CLASIFICACION DE LOS MEDIOS DE PUBLICACION DE UNA APP WEB

Jorge Saucedo Flores
Josué Abdi González Garcia
Daray Palomino Macías
Angel Yassel Soriano Gutiérrez

Tenemos nuestra aplicación web terminada en nuestra laptop... ¿cómo hacemos para que el mundo la vea en www.mi-app.com?

Lo primordial es saber que "publicar" no es solo copiar archivos. Implica:

- Un servidor (hardware).
- Un sistema operativo.
- Un servidor web (Apache, Nginx).
- Un entorno de ejecución (Node.js, PHP, Java).
- Una base de datos.
- Un dominio y un certificado SSL.

Objetivo de la Exposición:

"Hoy vamos a clasificar los 5 modelos principales que existen para publicar una aplicación web, desde el más complejo (hacerlo todo tú mismo) hasta el más simple (subir solo el código)."

Agenda:

- Modelo 1: On-Premise (Alojamiento Propio)
- Modelo 2: IaaS (Infraestructura como Servicio)
- Modelo 3: PaaS (Plataforma como Servicio)
- Modelo 4: Serverless / FaaS (Funciones como Servicio)
- Modelo 5: Alojamiento Estático
- Tabla Comparativa y Conclusiones.

Modelo 1: On-Premise (Alojamiento Propio)

Concepto: Tú eres el dueño de todo. Compras el hardware, instalas el SO, gestionas la red, la electricidad y el aire acondicionado.

Analogía: Construir tu propia casa desde cero. Tienes que comprar el terreno, poner los cimientos, instalar la tubería y la electricidad.

¿Qué gestionas tú? TODO.

- Hardware (Servidores, Discos Duros, Red).
- Sistema Operativo.
- Servidor Web (Apache, Nginx).
- Runtime (Node.js, Java).
- Código de la aplicación.

Ventajas:

- Control Total: Tienes control absoluto sobre el hardware y el software.
- Seguridad (Potencial): Los datos nunca salen de tus instalaciones (ideal para bancos, gobiernos).

Desventajas:

- Costo Inicial Elevado: Comprar servidores es caro.
- Mantenimiento Constante: Necesitas un equipo de TI 24/7.
- Poca Escalabilidad: ¿Necesitas más potencia? Tienes que comprar y configurar un nuevo servidor (tarda semanas).

Ejemplos: Centros de datos de bancos, agencias gubernamentales, empresas muy grandes antes de la nube.

Este es el modelo "antiguo" y el más complejo, pero sirve como base para entender los demás.

Modelo 2: IaaS - Infraestructura como Servicio

Concepto: Alquilas el hardware (servidores virtuales) en la nube. Ya no compras el servidor físico, pero sigues gestionando el software.

Analogía: Alquilar el terreno y las herramientas. No construyes el edificio, pero sí tienes que gestionar las paredes, la pintura y los muebles.

¿Qué gestionas tú?

- Sistema Operativo (Instalar parches).
- Servidor Web.
- Runtime.
- Código de la aplicación.

¿Qué gestiona el Proveedor?

- El hardware físico, la virtualización, la red.

Ventajas:

- Flexibilidad: "Pago por uso" (Pay-as-you-go).
- Escalabilidad Rápida: Puedes "pedir" un nuevo servidor en minutos.

Desventajas:

- Sigue siendo complejo: Necesitas un SysAdmin (Administrador de Sistemas) para configurar la seguridad, las redes, las bases de datos y mantener el SO actualizado.

Ejemplos Clave:

- Amazon EC2
- Google Compute Engine
- Azure Virtual Machines

Modelo 3: PaaS - Plataforma como Servicio

Concepto: El proveedor te da una plataforma lista para usar. Tú solo te preocupas por tu código y tus datos.

Analogía: Alquilar un apartamento amueblado. Ya tiene electricidad, agua, muebles. Tú solo traes tu ropa (tu código).

¿Qué gestionas tú?

- Código de la aplicación.
- Datos.

¿Qué gestiona el Proveedor?

- Hardware.
- Sistema Operativo.
- Servidor Web.
- Runtime (tú eliges: "quiero correr esto en Node.js 18").
- Escalado automático.

Ventajas:

- Enfoque en el Desarrollo: El desarrollador no se preocupa por la infraestructura.
- Rápida Puesta en Marcha: Puedes publicar una app en minutos.

Desventajas:

- Menos Control: No puedes modificar el Sistema Operativo o el servidor web.
- "Vendor Lock-in": Atado al proveedor. Moverse de Heroku a AWS puede ser difícil.

Ejemplos Clave:

- Heroku (El ejemplo clásico)
- AWS Elastic Beanstalk
- Google App Engine

Modelo 4: Serverless (FaaS - Funciones como Servicio)

Concepto: Es la evolución del PaaS. Ni siquiera publicas una "aplicación", publicas "funciones" individuales que responden a eventos.

Analogía: Pagar un Uber para un solo viaje. No alquilas el coche (PaaS), solo pagas por el trayecto exacto que necesitas, cuando lo necesitas.

¿Qué gestionas tú?

- El código de una función (ej. function procesarPago() { ... }).

¿Qué gestiona el Proveedor?

- Todo lo demás. La función "despierta" cuando se la llama y "duerme" cuando termina.

Ventajas:

- Costo Extremadamente Bajo: Solo pagas por los milisegundos que tu función se ejecuta.
- Escalabilidad Infinita: Si 1 millón de usuarios llaman tu función al mismo tiempo, el proveedor la ejecutará 1 millón de veces en paralelo.

Desventajas:

- Complejidad de Arquitectura: Requiere pensar en "eventos" y microservicios.
- "Cold Starts": La función puede tardar un segundo en "despertar" si no se ha usado recientemente.

Ejemplos Clave:

- AWS Lambda
- Google Cloud Functions
- Azure Functions

Modelo 5: Alojamiento Estático (Static Hosting)

Concepto: Un servicio especializado para aplicaciones web que no tienen backend. (HTML, CSS, JavaScript puro, o *builds* de React/Vue/Angular).

Analogía: Publicar un documento en una biblioteca pública. Es solo para lectura; no puedes "procesar" nada en el servidor.

¿Qué gestionas tú?

- Tus archivos HTML, CSS y JS.

¿Qué gestiona el Proveedor?

- Un servidor web optimizado (CDN) que distribuye tus archivos por todo el mundo.

Ventajas:

- Velocidad Extrema: Los archivos se sirven desde el (CDN) más cercano al usuario.
- Casi Siempre Gratis: Muchos servicios ofrecen planes gratuitos muy generosos.
- Máxima Seguridad: No hay base de datos ni servidor que hackear.

Desventajas:

- Solo para sitios estáticos. Si necesitas una base de datos o lógica de servidor, debes combinarlo con FaaS (Serverless).

Ejemplos Clave:

- Netlify
- Vercel
- GitHub Pages

Comparativa y Criterios de Decisión

Modelo	¿Qué Gestionas?	Costo	Escalabilidad	Facilidad
On-Premise	TODO	Muy Alto	Difícil	Muy Difícil
IaaS	SO, Servidor, Código	Variable (Medio)	Manual	Difícil
PaaS	Código, Datos	Variable (Alto)	Automática	Fácil
Serverless	Solo Funciones	Muy Bajo (pago/ms)	Infinita	Fácil (código) / Difícil (arq.)
Static	Archivos (HTML/JS)	Gratis / Muy Bajo	Automática	Muy Fácil

¿Cómo elegir el servicio correcto?

- ¿Es un blog o un portafolio? → Static Hosting (Netlify, Vercel).
- ¿Eres un desarrollador solo o un startup probando una idea? → PaaS (Heroku).
- ¿Necesitas control total sobre el SO (ej. instalar software específico)? → IaaS (AWS EC2).
- ¿Tu tráfico es impredecible y quieres pagar solo por uso? → Serverless (AWS Lambda).
- ¿Eres un banco o una agencia de inteligencia? → On-Premise.

Conclusión

Resumen:

- Vimos que "publicar" una app ha evolucionado.
- Pasamos de gestionar hardware (On-Premise) a gestionar VMs (IaaS).
- Luego, a gestionar solo código (PaaS).
- Y finalmente, a gestionar solo funciones (Serverless) o archivos estáticos.

Tendencia: La industria se mueve hacia la abstracción. Los desarrolladores quieren enfocarse en el código, no en los servidores.

Cierre: "La próxima vez que uses una aplicación web, pregúntate: ¿cómo estará alojada? ¿Será un PaaS, un IaaS o Serverless?"

Bibliografía

Amazon Web Services. (2025). *Tipos de Cloud Computing*. AWS.

Google Cloud. (2025). *Compute Engine (IaaS)*.

Heroku. (2025). *What is a PaaS?*. Salesforce.

Netlify. (2025). *Static vs. Dynamic Websites*.**taja**