



# De la Idea a la Realidad

Construyendo tu Primera Aplicación  
con Angular Moderno

# El Plan Maestro: ¿Qué es Angular?

Angular es un framework de Google para construir aplicaciones web modernas.  
Su filosofía se basa en tres pilares clave.

---



## SPA (Single Page Application)

Crea aplicaciones web dinámicas, asíncronas y reactivas. La clave es una experiencia fluida, sin recargas de página.



## Cliente Front-end

Con Angular construimos la parte del cliente (Front-end), la interfaz con la que interactúa el usuario, sin relación directa con el Back-end.



## Experiencia de Usuario

Los cambios se reflejan al instante. No es necesario refrescar o recargar la página para ver nuevas actualizaciones.

# Preparando el Terreno: Instalación y Configuración

## Requisito Fundamental

Es indispensable tener instalado **Node.js** en tu sistema.

Tarea	Comando	Descripción
Instalar CLI	<code>npm install -g @angular/cli</code>	Instala la Interfaz de Línea de Comandos (CLI) de Angular de forma global. El comando `ng` será tu herramienta principal.
Comprobar Versión	<code>ng version</code>	Verifica que la instalación se ha completado correctamente y muestra la versión del CLI.

# Poniendo la Primera Piedra: Crear y Arrancar tu App

## Paso 1: Crear el Proyecto

### `ng new`: El Comando de Creación

```
ng new nombre-app
```

#### Notas Importantes (Opciones Recomendadas)

- ✓ Estilo: Elegir **CSS**.
- ✓ Server-Side Rendering (SSR): Decir **NO**.
- ✓ Herramientas AI: Decir **NO**.

**Resultado:** Se crea una nueva carpeta con toda la estructura del proyecto. Se recomienda abrirla con Visual Studio Code.

## Paso 2: Arrancar la Aplicación

### `ng serve`: Compilar y Ejecutar

```
ng serve
```

**Resultado:** Compila la app y la ejecuta en un servidor local, típicamente en `http://localhost:4200/`.

# Los Ladrillos Fundamentales: El Modelo de Componentes

Angular funciona completamente basado en componentes. Un componente es un bloque independiente de lógica, vista y estilo.

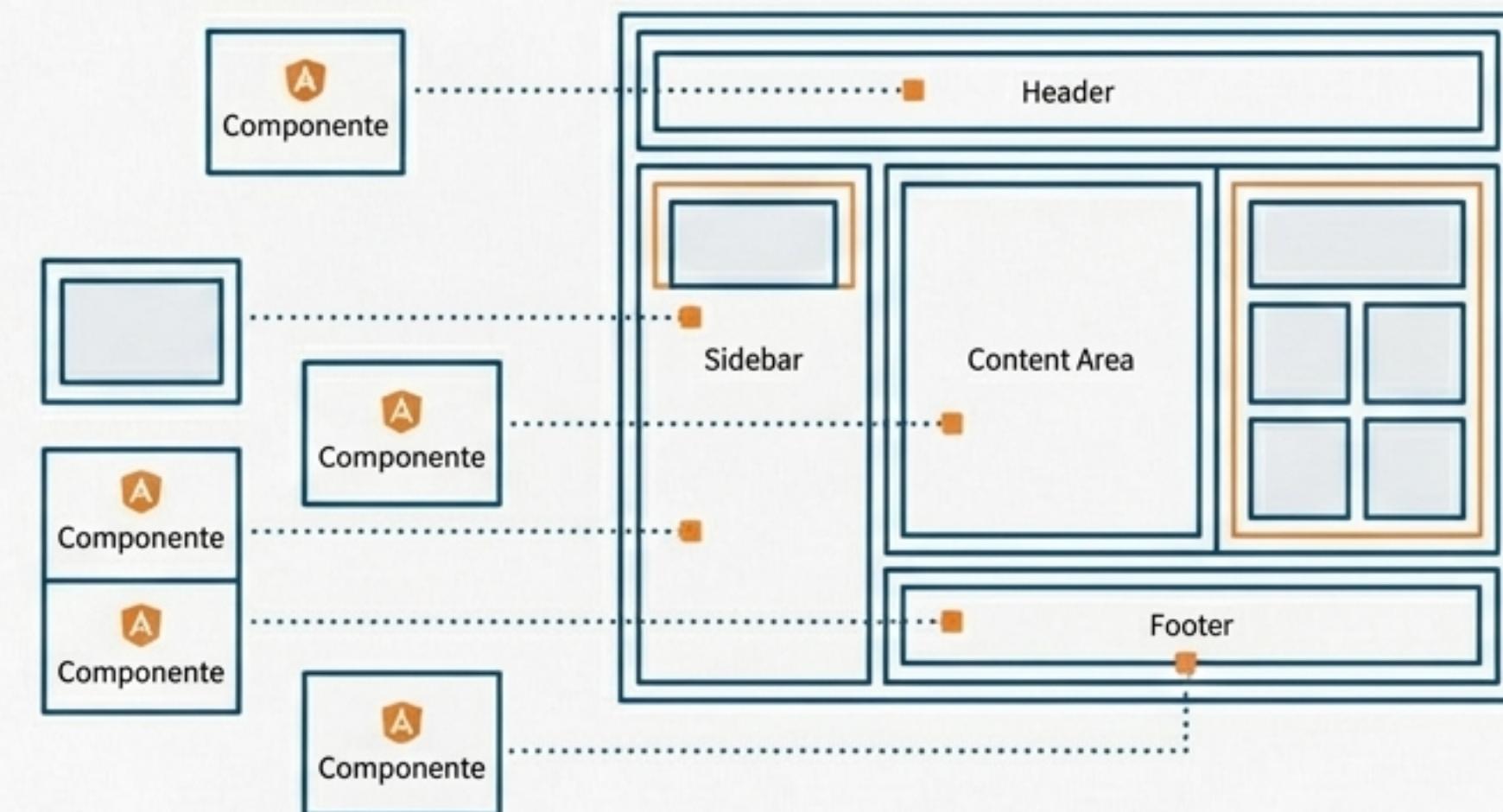
## El Comando para Crear Ladrillos

```
bash
# Versión larga
ng generate component nombre-componente

# Versión corta
ng g c nombre-componente
```

Este comando crea una nueva carpeta con todos los archivos necesarios para el componente: `.html`, `.ts` y `.css`.

La carpeta principal de trabajo es `src/app`. Aquí es donde residen todos tus componentes.



# Anatomía de un Componente

## Los Metadatos: Decorador (@Component)

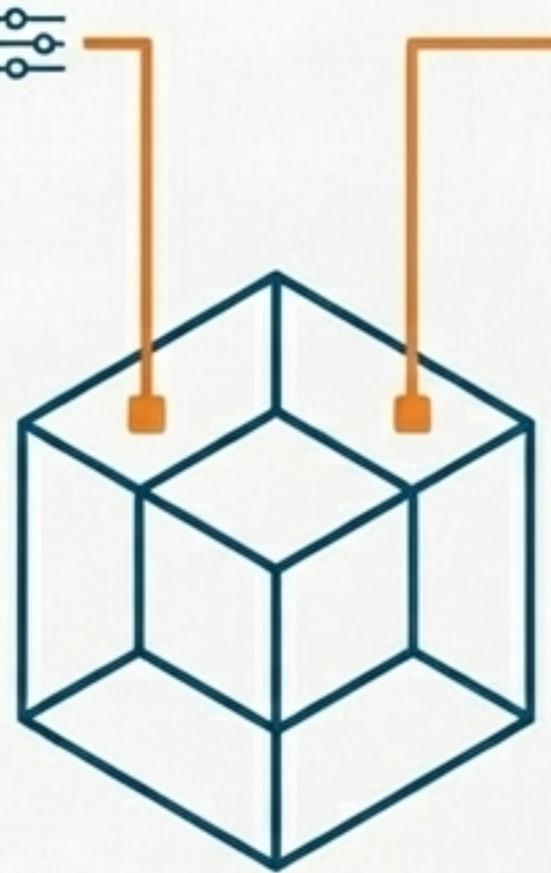
Es un bloque en el archivo `.ts` que une todo. Define metadatos cruciales como el selector, la ruta a la plantilla (`templateUrl`) y la hoja de estilos (`styleUrl`). Cada componente puede tener sus propios estilos.

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

## La Vista: Plantilla HTML (.html)

Archivo de ejemplo: `app.component.html`

Es la estructura visual del componente; lo que se renderiza en el navegador. Para mostrar una variable desde la clase, se usa la sintaxis de doble llave: ``{{ nombre_variable }}``.



## La Lógica: Clase TypeScript (.ts)

Archivo de ejemplo: `app.component.ts`

En `export class`, se definen las propiedades (variables) y la lógica del componente. Estos datos estarán disponibles para ser mostrados en la vista.

```
export class AppComponent {
  title = 'mi-primer-app';
  // Lógica y métodos aquí
}
```

```
<div>
  <h1>Bienvenido a {{ title }}!</h1>
</div>
```

# El Tapiz Principal: `AppComponent`

Piensa en ``app.component.html`` y ``app.component.ts`` como el ‘tapiz’ principal o el contenedor raíz de tu aplicación. Es la base sobre la que se construye todo lo demás.



El `app.component.html` contiene el layout principal, y `app.component.ts` maneja la lógica global.

# Ensamblando la Estructura: Colocando Componentes en el Tapiz

## 1. Paso 1: Registrar el Componente

Para usar un componente nuevo (ej. `MenuComponent`), primero debes importarlo en el archivo `ts` del componente que lo contendrá (el 'tapiz').

```
app.component.ts Fira Code
// 1. Importar el componente
import { MenuComponent } from './menu/menu.component';

@Component({
  selector: 'app-root',
  standalone: true,
  // 2. Añadirlo al array de imports
  imports: [MenuComponent],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent { ... }
```

## 2. Paso 2: Usar el Selector

Una vez registrado, usa el selector del componente (ej. `app-menu`) como si fuera una etiqueta HTML estándar dentro de la plantilla del 'tapiz'.

```
app.component.html Fira Code
<header>
  <h1>Mi Aplicación de Prueba</h1>
  <app-menu></app-menu>
</header>
<main>
  <!-- Contenido principal -->
</main>
```

Conclusión: Esta modularidad permite construir aplicaciones complejas ensamblando componentes pequeños e independientes.

# Lógica Constructiva: El Nuevo Control Flow

El Control Flow es la sintaxis nativa y optimizada de Angular para manejar la lógica de la plantilla (condicionales y bucles) directamente en el HTML.



## Principales Estructuras

- **`@if`**: Para mostrar u ocultar elementos del DOM según una condición.
- **`@for`**: Para iterar sobre colecciones de datos (**arrays**) y renderizar una lista de elementos.

# Control Flow en Acción: `@if` y `@for`

## Estructuras Condicionales

`@if`, `@else if`, `@else`

app.component.ts

```
Fira Code  
 @if (usuarioLogueado) {  
   <div>Bienvenido, usuario!</div>  
 } @else if (intentosRestantes > 0) {  
   <div>Por favor, introduce tus credenciales.</div>  
 } @else {  
   <div>Cuenta bloqueada.</div>  
 }
```

## Listas y Bucles

`@for` y `@empty`

app.component.html

```
Fira Code  
 <ul>  
   @for (item of items; track item.id) {  
     <li>{{ item.nombre }}</li>  
   } @empty {  
     <li>No hay elementos en la lista.</li>  
   }  
 </ul>
```



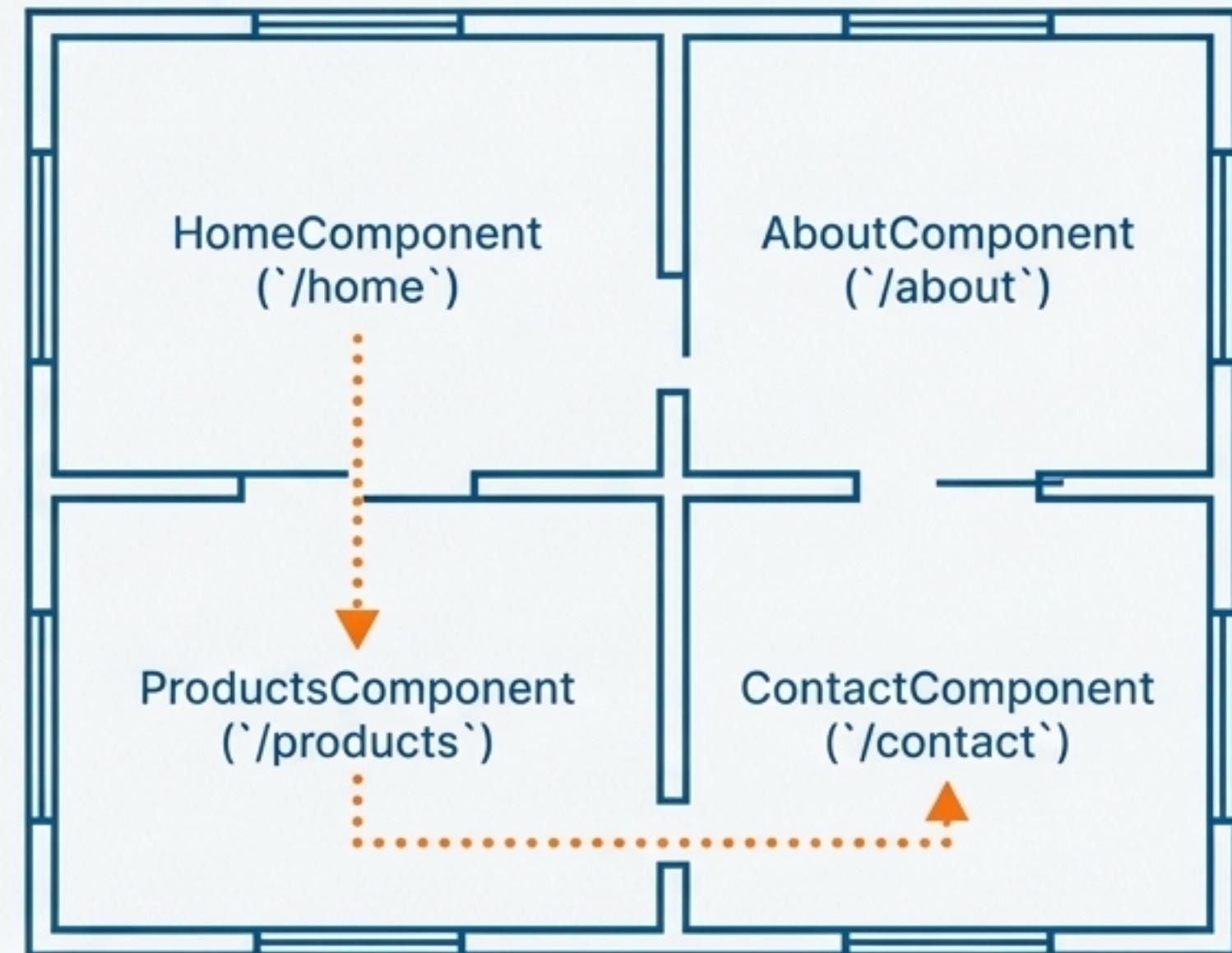
Nota Crítica: La cláusula `track item.id` es **obligatoria**. Ayuda a Angular a optimizar el renderizado de listas identificando únicamente cada elemento.

# Trazando los Caminos: Introducción al Enrutamiento

Los enrutadores (routers) gestionan la navegación entre diferentes vistas o componentes dentro de una SPA, sin necesidad de recargar toda la página.

## Funcionamiento Clave

- Se asigna una URL específica a cada componente (ej. `/home` carga `HomeComponent`).
- Al hacer clic en un enlace, Angular cambia la vista dinámicamente sin refrescar el navegador.



# Los Mecanismos del Enrutamiento

## Componentes Clave del Router

### 1. Definición de Rutas

Se configura un array donde cada objeto asocia un `path` (la URL) con un `component` (el componente a cargar).

### 2. El Marcador de Posición: `<router-outlet>`

Es una directiva que se coloca en tu plantilla principal (como `app.component.html`). Actúa como un marcador de posición donde se renderizará el componente correspondiente a la ruta actual.

**\*\*Configuración\*\*:** Típicamente, el enrutamiento se configura en su propio archivo (AppRoutingModule) para mantener el código organizado.

### 3. La Navegación: `'routerLink'`

Se usa en lugar de `href` en las etiquetas `` para navegar. Evita la recarga de la página.

Fira Code

```
<a routerLink="/users">Usuarios</a>
```

### 4. Estilo Activo: `'routerLinkActive'`

Una directiva que añade automáticamente una clase CSS a un enlace cuando su ruta está activa. Útil para resaltar la página actual en un menú de navegación.

# La Gran Inauguración: Despliegue Automatizado

**El Desafío:** Subir los cambios a GitHub y desplegar la aplicación requiere múltiples comandos manuales.

**La Solución:** Automatizar el proceso con scripts en el archivo `package.json`.

```
Fira Code  
"scripts": {  
  "deploy": "ng build --base-href /repo-name/ && npx angular-cli-ghpages  
--dir=dist/project-name/browser",  
  "deploy-full": "git add . && git commit -m \"Update\" && git push && npm run deploy"  
},
```

\*Note: The script paths like `/repo-name/` and `dist/project-name/browser` are examples and should be adapted to the specific project.\*

# Ejecutando el Despliegue

Una vez definidos los scripts en `package.json`, puedes ejecutarlos desde tu terminal con `npm run`.

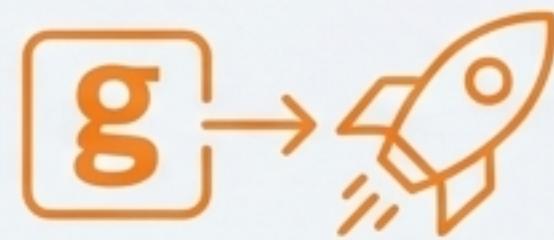


## Opción 1: Despliegue Rápido

```
npm run deploy
```

¿Qué hace?

Solo construye la aplicación (ng build) y la publica en GitHub Pages. Ideal para cuando ya has subido tus cambios a Git.



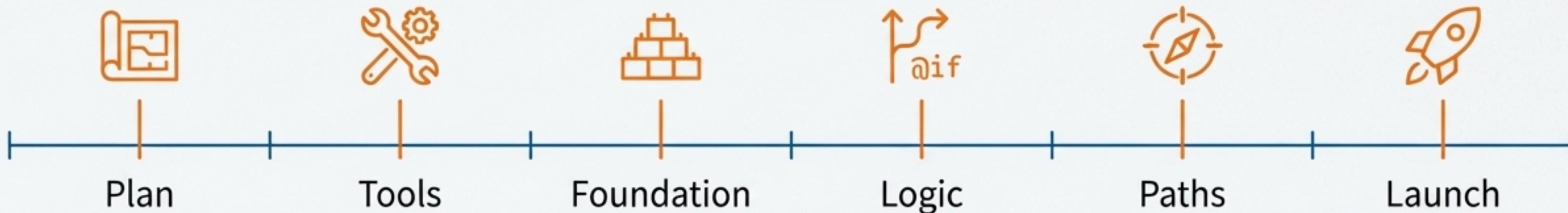
## Opción 2: Despliegue Completo

```
npm run deploy-full
```

¿Qué hace?

Realiza el ciclo completo. Sube todos los cambios a GitHub (`git add`, `commit`, `push`) y luego ejecuta el script `deploy` para construir y publicar la aplicación. Es una copia de seguridad y despliegue en un solo paso.

# De la Idea a la Realidad: Resumen del Viaje



## Puntos Clave

- **Arquitectura de Componentes:** Angular permite crear SPAs modernas basadas en componentes modulares y reutilizables.
- **Flujo de Trabajo Eficiente:** El CLI (`ng`) simplifica la creación, ejecución y generación de código.
- **Lógica de Plantilla Moderna:** El nuevo Control Flow (`@if`, `@for`) hace que la lógica en el HTML sea limpia y optimizada.
- **Navegación Fluida:** El sistema de enrutamiento es fundamental para una experiencia de usuario sin interrupciones.
- **Despliegue Sencillo:** Los scripts de automatización permiten publicar tu aplicación completa con un solo comando.