

**¿Qué es
Express.js?**

Tu primer server

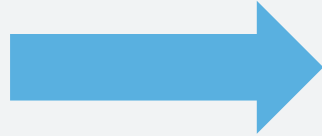
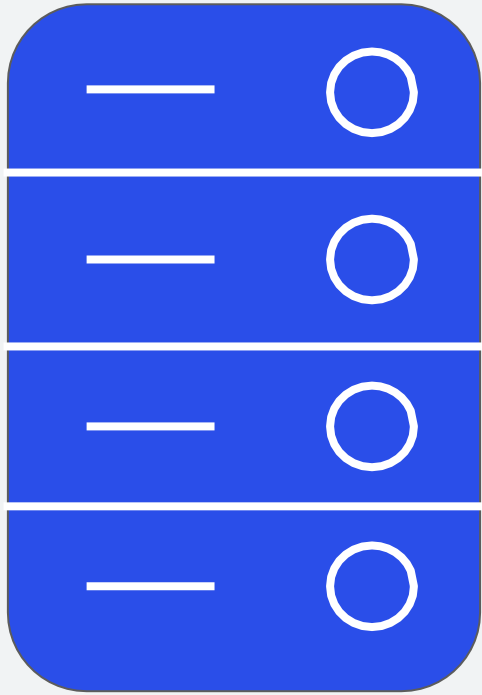
```
const express = require('express');  
const app = express();  
const port = 3000;
```

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

```
app.listen(port, () => {  
  console.log(`http://localhost:${port}`)  
});
```

Configuración

Routing



GET



PUT





POST



DELETE



Method	/song	/song/{id}
GET	Get list	Get
PUT	Replace 	Update/Replace
PATCH	No Apply	Update
POST	Create	No Apply
DELETE	Delete 	Delete

GET

api.example.com/tasks/{id}/
api.example.com/people/{id}/
api.example.com/users/{id}/tasks/

Query Params

api.example.com/products

api.example.com/products?page=1

api.example.com/products?limit=10&offset=0

api.example.com/products?region=USA

api.example.com/products?region=USA&brand=XYZ



POST

api.example.com/api/v1/users/

api.example.com/api/v1/tasks/

api.example.com/api/v1/customers/

DELETE
PUT
PATCH

Method	/song	/song/{id}
GET	Get list	Get
PUT	Replace 	Update/Replace
PATCH	No Apply	Update
POST	Create	No Apply
DELETE	Delete 	Delete

STATUS CODE

Middlewares



```
graph LR; Request[Request] --> Middleware[Middleware]; Middleware --> Response[Response];
```

Request

Middleware

Response

```
function (req, res, next) {  
  if (something) {  
    res.send( 'end' );  
  } else {  
    next();  
  }  
}
```

```
function (error, req, res, next) {  
  if (error) {  
    res.status(500).json({error});  
  } else {  
    next();  
  }  
}
```

Uses Cases

- Funcionar como pipes
- Validar datos
- Capturar errores
- Validar permisos
- Controlar accesos

Middleware for Errors

Validación de datos con express-validations

Consideraciones para producción

Recomendaciones

- Cors
- Https
- Procesos de Build
- Remover logs
- Seguridad (Helmet)
- Testing

CORS

Deployment to Heroku