respuesta.
  3. El servidor envía la respuesta al cliente, lo muestra en la interfaz del usuario. (navegador)
- El proceso ocurre en milisegundos y es la base de cualquier aplicación web moderna.

# Arquitectura Cliente Servidor





# Funcionamiento Backend

---

## Laboratorio 1.

### Solicitud al servidor y analizar la respuesta

1. En un navegador ingresamos  
**<https://jsonplaceholder.typicode.com/users>**
2. Presiona F12 herramientas de desarrollador
3. Pestaña **network** y actualizar la pagina
4. Revisamos **users** y verificamos la petición GET

# Funcionamiento Backend

## Herramientas de desarrollador

The screenshot displays the Chrome DevTools Network tab. The top toolbar includes icons for Elements, Console, Sources, Network (active), Performance, Memory, Application, Security, Lighthouse, and Recorder. Below the toolbar, there are checkboxes for 'Preserve log', 'Disable cache', and 'No throttling'. A filter bar shows 'Filter' and 'Invert' options, followed by a list of filter categories: All, Fetch/XHR, Doc, CSS, JS, Font, Img, Media, Manifest, WS, Wasm, and Other. A timeline at the top shows a green bar for the 'users' request, with a red vertical line indicating the response time at approximately 180 ms. The main panel is divided into two sections: 'Name' on the left and 'Headers' on the right. The 'Name' section lists three items: 'users' (selected), 'copy.js', and 'favicon.ico'. The 'Headers' section shows the 'General' tab with the following details:

Name	Value
Request URL:	https://jsonplaceholder.typicode.com/users
Request Method:	GET
Status Code:	304 Not Modified
Remote Address:	104.21.32.1:443
Referrer Policy:	strict-origin-when-cross-origin



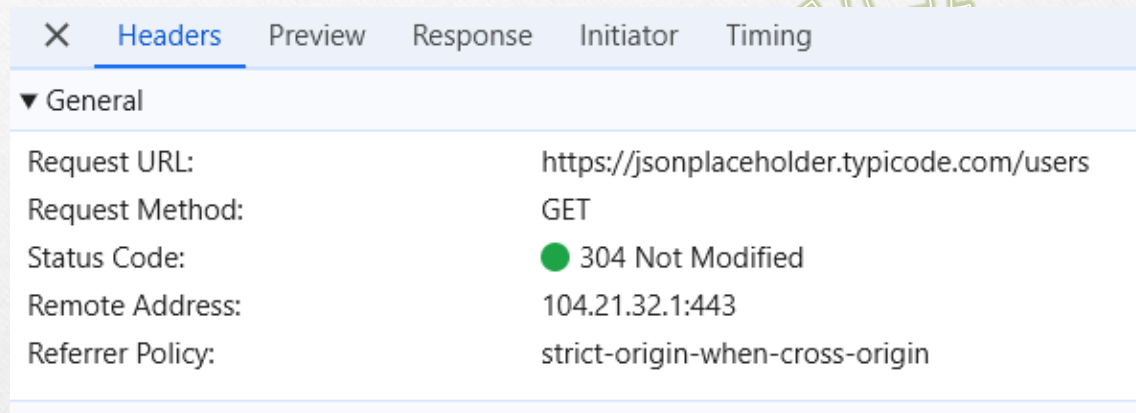
# Funcionamiento Backend

## Respuesta en formato JSON a la petición GET

1. Dirección del Usuario (address):  
street, suite, city, zipcode, objeto geo
2. Empresa del Usuario (company)  
name, catchPhrase, bs
3. Información del Usuario:  
id, name, username, email, phone, website,

```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
]
```

# Funcionamiento Backend



×	Headers	Preview	Response	Initiator	Timing
▼ General					
Request URL:	https://jsonplaceholder.typicode.com/users				
Request Method:	GET				
Status Code:	● 304 Not Modified				
Remote Address:	104.21.32.1:443				
Referrer Policy:	strict-origin-when-cross-origin				

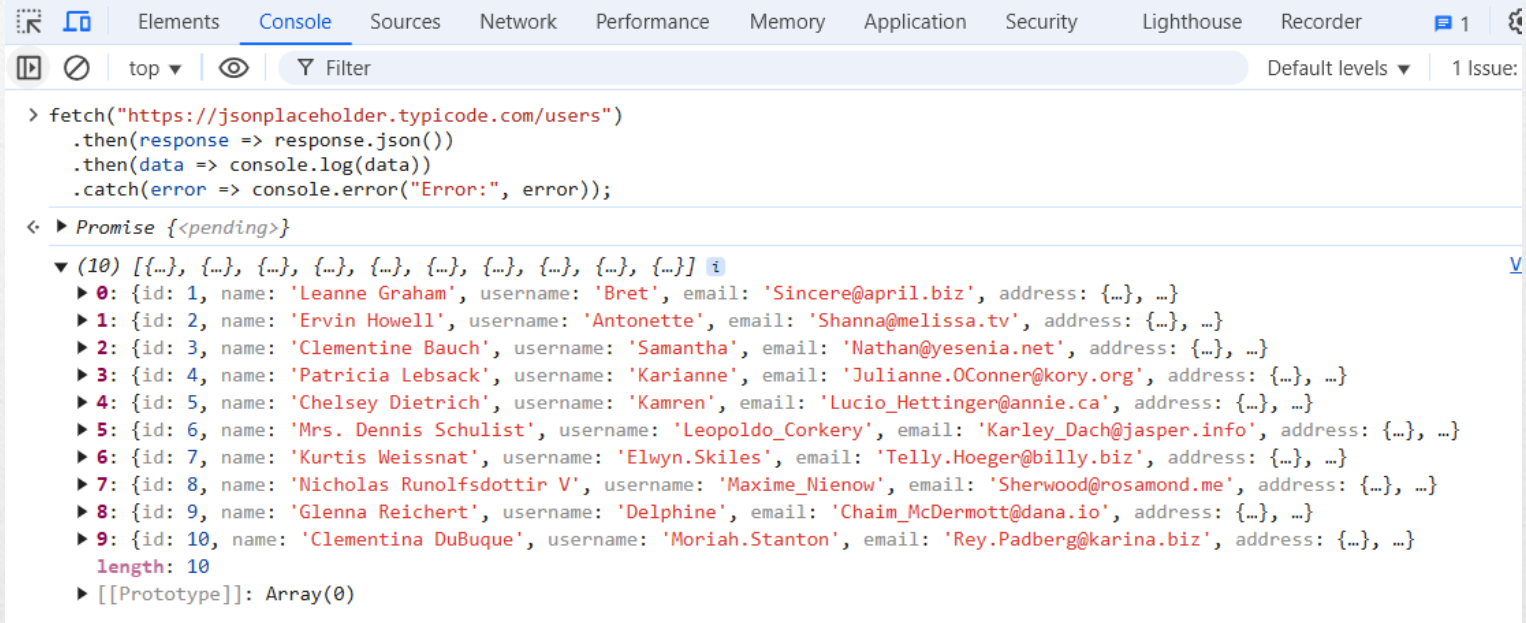
Donde:

- **Request URL:** URL a la que el navegador hizo la solicitud
- **Request Method:** Método HTTP GET que se utiliza para solicitar información
- **Status Code:** Código de estado HTTP (304 recurso no cambio desde la ultima vez)
- **Remote Address:** Dirección IP del servidor que respondió la solicitud
- **Referrer Policy:** Define qué información se envía cuando la solicitud viene de otro sitio web (cross-origin)



# Funcionamiento Backend

## Solicitud GET con JavaScript



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the execution of a JavaScript fetch request to 'https://jsonplaceholder.typicode.com/users'. The request is successful, and the response is a JSON array of 10 user objects. The console output shows the promise state, the resolved array of users, and the array's length and prototype.

```
> fetch("https://jsonplaceholder.typicode.com/users")
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error("Error:", error));

< Promise {<pending>}

▼ (10) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}] ⓘ
  ▶ 0: {id: 1, name: 'Leanne Graham', username: 'Bret', email: 'Sincere@april.biz', address: {...}, ...}
  ▶ 1: {id: 2, name: 'Ervin Howell', username: 'Antonette', email: 'Shanna@melissa.tv', address: {...}, ...}
  ▶ 2: {id: 3, name: 'Clementine Bauch', username: 'Samantha', email: 'Nathan@yesenia.net', address: {...}, ...}
  ▶ 3: {id: 4, name: 'Patricia Lebsack', username: 'Karianne', email: 'Julianne.OConner@kory.org', address: {...}, ...}
  ▶ 4: {id: 5, name: 'Chelsey Dietrich', username: 'Kamren', email: 'Lucio_Hettinger@annie.ca', address: {...}, ...}
  ▶ 5: {id: 6, name: 'Mrs. Dennis Schulist', username: 'Leopoldo_Corkery', email: 'Karley_Dach@jasper.info', address: {...}, ...}
  ▶ 6: {id: 7, name: 'Kurtis Weissnat', username: 'Elwyn.Skiles', email: 'Telly.Hoeger@billy.biz', address: {...}, ...}
  ▶ 7: {id: 8, name: 'Nicholas Runolfsson', username: 'Maxime_Nienow', email: 'Sherwood@rosamond.me', address: {...}, ...}
  ▶ 8: {id: 9, name: 'Glenn Reichert', username: 'Delphine', email: 'Chaim_McDermott@dana.io', address: {...}, ...}
  ▶ 9: {id: 10, name: 'Clementina DuBuque', username: 'Moriah.Stanton', email: 'Rey.Padberg@karina.biz', address: {...}, ...}
  length: 10
  [[Prototype]]: Array(0)
```

# Funcionamiento Backend

## Solicitud GET con JavaScript

- Modificar el código para mostrar solo los nombres de los usuarios en la consola

```
> fetch("https://jsonplaceholder.typicode.com/users")  
  .then(response => response.json())  
  .then(data => data.forEach(user => console.log(user.name)))  
  .catch(error => console.error("Error:", error));
```

```
< ▶ Promise {<pending>}
```

Leanne Graham

Ervin Howell

Clementine Bauch

Patricia Lebsack

Chelsey Dietrich

Mrs. Dennis Schulist

Kurtis Weissnat

Nicholas Runolfsdottir V

Glenna Reichert

Clementina DuBuque



# Conclusión

---

- Cómo hacer una solicitud GET en el navegador.
- Cómo ver la respuesta de una API REST.
- Cómo inspeccionar la solicitud en las herramientas de desarrollo.
- Cómo hacer la misma solicitud con JavaScript.

GRACIAS

---