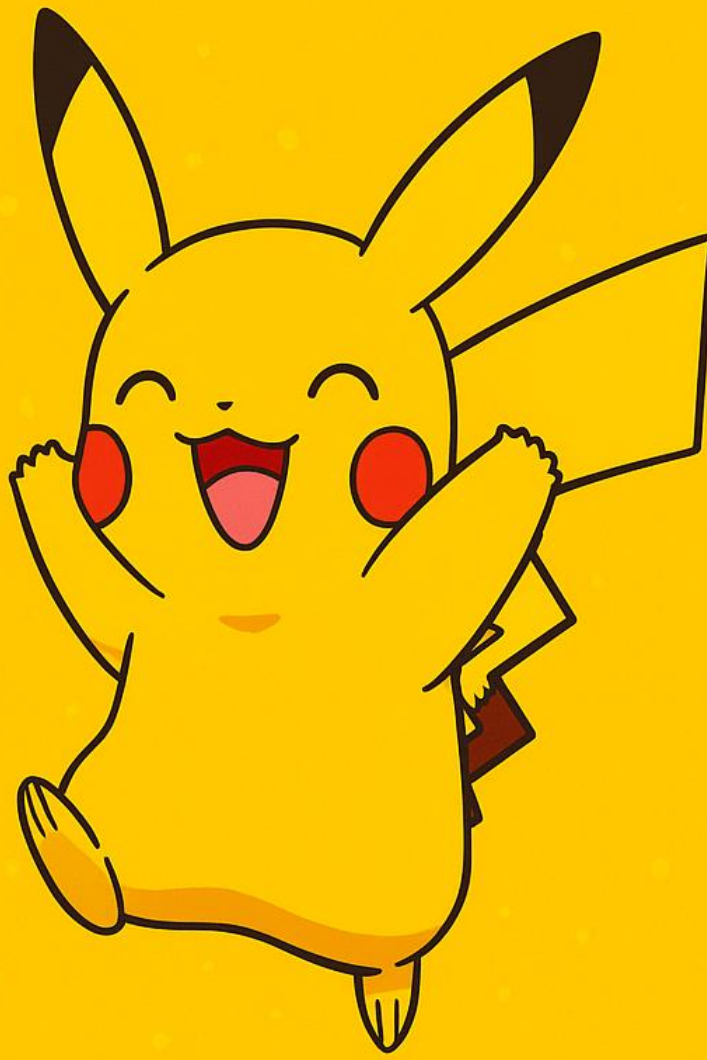
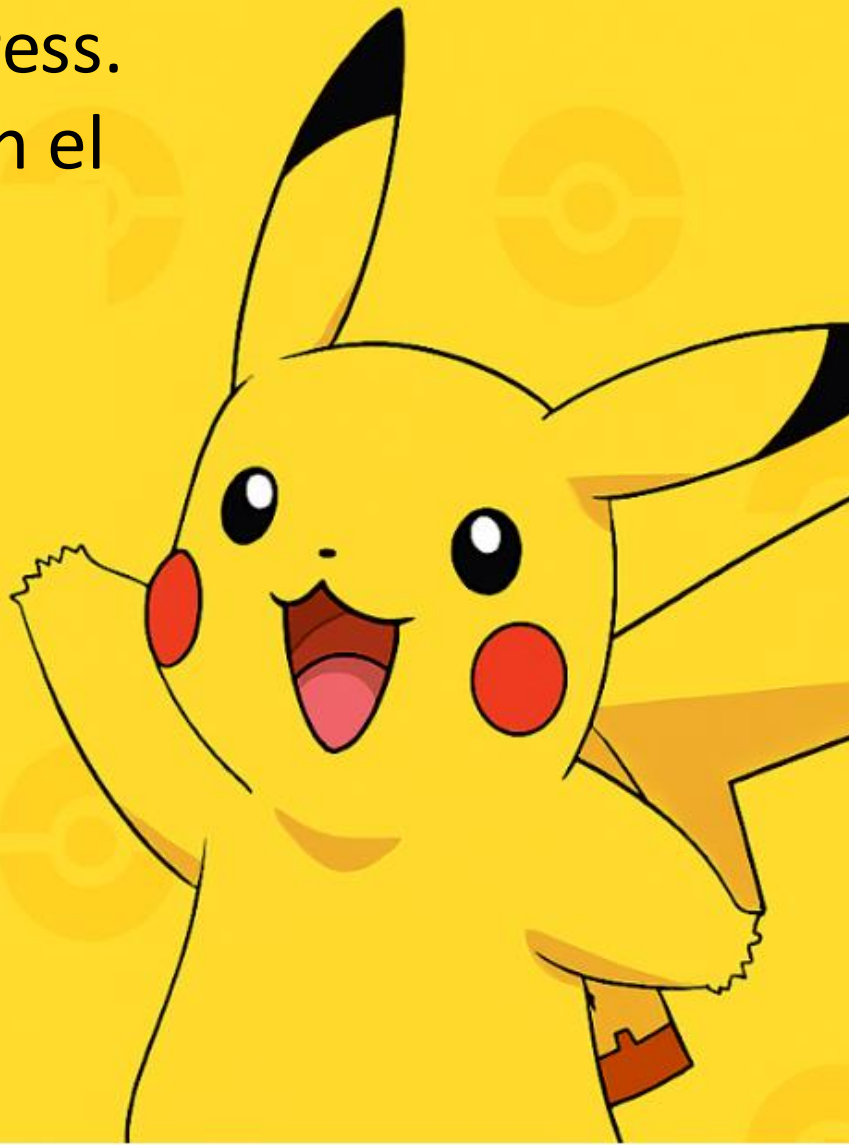
TP Integrador
okémon Business



Plucci Kevin y Messina Joaquin

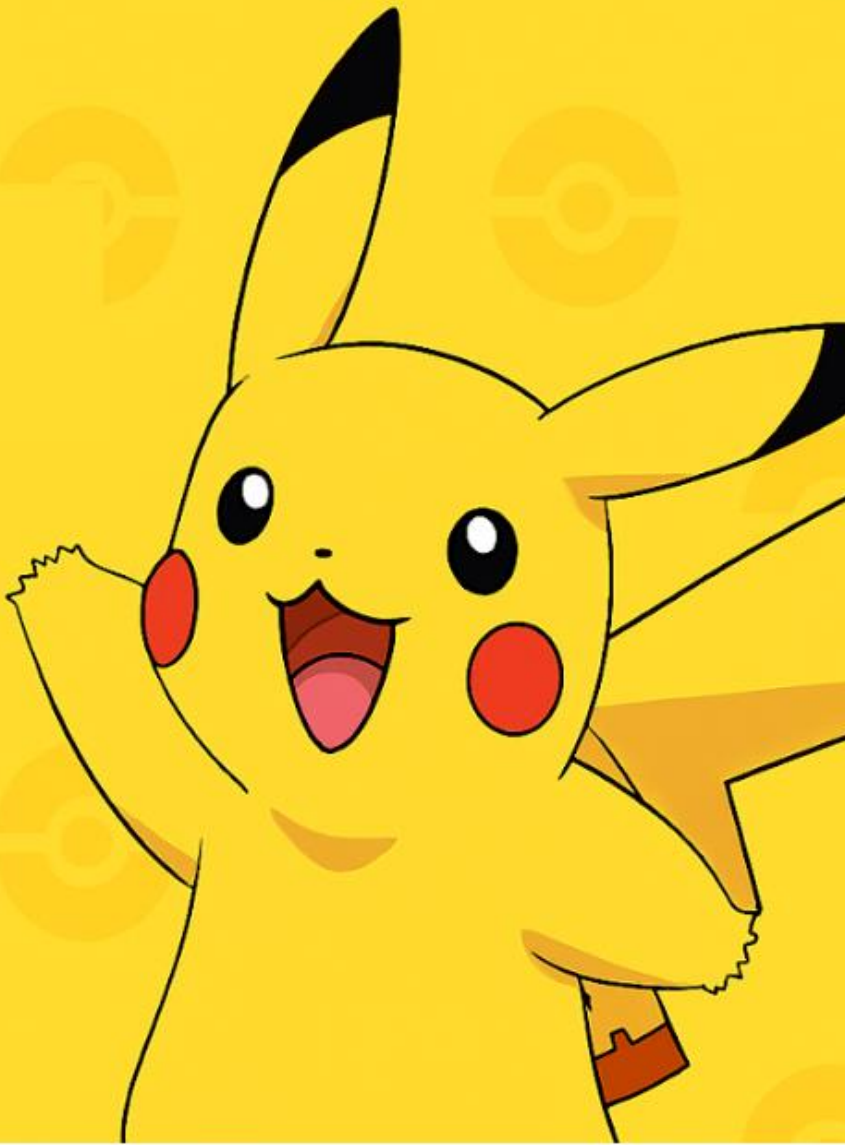
Backend del Proyecto

- El backend usa Node.js y Express.
- - Node.js ejecuta JavaScript en el servidor.
- - Express organiza rutas, middleware y estructura modular.
- Ventajas:
- - Código asincrónico
- - Modularidad
- - Ecosistema amplio
- - Todo JavaScript.



Índice

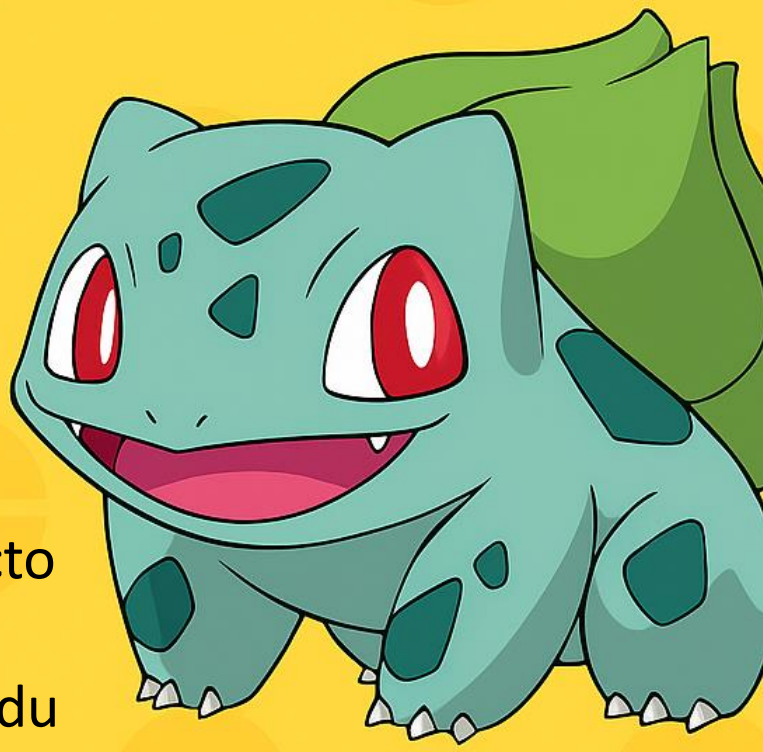
- 1. ¿Qué es CRUD?
- 2. Explicación de código: CRUD
- 3. Patrón MVC en el proyecto
- 4. Explicación de código: MVC
- 5. Middleware: concepto y ejemplos
- 6. Explicación de código: Middleware
- 7. EJS: vistas dinámicas en Node.js
- 8. Explicación de código: EJS
- 9. Dashboard
- 10. Explicación de código: Dashboard
- 11. Variables de entorno (.env)
- 12. Environments (Entornos)
- 13. Endpoints
- 14. Rutas (Routes) y su estructura
- 15. Explicación de código: Rutas
- 16. Ejemplos concretos
- 17. Tabla resumen
- 18. Importancia de cada componente
- 19. Conclusiones
- 20. Cierre



¿Qué es CRUD?

- CRUD define
  las operaciones básicas de datos:
- • Create (Crear)
- • Read (Leer)
- • Update (Actualizar)
- • Delete (Eliminar)

- En el proyecto:
- - POST /api/productos: crear producto
- - GET /api/productos: leer productos
- - PUT /api/productos/:id: actualizar producto
- - DELETE /api/productos/:id: eliminar/desactivarproducto



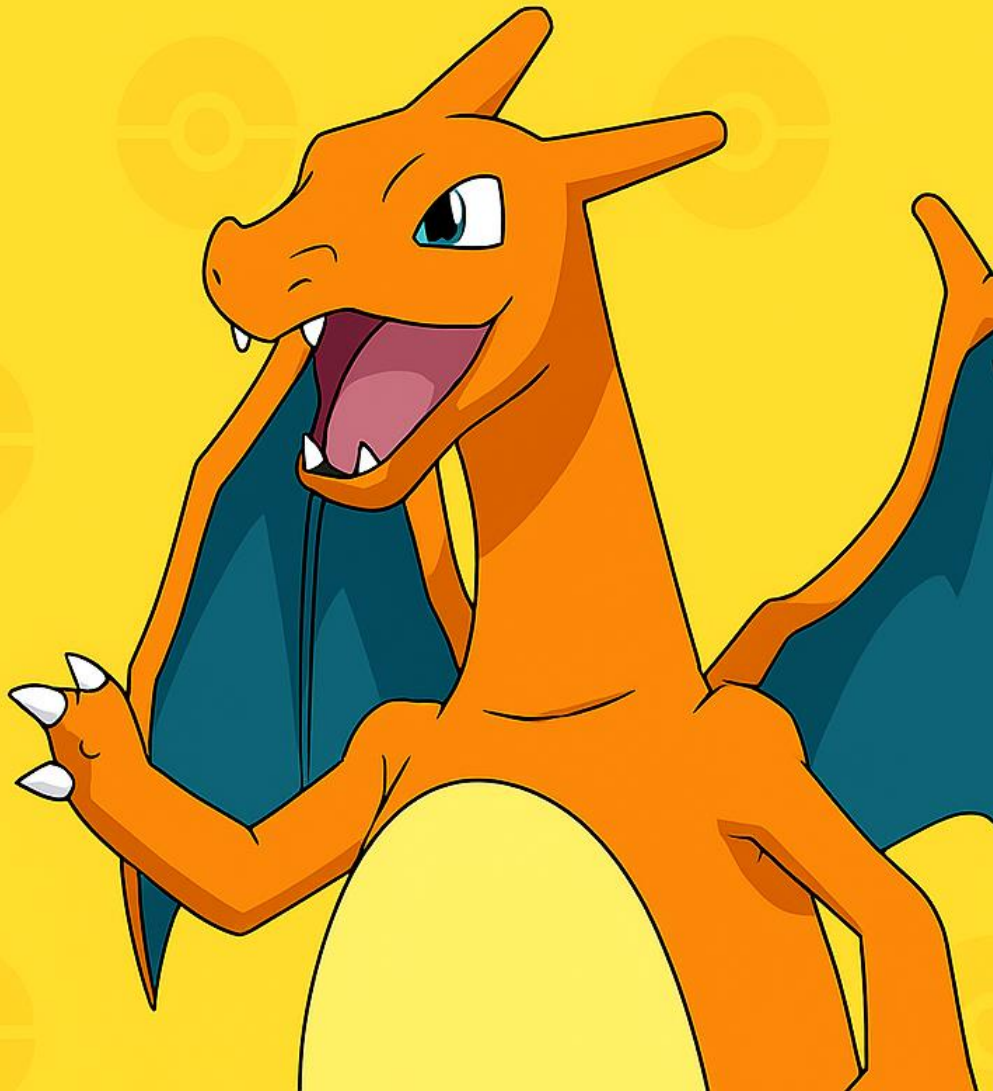
Explicación de código: CRUD

- Ejemplo en productoController.js:

- // Crear
- exports.crearProducto = (req, res) => { ... }

- // Leer
- exports.listarProductos = (req, res) => { ... }

- Las rutas llaman a estos métodos según corresponda.

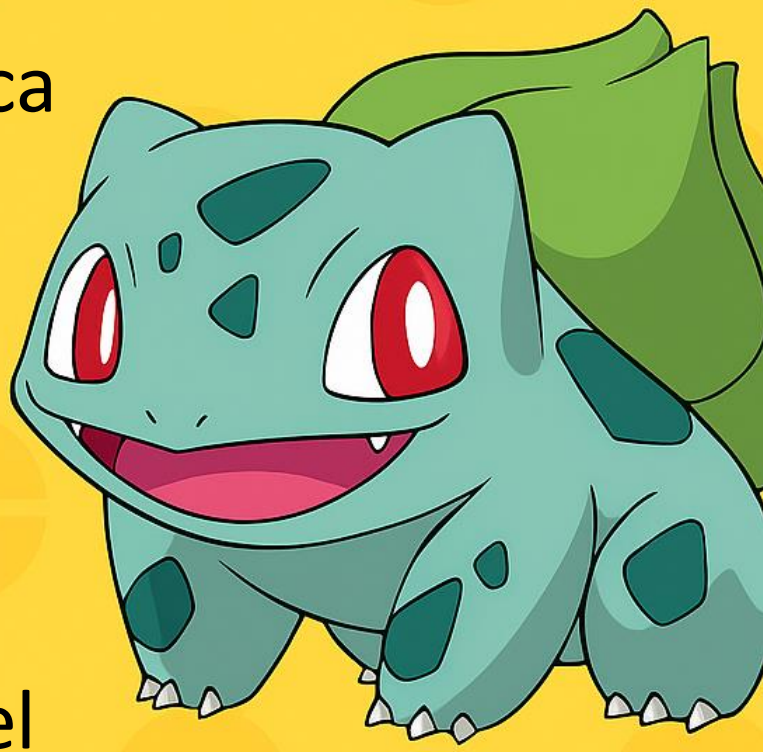


Patrón MVC en el proyecto

- MVC divide el sistema en capas:

- • Modelo: maneja datos y lógica
- • Vista: muestra los datos (EJS)
- • Controlador: conecta vistas y modelos

- Cada parte cumple su rol y permite modificar sin romper el resto.



Explicación de código: MVC

- Flujo típico:
- 1. Ruta:
  /routes/productos.js recibe la petición
- 2. Controlador:
  /controllers/productoController.js
  maneja la lógica
- 3. Modelo:
  /models/producto.js interactúa con la base
- 4. Vista:
  /views/productos/lista.ejs muestra losdatos
- →
  Permite orden y separación deresponsabilidades.

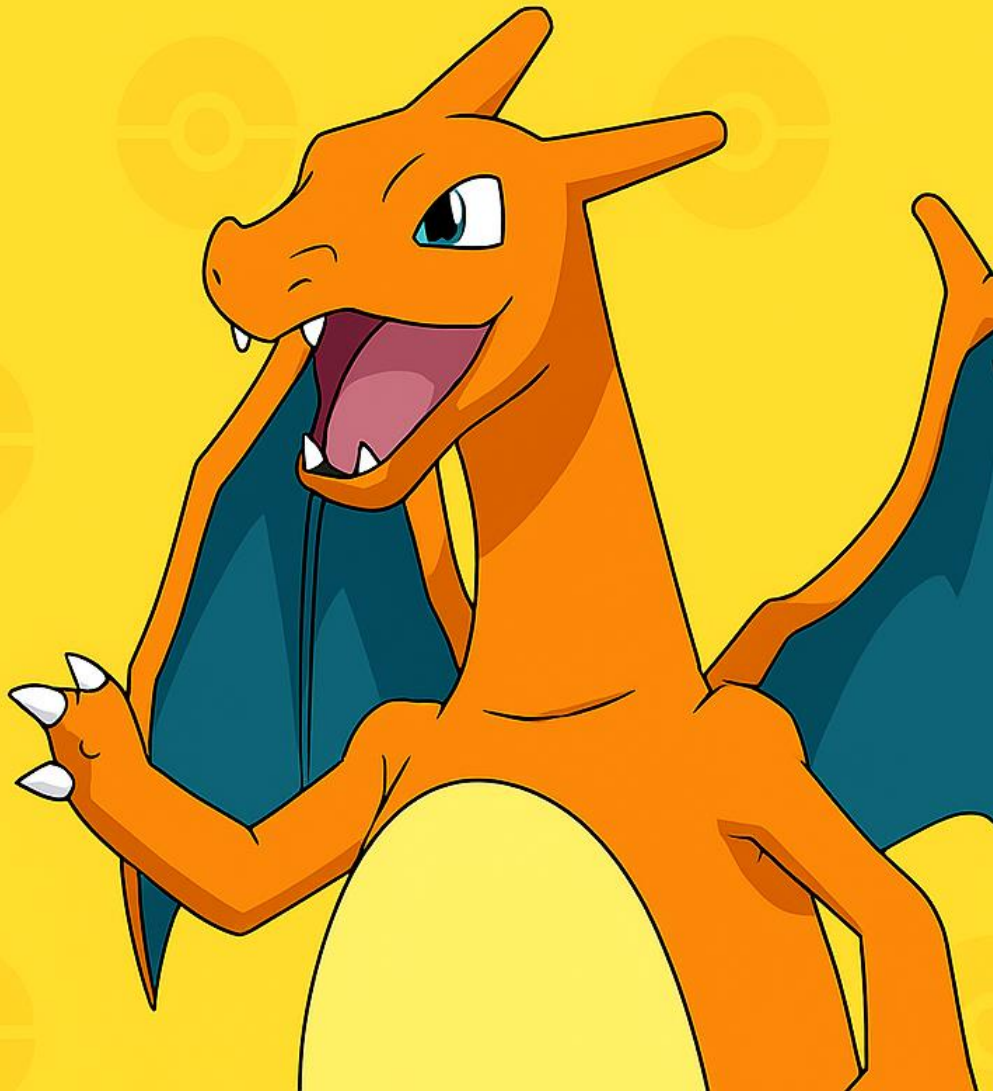
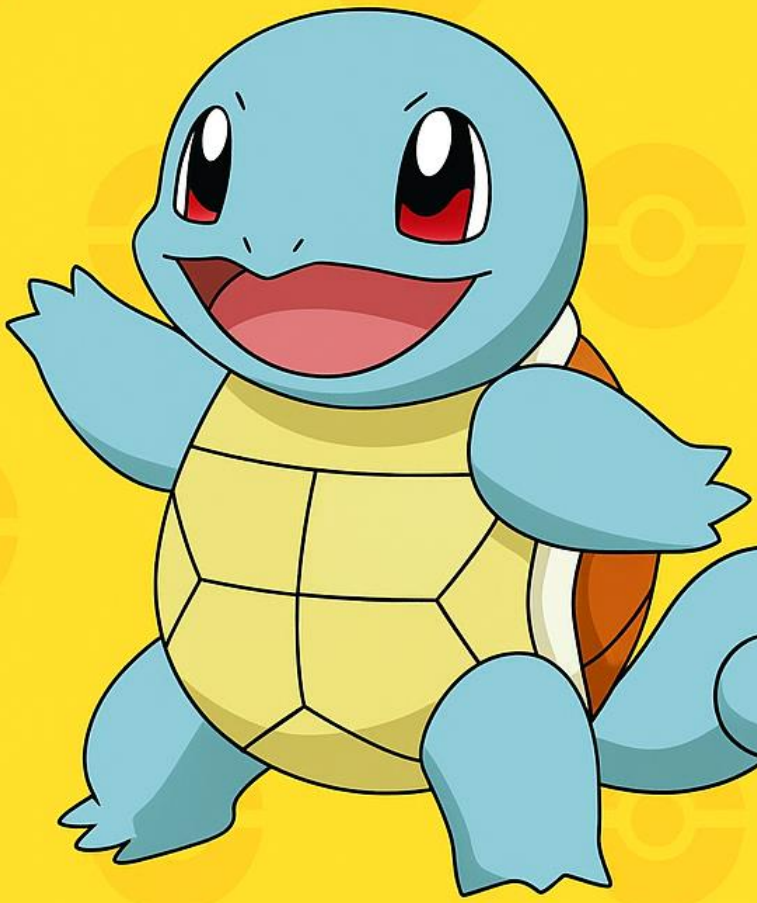


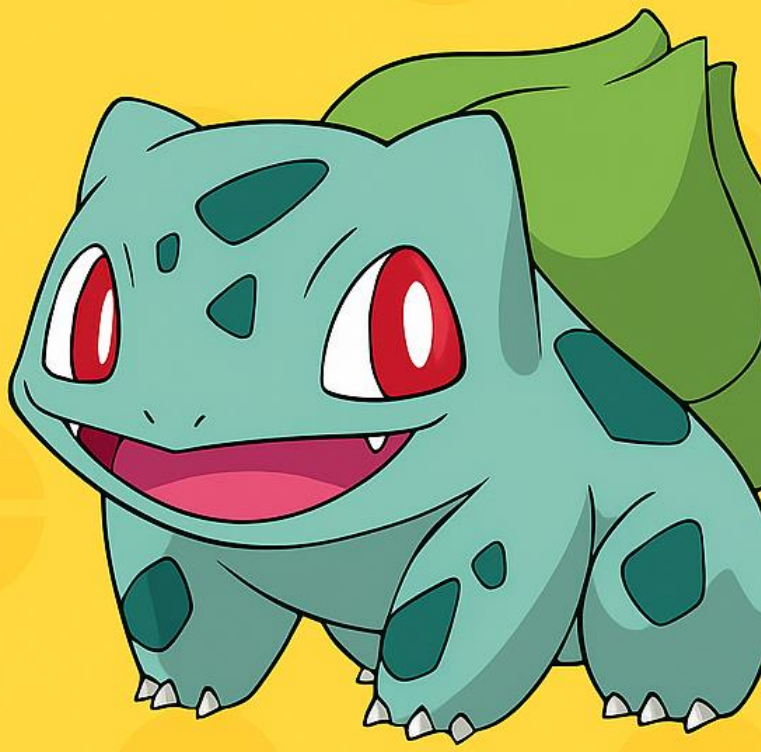
Diagrama MVC (flujo en el proyecto)

[Usuario]
↓
[Rutas]
↓
[Controlador]
↓
[Modelo]
↓
[Base de Datos]
↑
[Vista (EJS)]
↑
[Usuario]



Middleware: concepto y ejemplos

- Middleware: función que intercepta y procesa las peticiones.
- En el proyecto:
- • Validar autenticación
- • Manejar errores
- • Procesar datos antes del controlador
- Ejemplo: /middleware/auth.js



Explicación de código: Middleware

- Archivo: middleware/auth.js

- module.exports = function(req, res, next) {
-
- if (req.session && req.session.user) return next();
-
- return res.redirect('/login');
- }

- Filtro antes de los controladores protegidos.

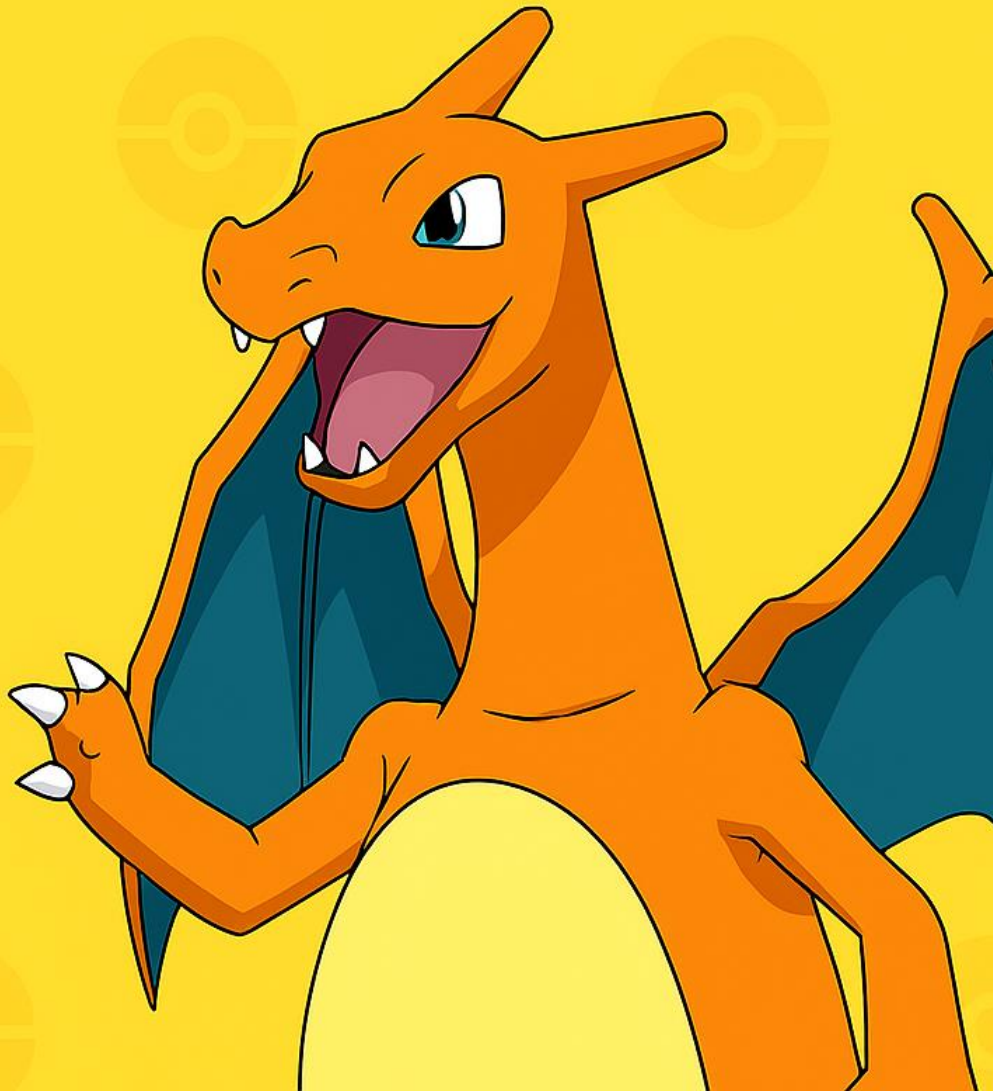
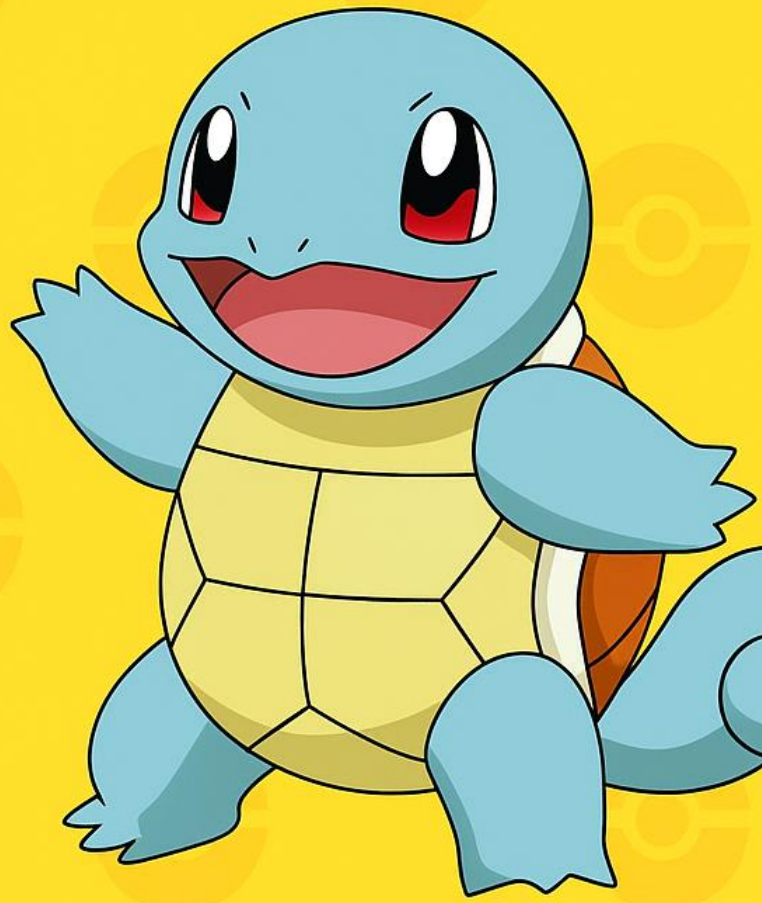


Diagrama Middleware

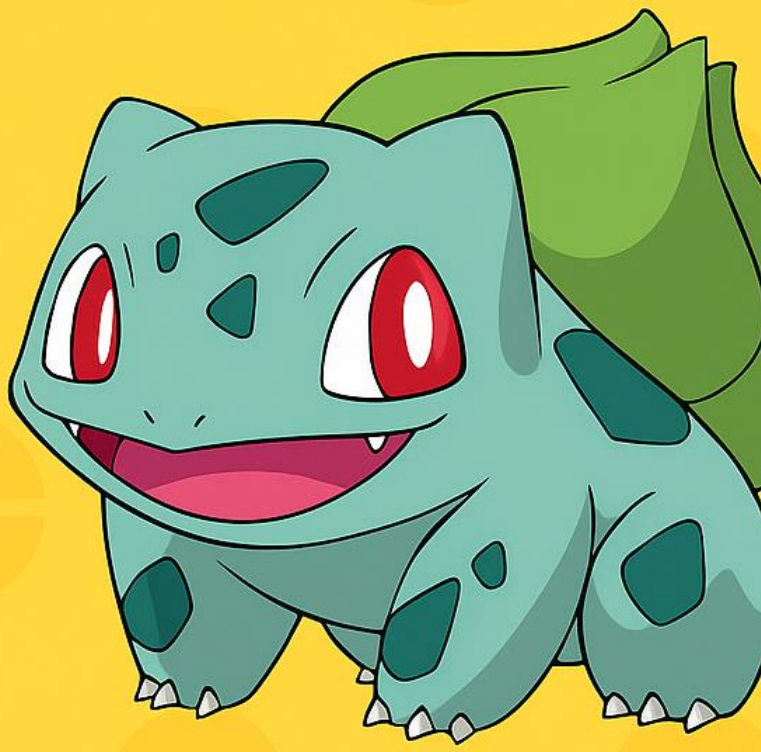
[Usuario]
↓
[Rutas]
↓
[Middleware]
↓
[Controlador]
↓
[Respuesta]



EJS: vistas dinámicas en Node.js

- EJS permite mostrar datos dinámicamente en las pantallas.

- Las plantillas EJS reciben datos desde el controlador y los despliegan en HTML.

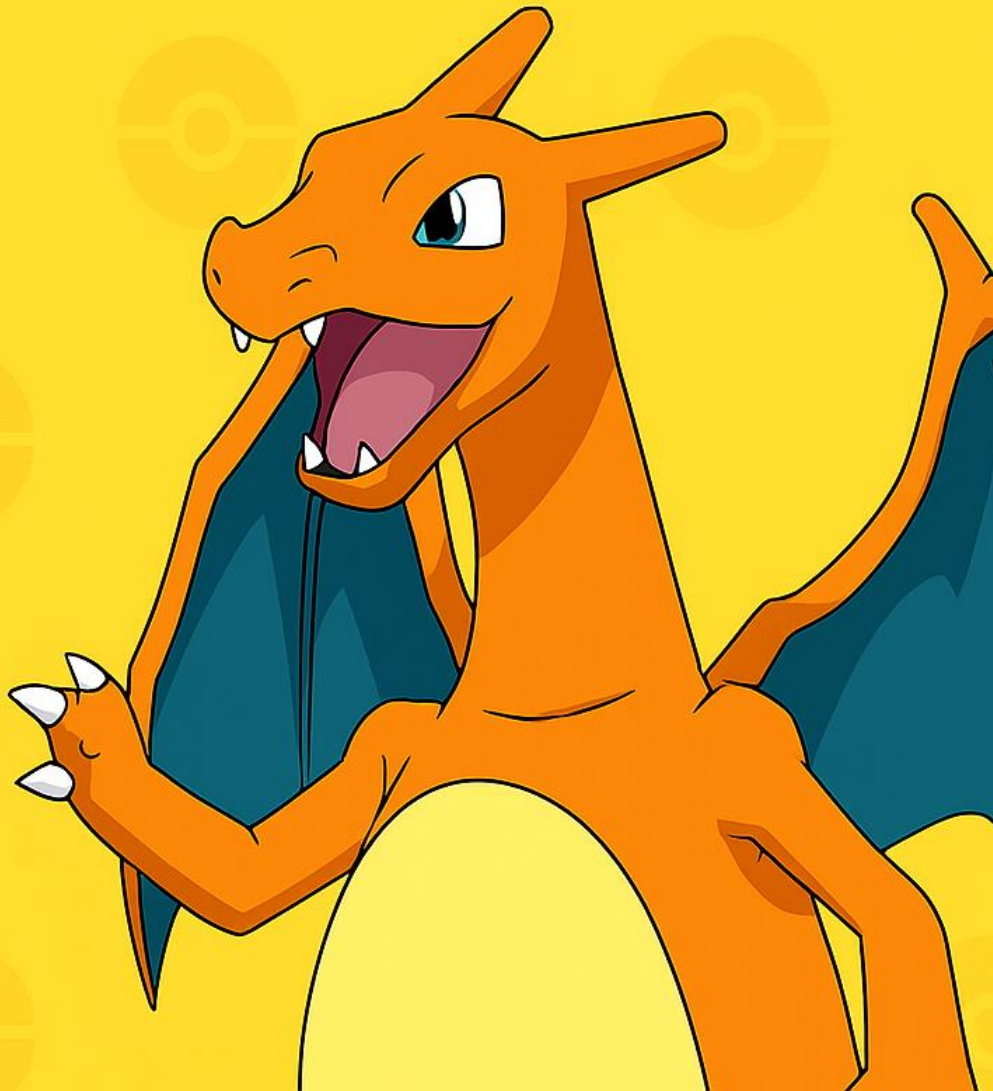


Explicación de código: EJS

- Ejemplo: /views/productos/lista.ejs

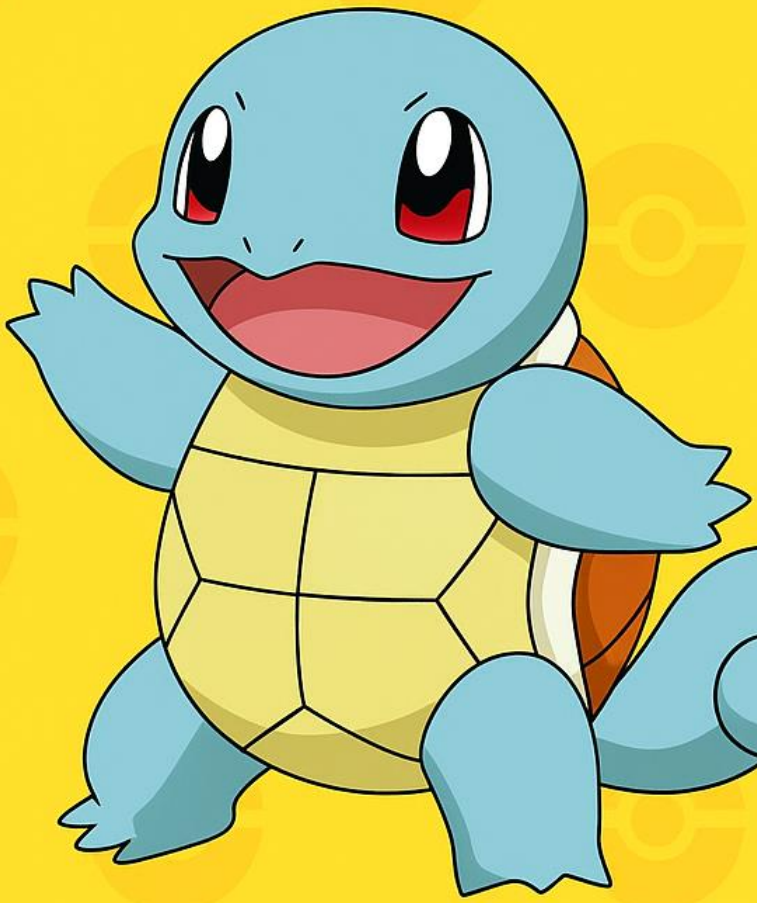
-
- <% productos.forEach(p => { %>
- <%= p.nombre %> - $<%= p.precio %>
- <% }) %>
-

- →
Genera una lista dinámica de pr oductos desdeel backend.



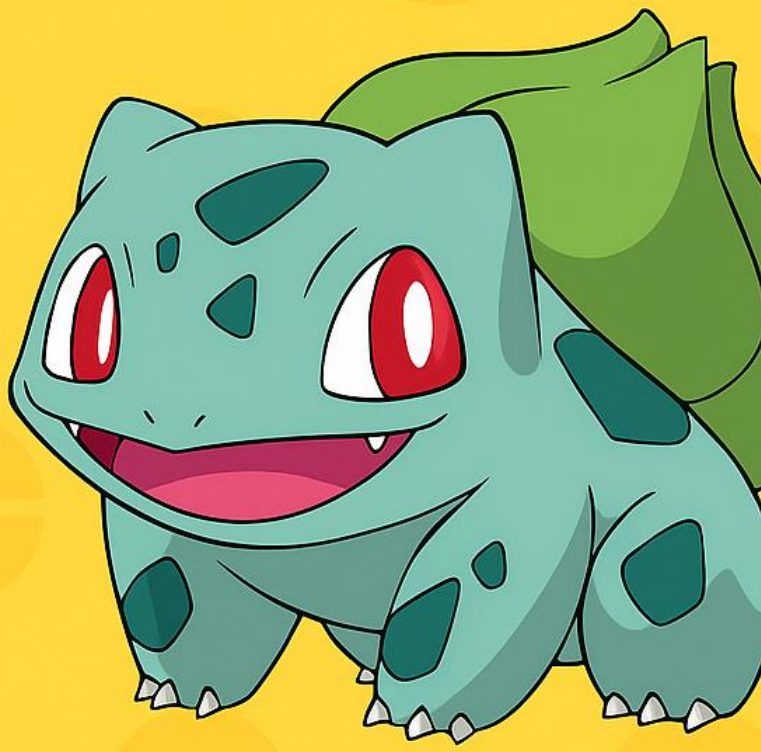
Flujo de EJS

[Usuario]
↓
[Controlador]
↓
[Renderiza EJS]
↓
[HTML generado]
↓
[Usuario]



Dashboard

- Dashboard: pantalla principal del administrador.
- •Resume ventas, usuarios y produc tos
- • Acceso: /admin/dashboard (solo admin)
- • Junta información de toda la app en una vista central
- • Protegido por middleware de autenticación.

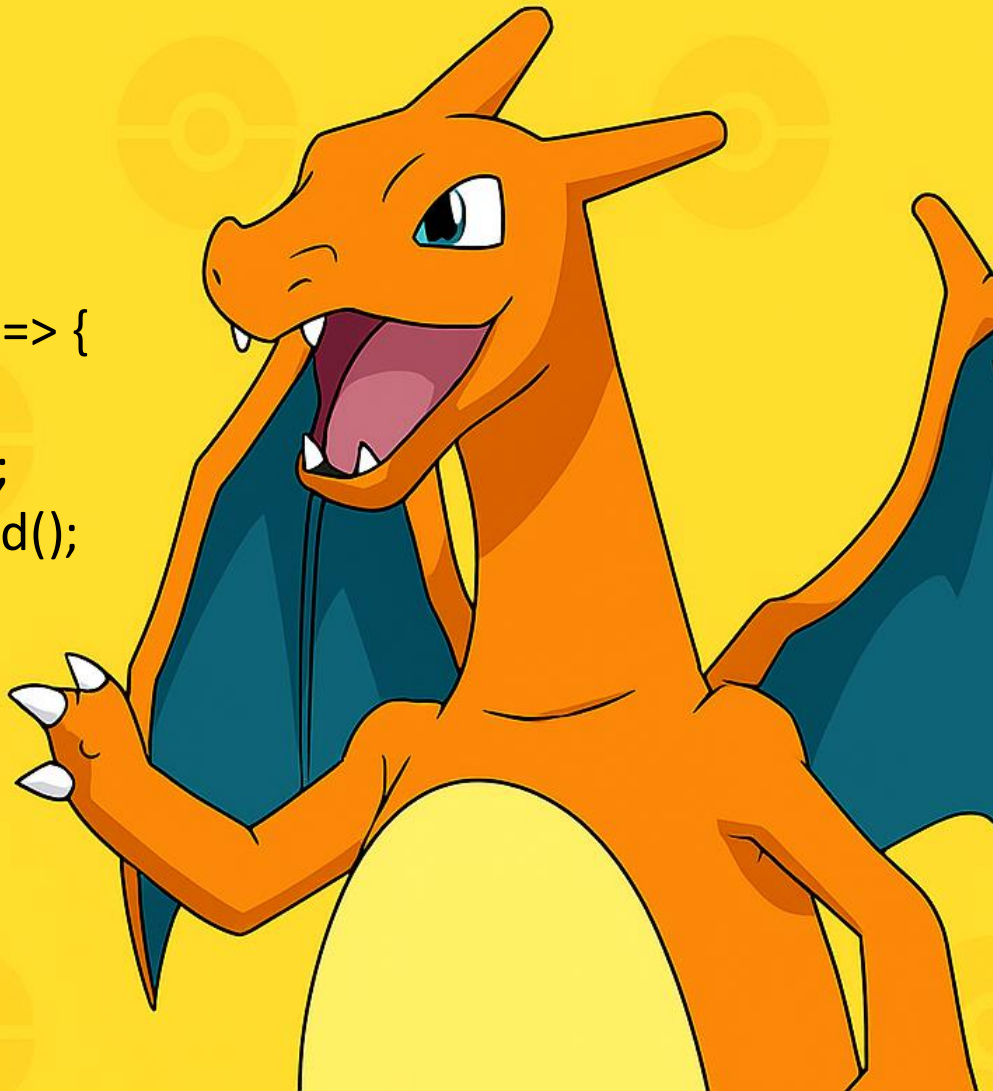


Explicación de código: Dashboard

- Ruta: /routes/admin.js
- Controlador: /controllers/adminController.js
- Vista: /views/admin/dashboard.ejs

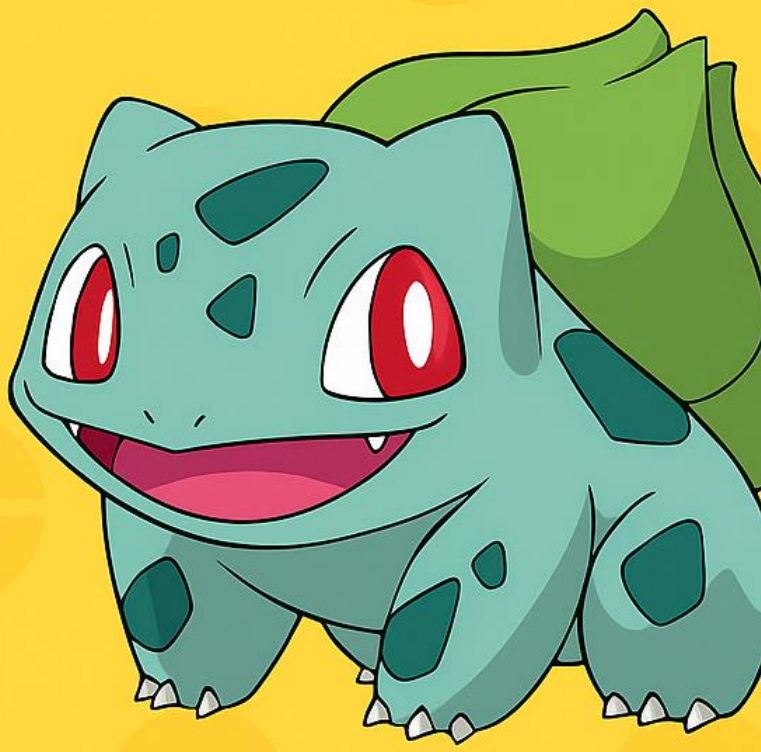
- exports.dashboard = async (req, res) => {
- const ventas = await Venta.find();
- const usuarios = await Usuario.find();
- const productos = await Producto.find();
- res.render('admin/dashboard', { ventas, usuarios,productos });
- };

- → Junta info de toda la app y la muestra al admin.



Variables de entorno (.env)

- Archivo oculto (.env) en la raíz.
- Guarda datos sensibles y configuraciones como:
- - Claves
- - Usuarios
- - Contraseñas
- - Puertos
- Uso: process.env.NOMBRE
- Permite cambiar settings sin tocar el código.



Endpoints

- Cada endpoint es una URL clara para una acción.
- Ejemplo: GET /api/productos, POST /api/auth/login.
- Permite separar frontend y backend y mantener el sistema ordenado.

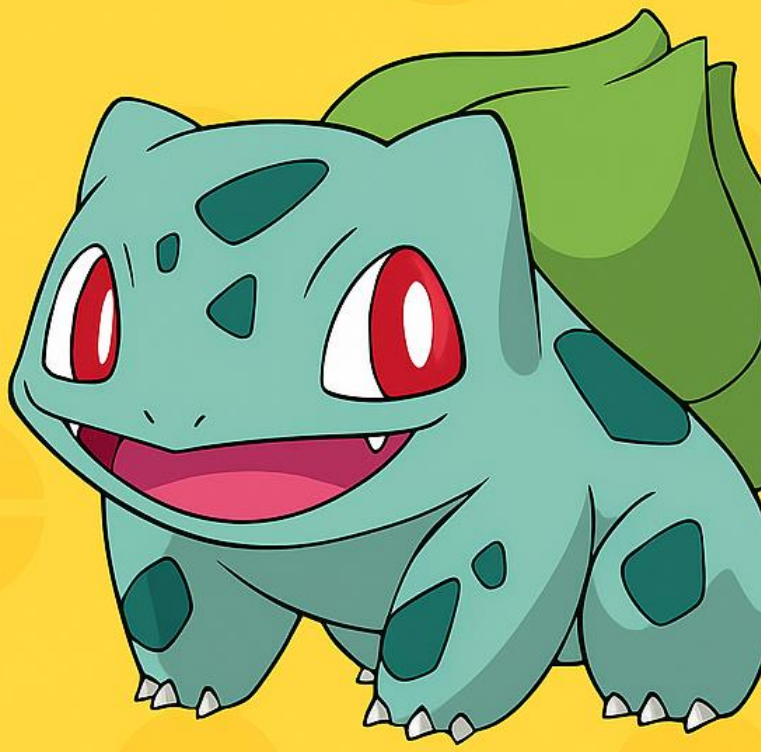
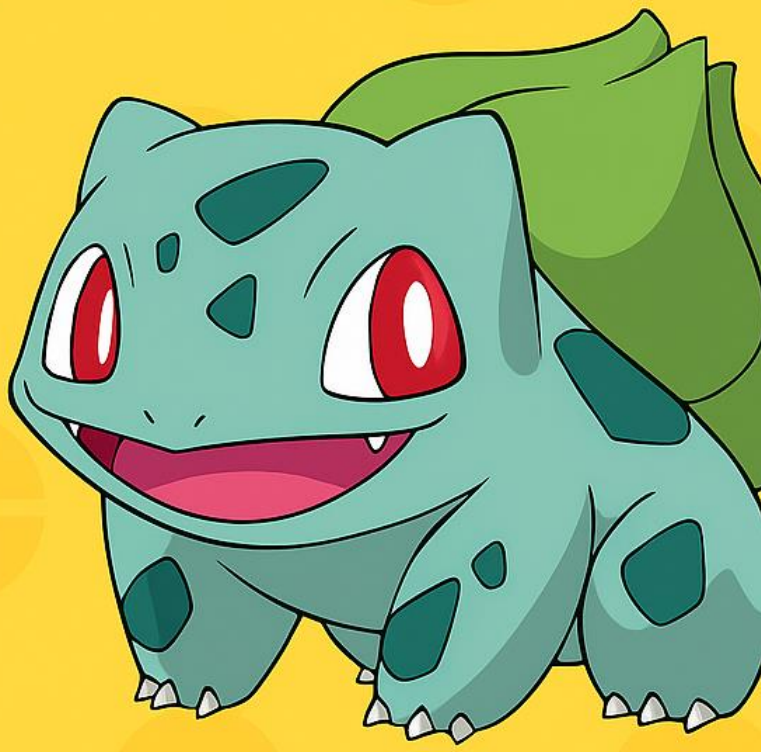


Tabla resumen de estructura

Carpeta/Archivo	Función Principal
/routes/productos.js	Define rutas/acciones CRUD de productos
/controllers/productoController.js	Lógica de negocio; maneja peticiones
/models/producto.js	Estructura y métodos de productos
/middleware/auth.js	Middleware para autenticación
/views/productos/lista.ejs	Vista dinámica: lista de productos
/views/layouts/main.ejs	Plantilla base
/routes/usuarios.js	Rutas de usuarios
/controllers/usuarioController.js	Lógica de usuarios
/controllers/adminController.js	Lógica del Dashboard

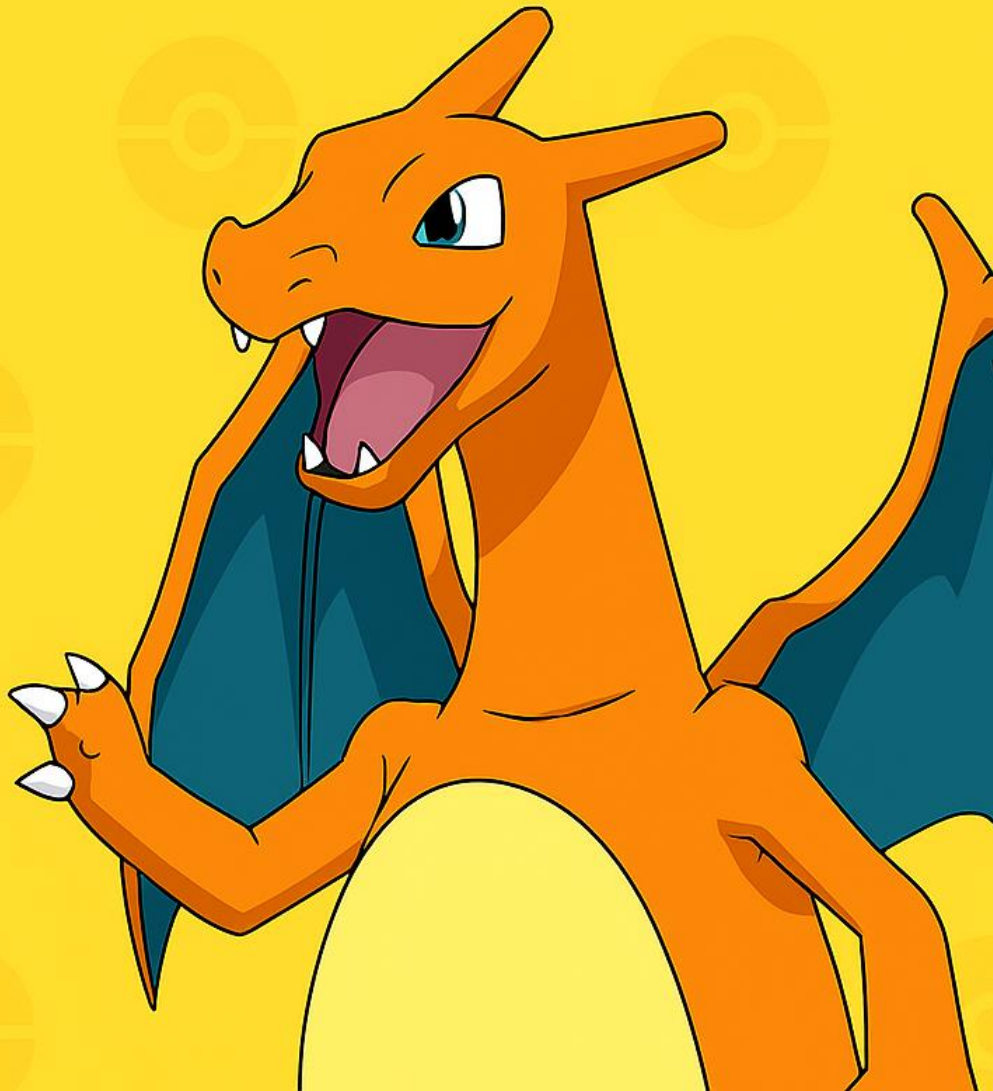
Rutas (Routes) y su estructura

- Las rutas definen los camino s disponibles en laapp.
- Ejemplo: /routes/productos.js
- Conectan URLs con métodos de los controladores.



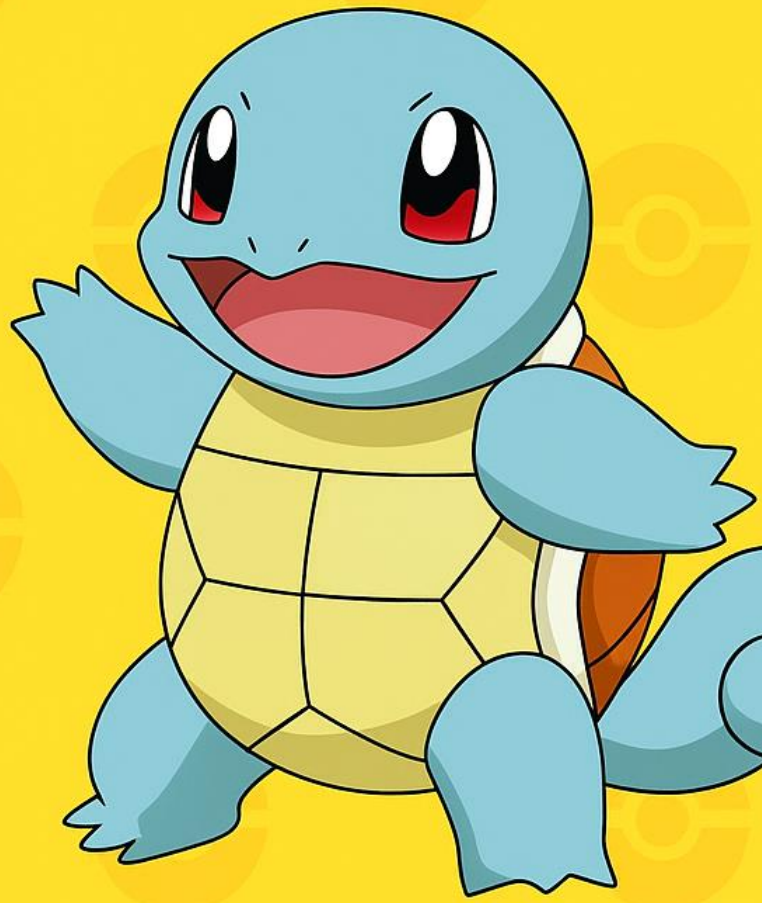
Explicación de código: Rutas

- Ejemplo:
- router.get('/', productoController.listarProductos);
- router.post('/', productoController.crearProducto);
- Cada acción tiene su URL y su función asociada.



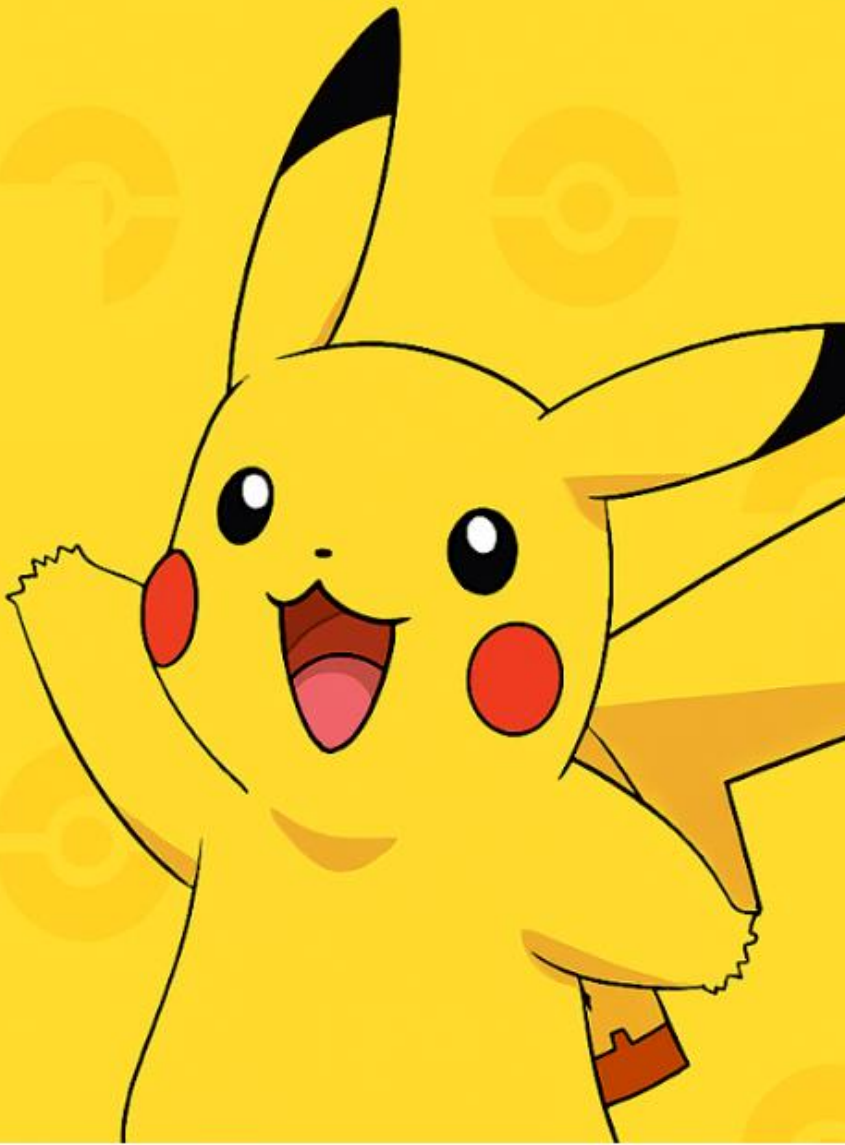
Flujo de Rutas

[Usuario]
↓
[Ruta]
↓
[Controlador]
↓
[Modelo]
↓
[Vista]
↓
[Usuario]



Ejemplo real: Crear producto

- 1.Usuario envía formulario en/views/productos/alta.ejs
- 2. Ruta POST /api/productos recibe la petición
- 3. Controlador productoController.js valida y guarda
- 4. Modelo producto.js almacena en la base
- 5. Vista muestra el resultado al usuario
- router.post('/', productoController.crearProducto)
- exports.crearProducto = (req, res) => { ... }



Importancia de cada componente

- • Rutas: Ordenan los caminos posibles, facilitanescalabilidad y claridad.
- • Controladores: Centralizan la lógica, fácil demantener y extender.
- • Modelos: Definen estructura y gestión de datos,facilitan cambios.
- • Middleware: Reutiliza lógica común comovalidación y autenticación.
- • Vistas: Separan diseño de lógica, permitemodificar interfaz sin tocar ba ckend.
- → Organización modular: mantenibilidad ytrabajo en equipo.

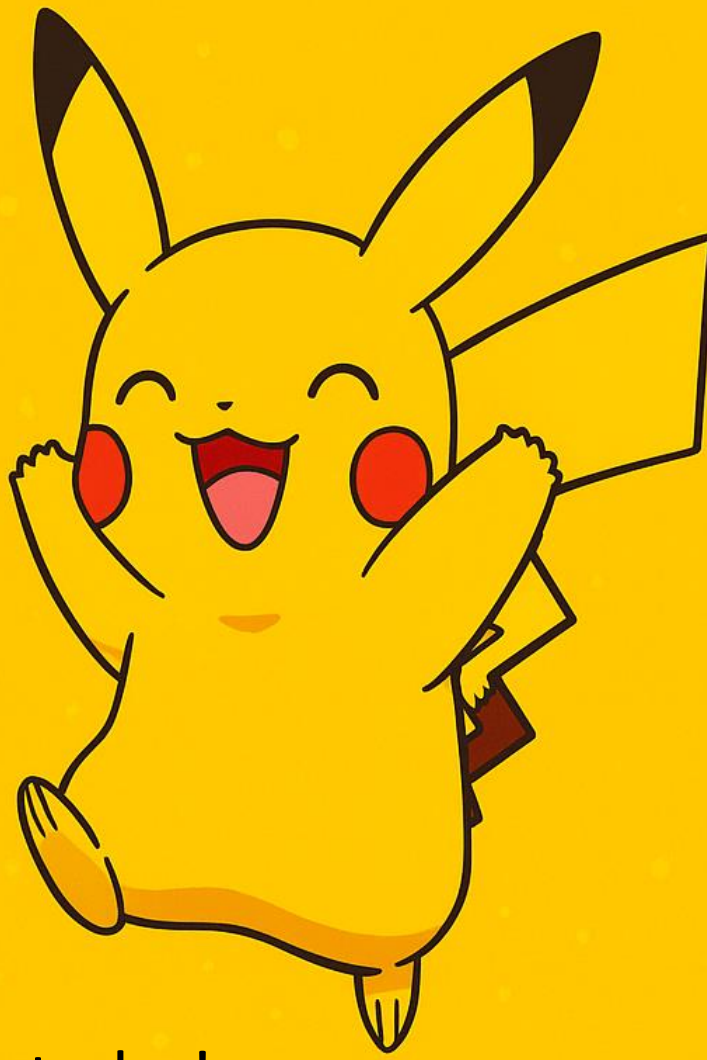
Conclusiones

- El proyecto usa arquitectura MVC
- CRUD en todas las entidades
- Middleware refuerza seguridad
- EJS permite vistas dinámicas
- Dashboard centraliza info clave
- Rutas organizadas por recurso
- Código mantenible y profesional
- Estándar Node.js/Express

Profe Gabi y Javi

TP Integrador

okémon Business



- ¡Gracias profesores por la excelente catedra!