**Curso de Introducción al Despliegue de Aplicaciones**

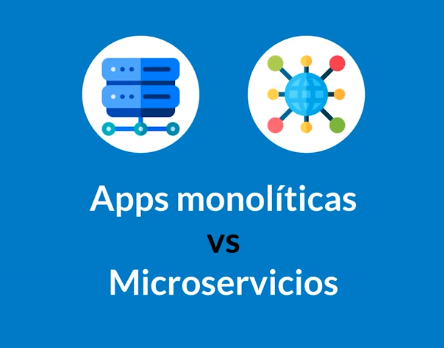
**Santiago Bernal**

**Platzi**

2/17 Historia de las aplicaciones

En los años 70, las aplicaciones eran de escritorio solamente tenían su base de datos o sistema de ficheros para guardar la información en la aplicación por si misma, y la interfaz gráfica era CLI o la terminal.  
Con el paso de los años comenzó a surgir un modelo llamado cliente servidor el cual recibe conexiones de muchos clientes, los cuales acceden al servidor para intercambiar información.  
Actualmente tenemos aplicaciones web que hasta hace unos anos eran monolíticas ya que todo el paquete de programación necesaria para la aplicación (FrontEnd, BackEnd, Base de Datos) iban empaquetadas en un solo servidor, sin embargo actualmente se ha comenzado a usar una técnica que involucra algo llamado [microservicios](https://www.redhat.com/es/topics/microservices).

**Apps Monolíticas vs microservicios**3/17

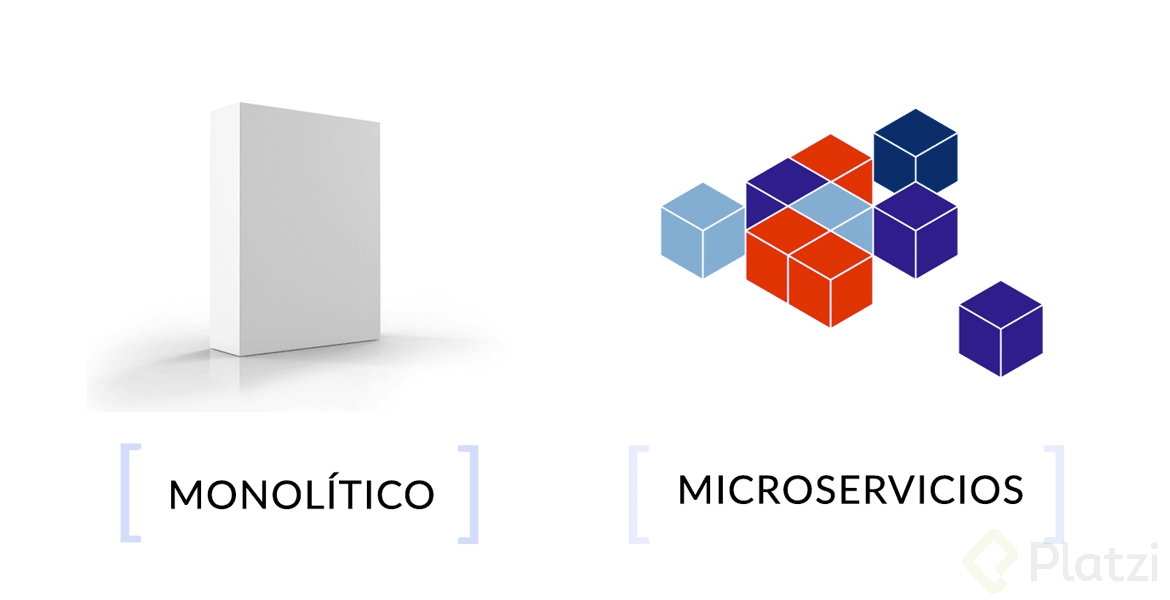


*bajo acoplamiento, alta cohesión*

**Aplicaciones Monolíticas vs Microservicios**

Monolíticas basadas en PHP y Perl, eran poco robustas y mantenibles. Todo dependía de un solo servidor.

Luego se inició con “bajo acoplamiento-alta cohesión” que separa los componentes de las apps tanto lógica como físicamente y de aquí los microservicios (se encargan de una parte del proceso)  
.



Pequeño resumen de la clase:  
**1. Apps Monolíticas**: En los 90’s surgieron lenguajes como PHP y Perl, los cuales permitian desarrollar aplicaciones de una forma fácil y rápida. donde se mezclaba código HTML con código PHP para consultas a la base de datos, etc. La desventaja de esto era que se tenía mucha disponibilidad del archivo (si una linea de código fallaba se podía romper toda la aplicación). Con el paso de los años esto se fue mejorando, separando el HTML y el CSS de los archivos PHP mediante el uso de motores de templating, se dejaba la lógica en otros archivos PHP y las consultas a la Base de Datos mediante ORMs. Esto mejoró un poco la estructura de la aplicación, sin embargo teníamos el problema de tener toda la aplicación en un solo servidor. Cuando algo fallaba en el servidor, esto hacía que toda la aplicación dejara de funcionar.  
Con el paso de los años se empezó a buscar un concepto llamado “*bajo acoplamiento, alta cohesión*”, que básicamente consiste en separar componentes de las aplicaciones no solo a nivel lógico sino a nivel físico, será mucho mejor.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**2. Microservicios**: Los microservicios son un enfoque para el desarrollo de aplicaciones en el que una aplicación se construye como un conjunto de servicios modulares (es decir, módulos / componentes débilmente acoplados). Cada módulo apoya un objetivo comercial específico y utiliza una interfaz simple y bien definida para comunicarse con otros conjuntos de servicios. En lugar de compartir una sola base de datos como en la aplicación Monolitica, cada microservicio tiene su propia base de datos. Tener una base de datos por servicio es esencial si desea beneficiarse de los microservicios, ya que garantiza un acoplamiento flexible. Cada uno de los servicios tiene su propia base de datos. Además, un servicio puede utilizar el tipo de base de datos que mejor se adapte a sus necesidades.

Diagrama

Descripción generada automáticamente

**Stacks LAMP, MERN, JOTL, JAM** 4/17

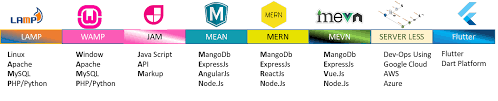
****

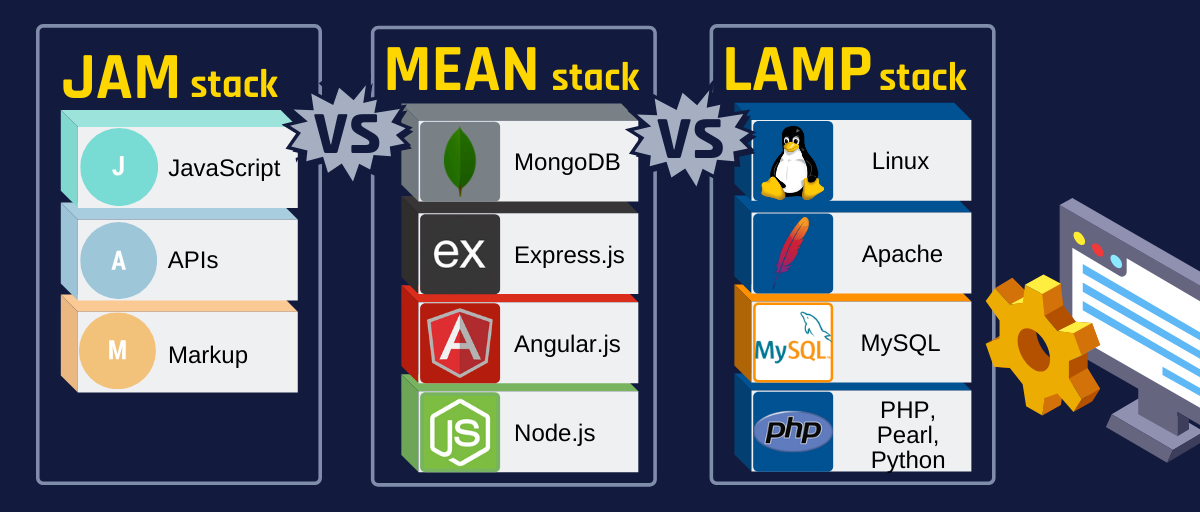
***Stacks de desarrollo resumen***

**Los stacks de tecnologías son el conjunto de herramientas para nuestra app en campos backend, frontend y bases de datos, estás pueden ser:**

* ***LAMP* : Compuesta por Linux, Apache, MySql y Php.**
* ***JOTL*: Compuesta por Java, Oracle, Tomcat y Linux.**
* ***MERN*: Mongo, React, Node, Express. (Cabe aclarar que los framework de javaScript pueden cambiar, entre angular Vue, React)**
* ***JAM*: JavaScript, Api y Markdown**

**Es importante tener en cuenta que estos stacks pueden variar según sea tu conveniencia.**

****



**LAMP**

**Logotipo, nombre de la empresa

Descripción generada automáticamente**

**El despliegue de una aplicación**[**LAMP**](https://es.wikipedia.org/wiki/LAMP)**(Linux, Apache, Mysql, PHP) puede ser de los más conocidos y populares dado la popularidad de php y mysql Cuando empezaron a aparecer las aplicaciones web .**

**Existen varias formas de desplegar, estas son las más comunes:**

* [**Hosting compartido**](https://es.wikipedia.org/wiki/Alojamiento_web)**: la fórmula más popular es comprar un servicio de hosting donde te proveen de una interfaz web llamada Cpanel donde puedes crear tu base de datos mysql, subir tus archivos php por ftp o administrador web y tener tu app en minutos.**
* **Hosting gratuito: Algunas empresas proveen hosting gratuito a cambio de que se integre publicidad en tus scripts php o de acceder a la información de tu sitio, sin embargo estas tienden a tener interfaces web menos amigables para subir archivos de la aplicación y la base de datos.**
* **Usar un VPS: utilizando plataformas como Digital Ocean, se puede crear un droplet (forma en que llaman a los VPS en esta empresa), para tener acceso SSH y poder instalar php,mysql, apache y lo que se necesite para instalar la aplicación web, puede tomar más tiempo en configurar todo, y el vps se debe administrar por la persona, a cambio, se gana acceso total al servidor para modificar php, mysql, y realizar tareas de gestión, o escalamiento de la aplicación.**

**JOTL**

****

**Por otra parte, en el mercado también es muy popular el stack de la empresa Oracle JOTL (Java, Oracle, Tomcat, Linux) dado el soporte y la fama que tiene Oracle de tener el sistema de base de datos más robusto, y esto sumado con Java que es un lenguaje de programación multiplataforma: funciona para hacer aplicaciones de escritorio, aplicaciones web, aplicaciones móviles para Android, etc.**

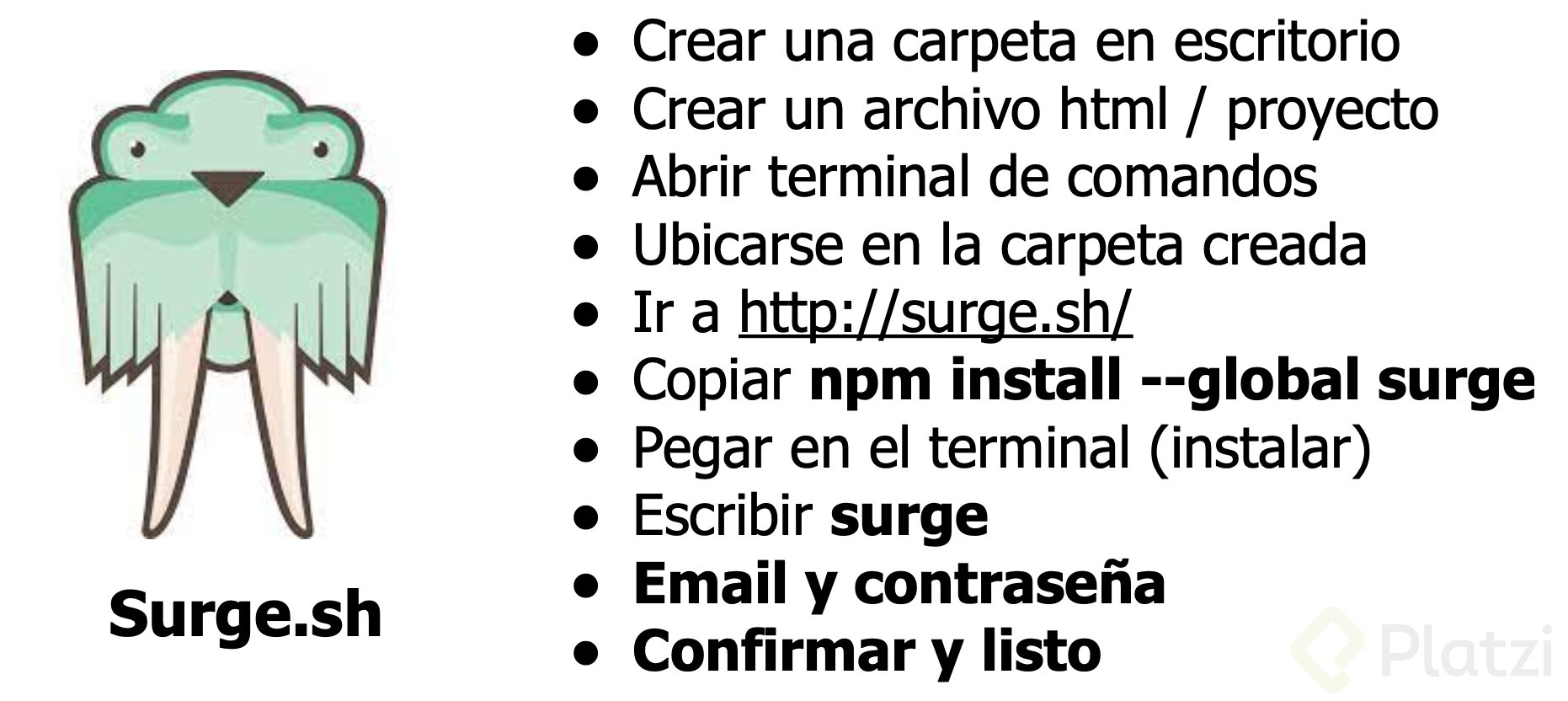
**Estas son las formas más comunes para desplegar una aplicación JOTL.**

* **Usar una plataforma como servicio: Se puede utilizar una PaaS - Platform As A Service, como es el caso de heroku, que se encarga del despliegue de la aplicación y se puede hacer un despliegue más rápido, pero se pierde el control sobre el servidor.**
* **Usar una Infraestructura como servicio: IaaS o Infrastructure As a Service, son empresas como AWS de Amazon, Cloud Platform de Google, Azure de microsoft o incluso IBM cloud, estas ofrecen un control mayor sobre la infraestructura, desde los servidores VPS, red, Backups, disponibilidad, escalabilidad, seguridad entre otras ventajas, sin embargo requieren de un conocimiento en manejo de infraestructuras para poder configurar todas estas opciones.**
* **Usar infraestructura propia: Algunas empresas prefieren disponer de una infraestructura propia, esto se conoce como on-premises, entre la razones y ventajas para este tipo de infraestructura están:  
  Privacidad del código fuente o aplicación, ya que este se encontrará local y no en servidores en una nube a los que terceros podrían acceder.**

**GitHub Pages**

**En mi**[**cuenta principal**](https://github.com/santiaguf)**de github tengo el repositorio apuntando a un**[**subdominio diferente**](https://sb.js.org/)**(**[**https://sb.js.org**](https://sb.js.org/)**) , te reto a que luego de realizar esta clase, configures tu repo con**[**js.org**](https://js.org/)

**Despliegue en Surge 7/17**



****

****

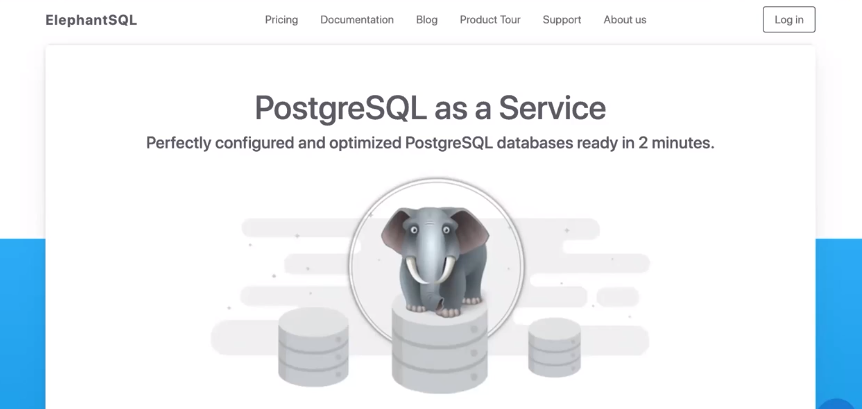
**// permite usar varios dominios**

****

**Algunos clientes MongoDB para Linux y Windows:**

* [**Mongo Shell**](https://docs.mongodb.com/manual/mongo/)
* [**Robo 3T**](https://robomongo.org/download)
* [**Mongo Compass**](https://www.mongodb.com/try/download/compass)
* Si su contraseña tienen caracteres especiales (#@!@#$%^&\*…), deben escribirla como URL encoded, por si lo necesitan aqui puede realizar el proceso <https://www.urlencoder.org/>
* Si no le pasan la contraseña como URL encoded no los dejara conectar





 //interfaz gráfica

Aquí en este sitio tambien tenemos la opcion de crear bases de datos relacionales como **mysql** y **postgresql**  
[https://www.clever-cloud.com](https://www.clever-cloud.com/)

El cliente grafico que me gusta utilizar es DBeaver es una muy buena alternativa y es open source (community edition) <https://dbeaver.io/>

[alwaydata](https://alwaysdata.com/) es otra alternativa para lo que es base de datos

**Qué es Heroku** 12/17

****

Heroku es un PaaS Platform As A Service y te permite desplegar tus aplicaciones backend con los lenguajes soportados por Heroku, ellos se encargan de la administración del servidor

El curso ya tiene un rato de haber salido, pero para aquellos que apenas están empezando, les dejo el comentario de que hace unos días en agosto de 2022, Heroku dejará de ser gratuito y todos sus servicios estarán en un plan de pago. Aquí colocó algunas alternativas a considerar.

* **Back4app:** es una de las mejores plataformas de código abierto que puede utilizar tanto para PaaS como para BaaS; Nivel gratuito disponible, API GraphQL, API REST, Código de la nube.
* **Elastic Beanstalk:** es también conocido como AWS y una gran mayoría de la población en general ya lo está utilizando; Soporta múltiples lenguajes de programación, Registro, monitoreo y estado de la aplicación, Solo paga por recursos de AWS como EC2 y S3.
* **Engine Yard:** es una PaaS (Plataforma como servicio) que es líder en Ruby DevOps. Las características incluyen escalado horizontal y vertical, actualizaciones de seguridad constantes y copiado del entorno con fines de prueba.
* **Firebase:** ofrece muchas ventajas y características increíbles; base de datos en tiempo real, almacenamiento, nivel gratuito.
* **Dokku:** es una plataforma bastante simple y fácil de usar, y ofrece muchas técnicas simples para desarrollar aplicaciones; Impulsado por Docker, Implementaciones de Git sencillas, Multi Cloud.

**Desplegando Api en Heroku** 13/17

También existen otras cosas llamadas BaaS (Backend As A Service) donde tú no te preocupas por programar backend, sino que ya está programado y tu solo lo configuras

Lo que podemos usar en vez de heroku, es Glitch o Vercel.

Les dejo el [link al proyecto](https://glitch.com/edit/#!/nodetodoapp) en Glitch, y el [URL público](https://nodetodoapp.glitch.me/)

Firebase es la ultima generacion de plataforamas, se considera BPaaS de Business As a Services, no confundir con BaaS que es Backend as a Service,

La diferencias contra BaaS, SaaS, IaaS, PaaS es que en un BPaaS no te importa como esta hecho por detras, ni que tecnologia, simplemente tu solucitar una funcionalidad y el BPaaS te la da.

Como bien dice @JuanGalvis, Firebase tiene muchos elementos para utilizar como lo es el Hosting, Functions, Firestore, pero en ningun momento se dice una tecnologia en particular, osea que por detras puede tener base de datos MySQL, Maria, Mongo, etc… lo mismos para el servidor web Apache,Tomcat, Express, etc… Pero en realidad eso no importa porque simplemente se utiliza, delegando todo a Google

**Consultando nuestra API desde Postman** 14/17

**Generar documentación de API con Postman** 15/17

Documentación API con todos los métodos para realizar las operaciones CRUD  
https://documenter.getpostman.com/view/3057287/SzKVRJFd

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



