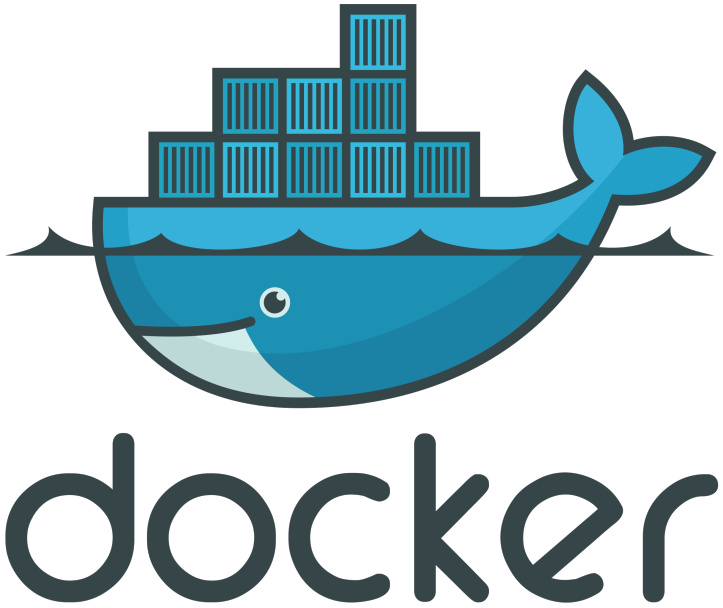
Docker 101 – Módulo 4

GBM Tech Academy

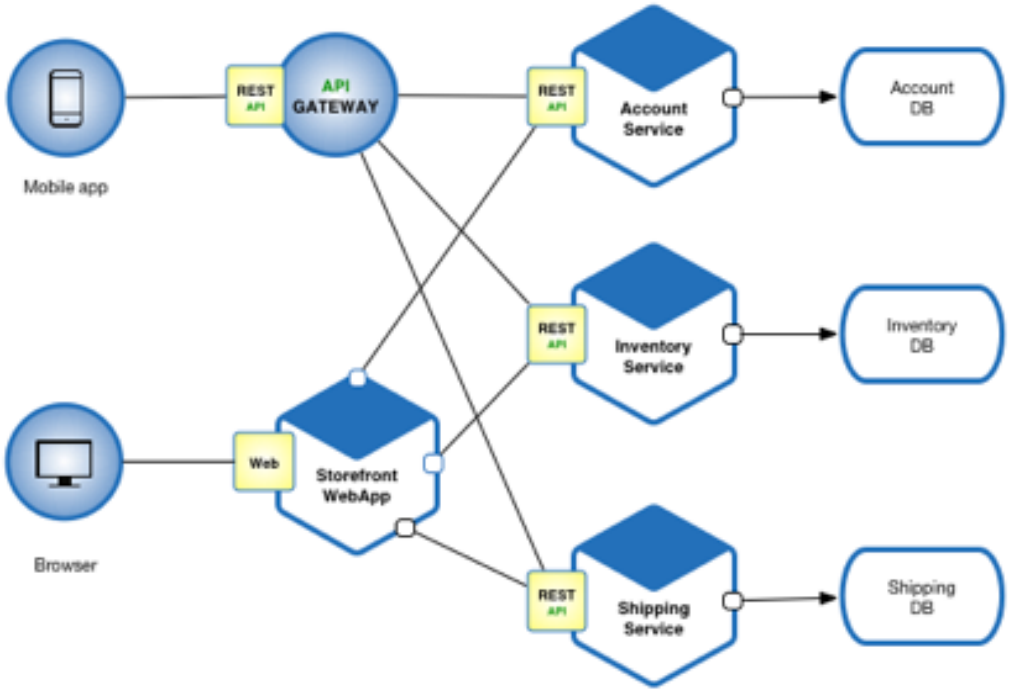
Agenda

- Stacks de Programación
- Aplicación de Ejemplo
- Empaquetado de Aplicaciones



Docker Container Lifecycle

- Patrón de diseño de aplicaciones (listas para integrarse)
- Su propósito es estructurar aplicaciones como un conjunto de funcionalidades de negocio desacopladas
- Fomenta la exposición rápida y segura de servicios a los clientes.
- Pieza fundamental de la estrategia de Transformación Digital que busca mejorar la experiencia de los clientes
- Enfocada en incrementar los ingresos a partir de la entrega de nuevos servicios



Principios y Beneficios de los Microservicios

Principios

One Job
Cada microservicio debe estar optimizado para una sola función

Separate Processes
Comunicación entre microservicios debe estar basada en REST API y Message Brokers

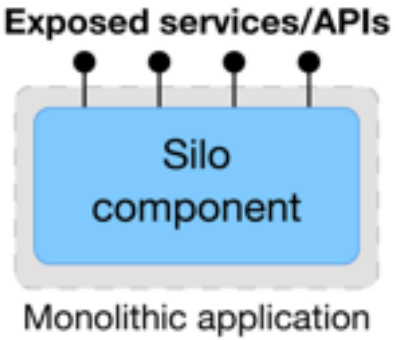
Execution Scope
El foco no está en la interfaz sino en el componente

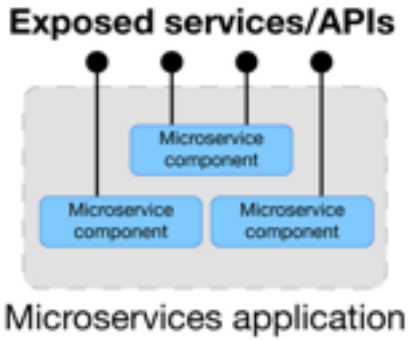
Separate CI/CD
Cada microservicio debe poder evolucionar a su propio ritmo

Resiliency
Cada microservicio tiene sus necesidades de alta disponibilidad y recuperación ante fallos

Beneficios

- Efficient Teams
- Simplified Deployment
- Right Tools for the Job
- Improved Application Quality
- Scalability





Patrones y frameworks para microservicios

Stacks Integrados

- LAMP: Linux/Apache/MySQL/PHP (WAMP, LAPP, MAMP, XAMPP)
- MEAN: MongoDB/Express.js/AngularJS/Node.js (MERN, MEEN)
- Bitnami-Hosted Stacks
- Ruby Stack: Ruby/Ruby on Rails/RVM (Ruby Virtual Machine)/MySQL/Apache/PHP
- Django Stack: Python/Django/Apache/MySQL

Node.js

- StrongLoop
- Express
- Koa

Java

- SpringBoot
- Java Microprofile

Bases de Datos

- Relational: DB2, PostgreSQL, MySQL, SQL Server, Oracle
- NoSQL: Cloudant, MongoDB
- In-Memory/Cache: Redis

Messaging/Events

- IBM MQ
- MessageHub
- RabbitMQ
- Kafka

Twelve Factor Apps

I. **Codebase**

Única base de código con control de versiones, múltiples despliegues.

II. **Dependencies**

Explicitamente declaradas y aisladas

III. **Config**

La configuración provista por el entorno

IV. **Backing services**

Servicios de soporte (almacenamiento, mensajes, smtp) como recursos adjuntos

V. **Build, release, run**

Etapas de integración y ejecución estrictamente separadas

VI. **Processes**

Uno o varios procesos estables y aislados entre ellos

VII. **Port binding**

Servicios se exponen a través de puertos enlazados

VIII. **Concurrency**

Escalabilidad basada en modelo de procesos

IX. **Disposability**

Robustez basada en desechar/crear en lugar de reiniciar

X. **Dev/prod parity**

Mantener ambientes de desarrollo, pruebas y producción “iguales”

XI. **Logs**

Tratar los logs como un flujo de eventos (event stream)

XII. **Admin processes**

Tareas administrativas como procesos de una ejecución

Demo Aplicación

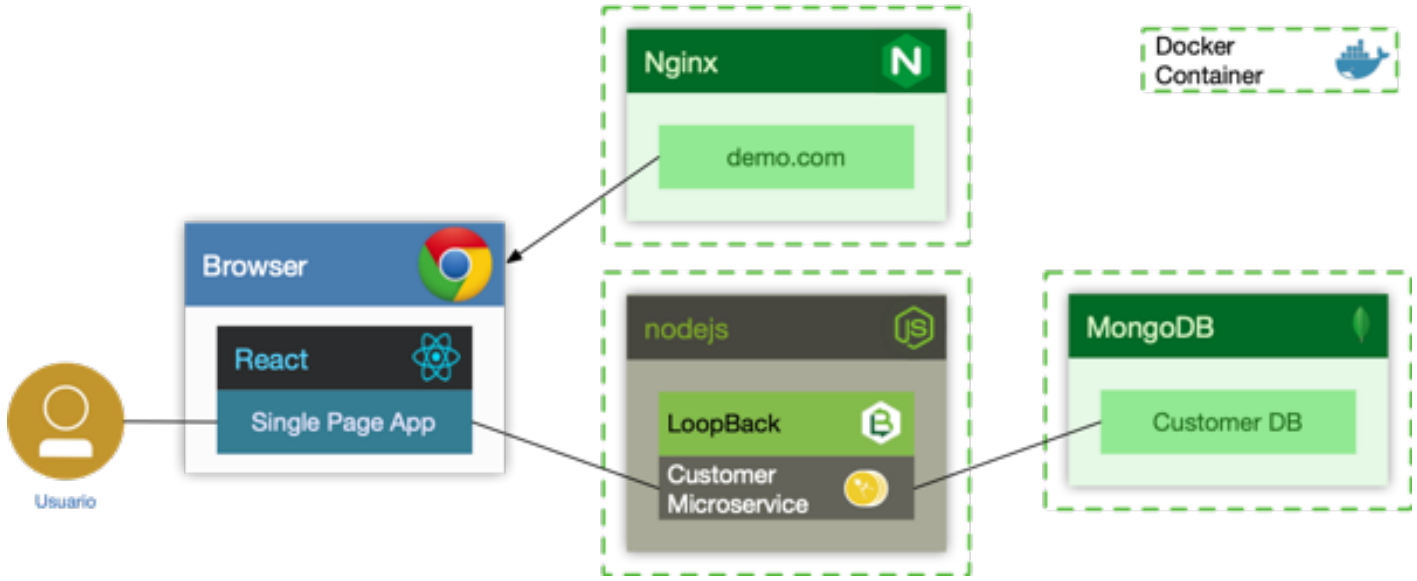
Demo Arquitectura Microservicios

MERN Framework

- MongoDB (noSQL)
- ExpressJS (Loopback)
- React (UI)
- NodeJS

Herramientas de Desarrollo

- Git/GitHub
- VSCode
- PostMan
- Studio 3T



Demo

- Crear la estructura del Proyecto y agregarlos a Git
- Crear servicio de Datos
- Crear microservicio Customer
- Crear interfaz de usuario
- Colocar la aplicación en contenedores