



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES



Node.js Express

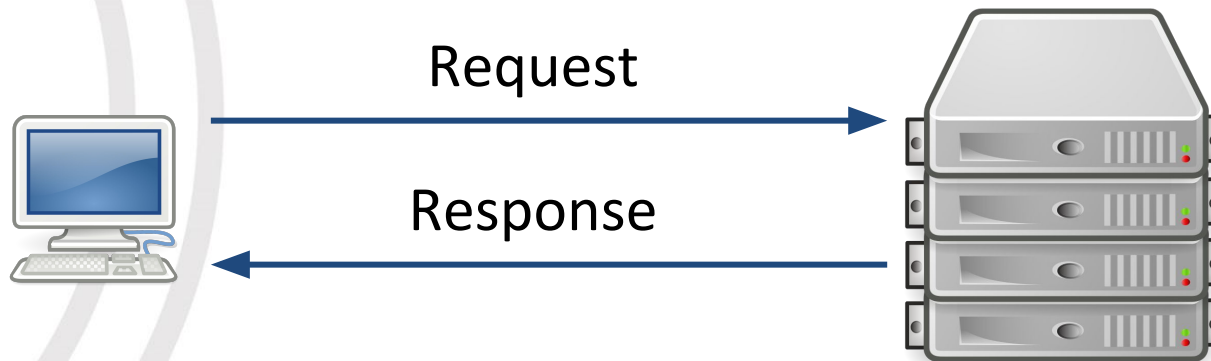
Contenido

- Cliente / Servidor
- Servicios
 - API Rest
- Express (Framework de Node.js)
 - Express Generator
 - Direccionamiento
 - Middleware

Cliente / Servidor

Cliente / Servidor

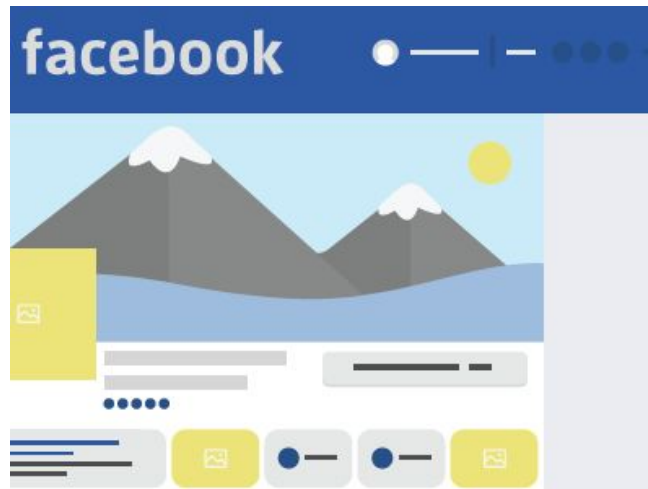
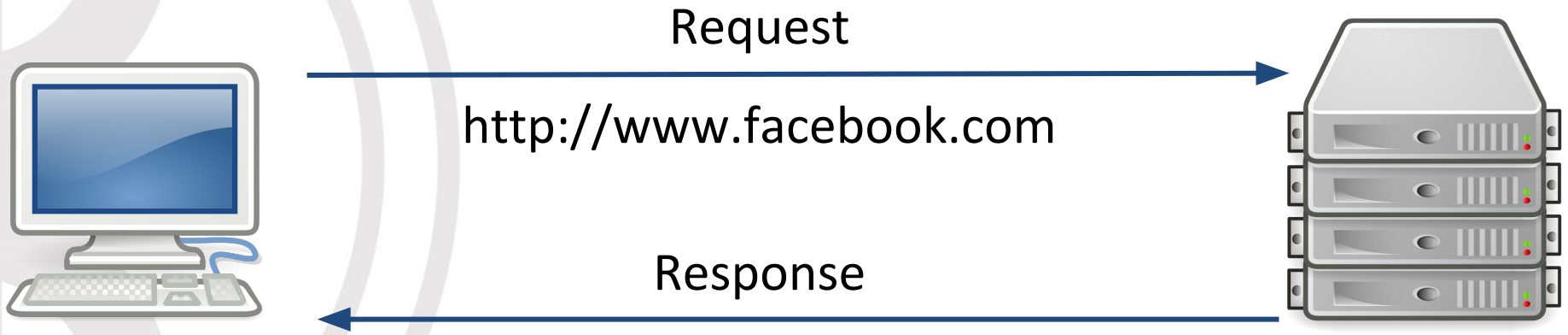
Esquema de comunicación básico cliente / servidor



- WEB (Páginas web)
- Servicios

Cliente / Servidor

Ejemplo WEB



Ejemplo WEB

Request:

Request URL: <https://www.facebook.com/>

Request Method: GET

user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36

Ejemplo WEB

Response - Body:

```
<!DOCTYPE html>
<html lang="es" id="facebook" class="no_js">
  <head>
    <meta charset="utf-8"/>
    <meta name="referrer" content="origin-when-crossorigin" id="meta_referrer"/>
    <link rel="preload" href="https://static.xx.fbcdn.net/rsrc.php/v3/yg/r/oLeg5FRnkrz.js" as="script"
crossorigin="anonymous"/>
    <link rel="preload" href="https://static.xx.fbcdn.net/rsrc.php/v3/yw/r/Rybo3uelw0z.js" as="script"
crossorigin="anonymous"/>
    <link rel="preload" href="https://static.xx.fbcdn.net/rsrc.php/v3/yo/l/0,cross/encqKUByQeT.css"
as="style" crossorigin="anonymous"/>
    <link rel="preload" href="https://static.xx.fbcdn.net/rsrc.php/v3/iqrt4/yn/l/es_LA/kdnxb6QqcW6.js"
as="script" crossorigin="anonymous"/>
    <link rel="preload" href="https://static.xx.fbcdn.net/rsrc.php/v3/i7Hl4/yH/l/es_LA/p_1ko6UnzFo.js"
as="script" crossorigin="anonymous"/>
```



Cliente / Servidor

Ejemplo Servicios



Request

<https://restcountries.eu/rest/v2/name/argentina>



Response

```
{
  hey: "guy",
  anumber: 243,
  - anobject: {
    whoa: "nuts",
    - anarray: [
      1,
      2,
      "thr<h1>ee"
    ],
    more: "stuff"
  },
  awesome: true,
  bogus: false,
  meaning: null,
  japanese: "明日がある。",
  link: http://jsonview.com,
  notLink: "http://jsonview.com is great"
}
```



UTN.BA

FACULTAD
REGIONAL
BUENOS AIRES

Ejemplo Servicios

Request:

Request URL:

<https://restcountries.eu/rest/v2/name/argentina>

Request Method: GET

user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36



Ejemplo WEB

Response - Body:

```
[{"name": "Argentina", "topLevelDomain": [".ar"], "alpha2Code": "AR", "alpha3Code": "ARG", "callingCodes": [54], "capital": "Buenos Aires", "altSpellings": ["AR", "Argentine Republic", "República Argentina"], "region": "Americas", "subregion": "South America", "population": 43590400, "latlng": [-34.0, -64.0], "demonym": "Argentinean", "area": 2780400.0, "gini": 44.5, "timezones": ["UTC-03:00"], "borders": ["BOL", "BRA", "CHL", "PRY", "URY"], "nativeName": "Argentina", "numericCode": "032", "currencies": [{"code": "ARS", "name": "Argentine peso", "symbol": "$"}], "languages": [{"iso639 1": "es", "iso639 2": "spa", "name": "Spanish", "nativeName": "Español"}, {"iso639 1": "gn", "iso639 2": "grn", "name": "Guaraní", "nativeName": "Avañe'ẽ"}], "translations": {"de": "Argentinien", "es": "Argentina", "fr": "Argentine", "ja": " アルゼンチン", "it": "Argentina", "br": "Argentina", "pt": "Argentina", "nl": "Argentinië", "hr": "Argentina", "fa": "آرژانتین"}}
```



Cliente / Servidor

- Cuando trabajamos con peticiones **web**, hacemos un request y obtenemos una respuesta de tipo web (HTML, CSS, JS)
- Cuando trabajamos con **servicios**, obtenemos datos sin formato para ser representados por nuestra aplicación cliente o incluso por otro equipo o dispositivo.



Servicios API Rest

Servicios - API Rest

- El término **REST** (Representational State Transfer) se originó en el año 2000, descrito en la tesis de Roy Fielding, padre de la especificación HTTP.
- Un servicio REST no es una arquitectura software, sino un conjunto de restricciones con las que podemos crear un estilo de arquitectura software, la cual podremos usar para crear aplicaciones web respetando HTTP.
- Por poner algún ejemplo tenemos los sistemas de identificación de Facebook, la autenticación en los servicios de Google (hojas de cálculo, Google Analytics, ...), Mercado Libre, Mercado Pago, etc... exponen servicios basados en API Rest.



Servicios - API Rest

- Las operaciones más importantes que nos permitirán manipular los recursos son cuatro: **GET** para consultar y leer, **POST** para crear, **PUT** para editar y **DELETE** para eliminar.
- Para terminar, comentar que lo más importante a tener en cuenta al crear nuestro servicio o API REST no es el lenguaje en el que se implemente sino que las respuestas a las peticiones se hagan en **XML** o **JSON**, ya que es el lenguaje de intercambio de información más usado.





Express

- Las ventajas que nos ofrece Express, como cualquier otro framework, es que sus librerías nos facilitan mucho el trabajo tanto para la realización de proyectos web como de proyectos basados en servicios.
- Las ventajas específicas de Express son:
 - Sistema de plantillas web
 - Manejo de rutas para API Rest
 - Middleware
 - Otros...

Express - Instalación

- Express es parte de los paquetes contenidos en el NPM. Para poder utilizar paquetes del NPM en nuestro proyecto debemos pararnos en la carpeta contenedora del proyecto y desde la consola ejecutar el comando:

npm init

Esto nos creará un archivo llamado **package.json** que es un archivo descriptor de nuestro proyecto y que además, describe los módulos externos que éste necesita para funcionar.

Express - Instalación

- Para instalar Express utilizamos el siguiente comando:

npm install express

Esta instrucción instalará los módulos necesarios en la carpeta **node_modules** de nuestro proyecto.

Express - Hello World

Express server

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```





Express Generator

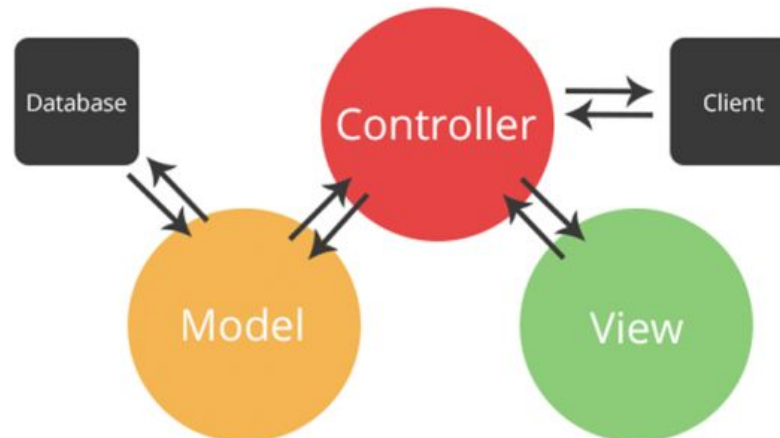


UTN.BA

FACULTAD
REGIONAL
BUENOS AIRES

Express - Generator

- El **Express Generator** es una herramienta que nos va a generar el esqueleto de un servidor web utilizando el patrón de diseño **MVC**.
- **MVC** (Model View Controller) es un patrón de diseño que separa nuestro código en tres capas bien diferenciadas:
 - **Model**: Es la capa que se encarga de manipular los datos
 - **View**: Se encarga de la parte visual (las páginas web)
 - **Controller**: Es la capa que implementa la lógica utilizando los datos del modelo para finalmente enviar el resultado a una vista que es lo que finalmente el servidor terminará enviando.



Express - Generator

- Para instalar el **Express Generator** utilizamos el siguiente comando:

npm install express-generator -g

Esta instrucción instalará el express generator y el modificador **-g** hará que esta herramienta se instale de forma global en nuestro sistema para poder iniciar un proyecto nuevo en cualquier carpeta.

Express - Generator

Express Generator – Crear aplicación

Abrimos la consola y nos ubicamos en el directorio en el cual queremos crear nuestra aplicación, luego ejecutamos:

express --view=ejs miAplicacion


Express Generator

Nombre de
nuestro proyecto

Sistema de plantillas

Express - Generator

Estructura de archivos y directorios creadas por el **Express Generator**



```
create : myApp/  
create : myApp/public/  
create : myApp/public/javascripts/  
create : myApp/public/images/  
create : myApp/public/stylesheets/  
create : myApp/public/stylesheets/style.css  
create : myApp/routes/  
create : myApp/routes/index.js  
create : myApp/routes/users.js  
create : myApp/views/  
create : myApp/views/error.ejs  
create : myApp/views/index.ejs  
create : myApp/app.js  
create : myApp/package.json  
create : myApp/bin/  
create : myApp/bin/www
```



UTN.BA

FACULTAD
REGIONAL
BUENOS AIRES

Express - Generator


Para correr nuestro servidor recién creado tenemos que hacer lo siguiente:

1. Ejecutar el comando para instalar todas las dependencias que necesita nuestro programa:
 - **npm install**
2. Iniciar el servidor con el comando:
 - **npm start**

Una vez hecho esto, tendremos un servidor con dos rutas de ejemplo predefinidas para empezar a personalizar.

Express - Generator

Estructura de archivos y directorios creadas por el **Express Generator**



```
create : myApp/  
create : myApp/public/  
create : myApp/public/javascripts/  
create : myApp/public/images/  
create : myApp/public/stylesheets/  
create : myApp/public/stylesheets/style.css  
create : myApp/routes/  
create : myApp/routes/index.js  
create : myApp/routes/users.js  
create : myApp/views/  
create : myApp/views/error.ejs  
create : myApp/views/index.ejs  
create : myApp/app.js  
create : myApp/package.json  
create : myApp/bin/  
create : myApp/bin/www
```



UTN.BA

FACULTAD
REGIONAL
BUENOS AIRES

Express - Generator

bin: Directorio propio de express, en el cual podremos visualizar la creación de un servidor en node js.

node_modules: Carpeta propia de node, allí se alojarán todos los módulos instalados con npm install.

public: Se aloja el contenido público como **Imágenes, Javascript, CSS** u otros archivos estáticos que sean de acceso público.

routes: Se alojan los archivos que procesarán las rutas, es decir que “rutearán” los requests, es decir que serán visualizados de acuerdo a la URL que accedamos.

views: Se alojan las vistas de nuestra aplicación. En este caso son de tipo ejs.

Direccionamiento

Express - Direcccionamiento

El direccionamiento hace referencia a la determinación de cómo responde una aplicación a una solicitud de cliente en un determinado punto final, que es un URI (o una vía de acceso) y un método de solicitud HTTP específico (GET, POST, etc.). Cada ruta puede tener una o varias funciones de manejador, que se excluyen cuando se correlaciona la ruta. La definición de ruta tiene la siguiente estructura:

app.**METHOD**(PATH, HANDLER)

app es una instancia de express.

METHOD es un método de solicitud HTTP (GET, POST, PUT, DELETE).

PATH es una vía de acceso en el servidor. Ejemplo “/”, “users”, “users/1”

HANDLER es la función que se ejecuta cuando se correlaciona la ruta. Es la función de callback que se ejecutara al ingresar por esa ruta.

Express - Direccionamiento

Por ejemplo en el directorio route encontramos el archivo index.js, el mismo tendrá la definición de nuestra ruta por default (es decir cuando accedemos a la home de nuestro sitio “/”)

```
router.get('/', function (req, res, next) {  
  res.render('index', {  
    title: 'Express'  
  });  
});
```





Middleware

Express - Middleware

Las funciones de middleware son funciones que tienen acceso al objeto de solicitud (**req**), al objeto de respuesta (**res**) y a la siguiente función de middleware en el ciclo de solicitud/respuestas de la aplicación. La siguiente función de middleware se denota normalmente con una variable denominada `next`.

Las funciones de middleware pueden realizar las siguientes tareas:

- Ejecutar cualquier código.
- Realizar cambios en la solicitud y los objetos de respuesta.
- Finalizar el ciclo de solicitud/respuestas.
- Invocar el siguiente middleware en la pila.

Express - Middleware

Ejemplo

```
var myLogger = function (req, res, next) {  
  console.log('LOGGED');  
  next();  
};  
  
app.use(myLogger);
```

Esta función se ejecutará antes de rutear el request a la ruta definida.
Es posible asociar un middleware a una ruta en particular.

Taller