Ingeniería de Software I
Patrones de Arquitectura

Jocelyn Simmonds

Departamento de Ciencias de la Computación

Características de un buen Diseño

- Un buen diseño debería conducir a un producto de calidad.
- Un buen diseño debe tener:
  1. Independencia de Componentes.
  2. Identificación y manejo de excepciones.
  3. Prevención y Tolerancia a fallas.

Requisitos no funcionales afectan el diseño

Algunos ejemplos:

- Performance: traten de concentrar las operaciones criticas, y minimizar la comunicación entre componentes
- Seguridad: definan una arq. por capas, donde solo capas internas tienen acceso a artefactos críticos
- Disponibilidad: incluyan componentes redundantes, mecanismos de tolerancia a fallos
- Mantenibilidad: diseñen los componentes para que sea reemplazables, con claras interfaces de comunicación y bajo acoplamiento

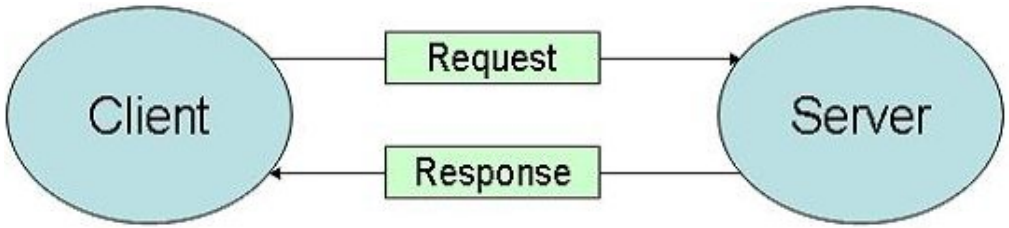
Decisiones a tomar durante el diseño

- ¿Cómo vamos a descomponer el sistema en modulos?
- ¿Cómo vamos a distribuir los componentes del sistema?
- ¿Cómo vamos a documentar nuestros diseños?
- ¿Cómo vamos a evaluar nuestros diseños?
- ¿Podemos usar algún patrón de arquitectura?

Cliente-Servidor

Separa el SW en 2 partes

- Cliente: código que se ejecuta en la maquina local
  - GUI + algo de lógica de negocios
- Servidor: código que corre en (potencialmente) otra maquina
  - mayoría de lógica de negocios, persistencia



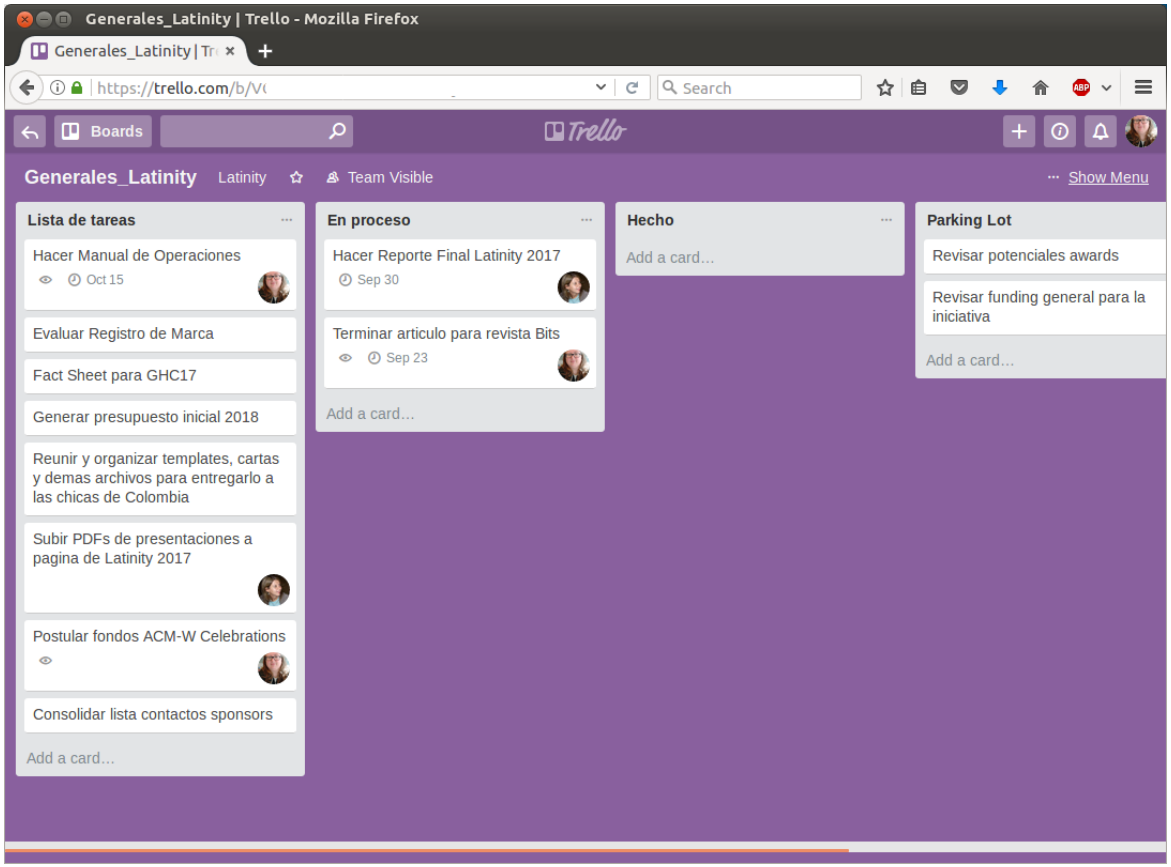
Cliente-Servidor

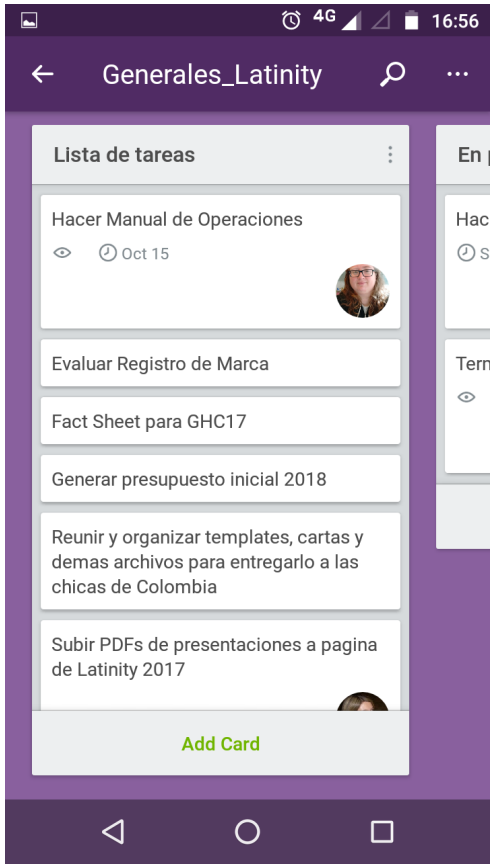
- Uno o mas clientes se comunican con el servidor usando algún protocolo (ej. http, ftp, ssh, etc.)
- Clientes pueden ser de diferentes tipos
  - PC (browser tradicional) vs. smart phone (mobile browser)
- Servidores replicados para entregar mejor servicio
- Uso de message queues para mejorar robustez de comunicación cliente-servidor

Cliente-Servidor: Trello

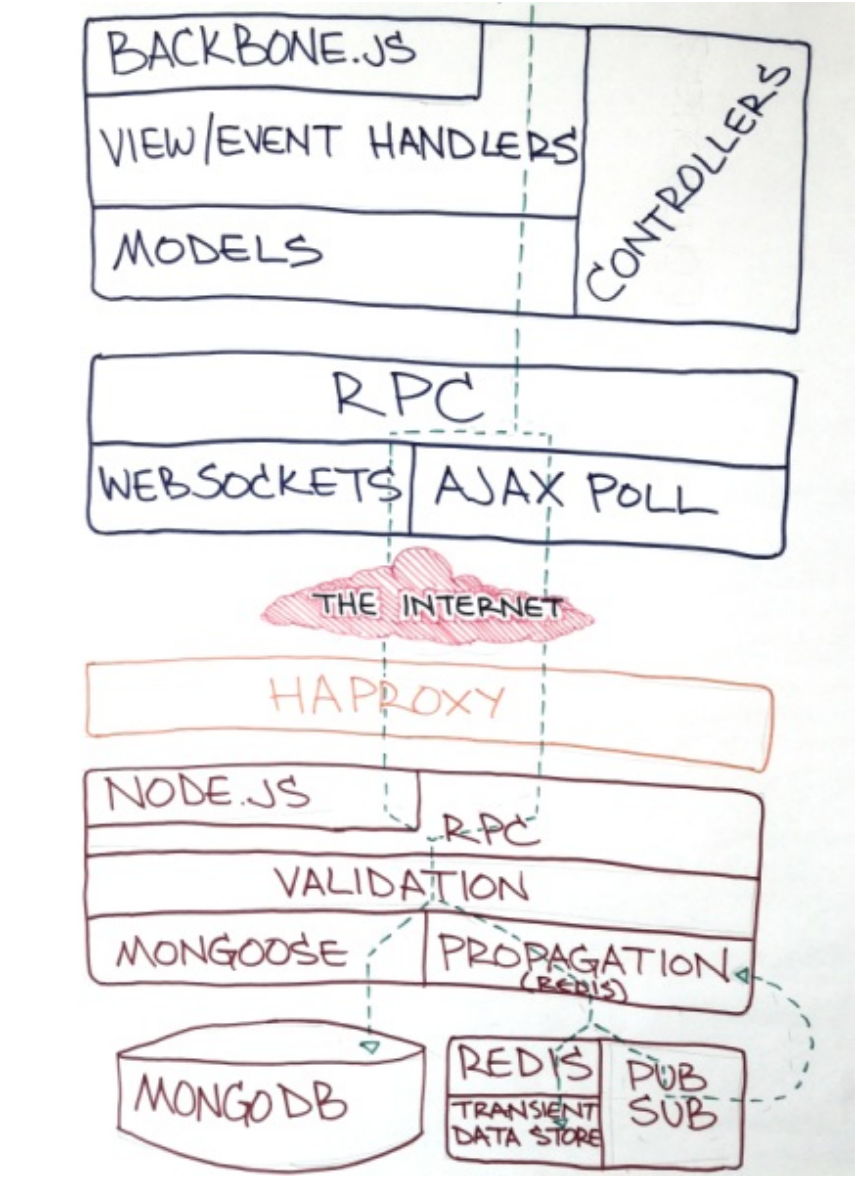
Plataforma para trabajo colaborativo:

- servidor: back-end donde se guardan los boards de los usuarios
- cliente: front-end para visualizar y editar los boards





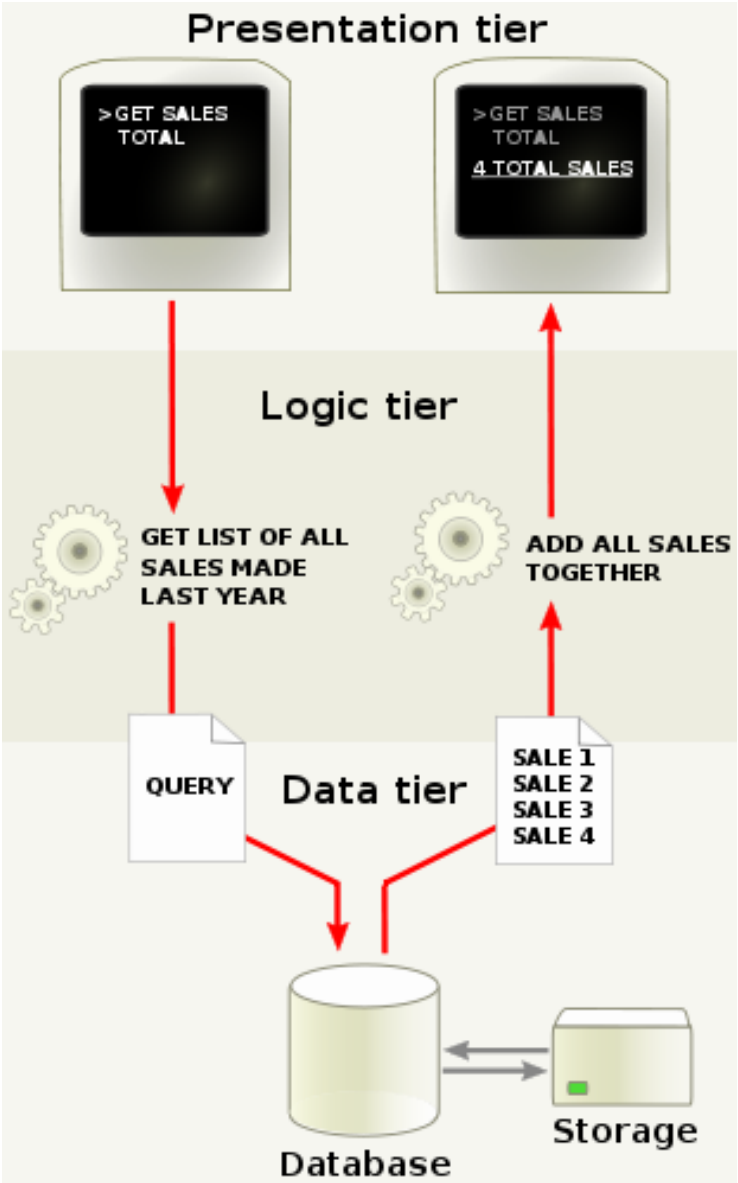
Cliente-Servidor: Trello



"No matter where you are, Trello stays in sync across all of your devices."

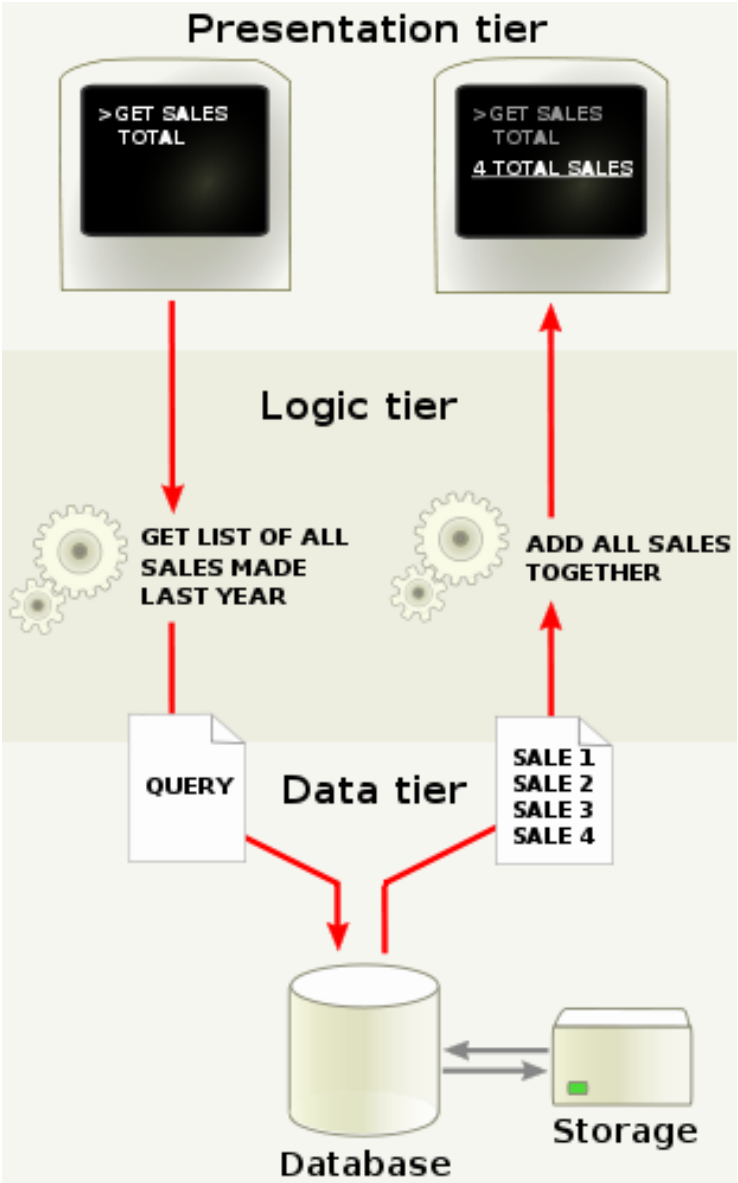
Arquitectura se discute en detalle en la siguiente pagina: `https://blog.trello.com/the-trello-tech-stack`

Tres capas



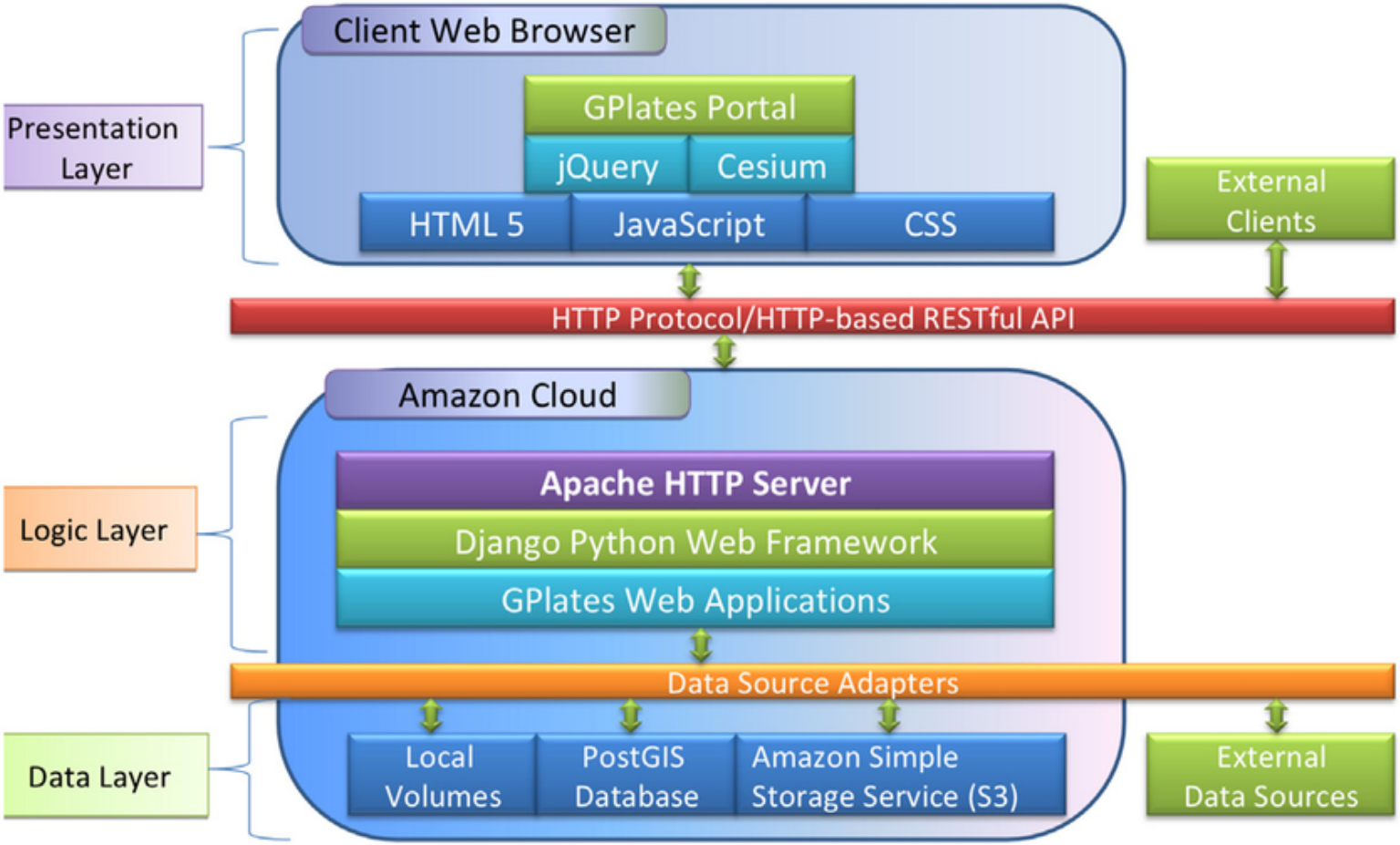
- Separa el SW en 3 partes
  - Capa de presentación: contiene la interfaz de usuario
  - Capa de lógica de negocio: contiene los programas que implementan las reglas del negocio de la aplicación
  - Capa de datos: interactúa con la BD

Tres capas



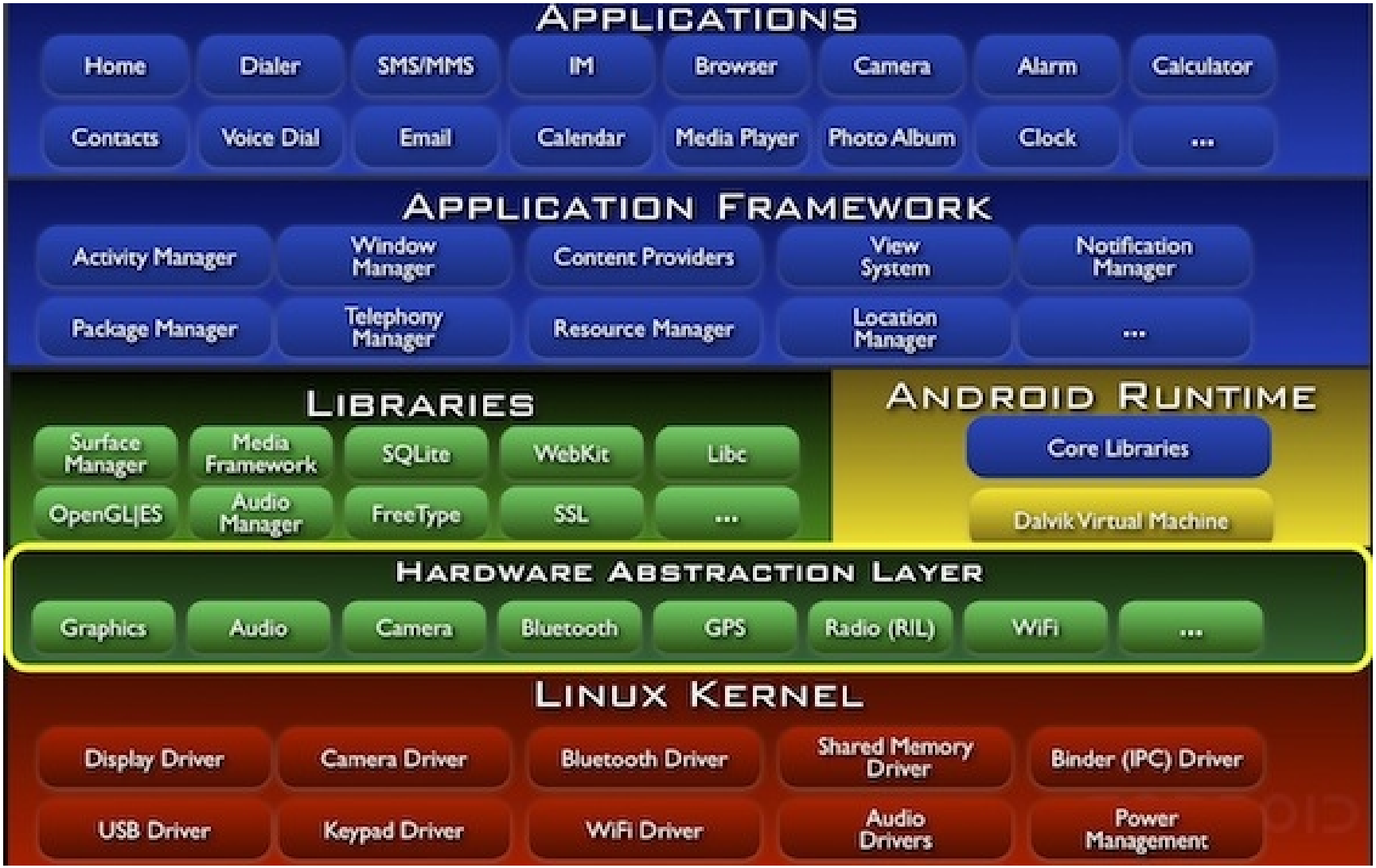
- Separa el SW en 3 partes
  - Capa de presentación: contiene la interfaz de usuario
  - Capa de lógica de negocio: contiene los programas que implementan las reglas del negocio de la aplicación
  - Capa de datos: interactúa con la BD
- Es ampliamente usada
- Capas pueden alojarse en el mismo o distintos procesadores, o incluso cada capa estar distribuida en varios procesadores
- Hereda y mejora beneficios de Cliente-Servidor

Ejemplo de tres capas

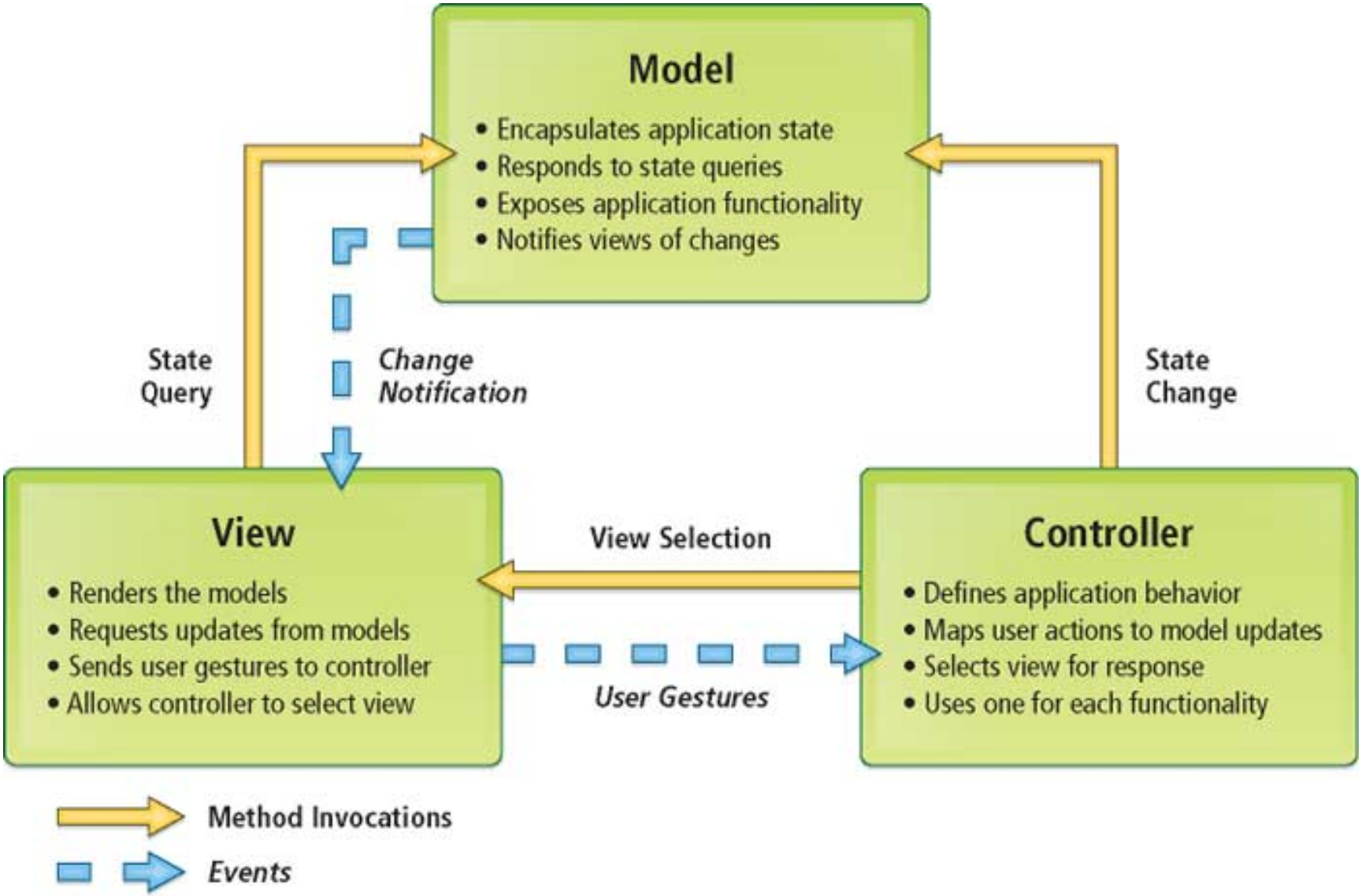


The GPlates Portal: Cloud-Based Interactive 3D Visualization of Global Geophysical and Geological Data in a Web Browser

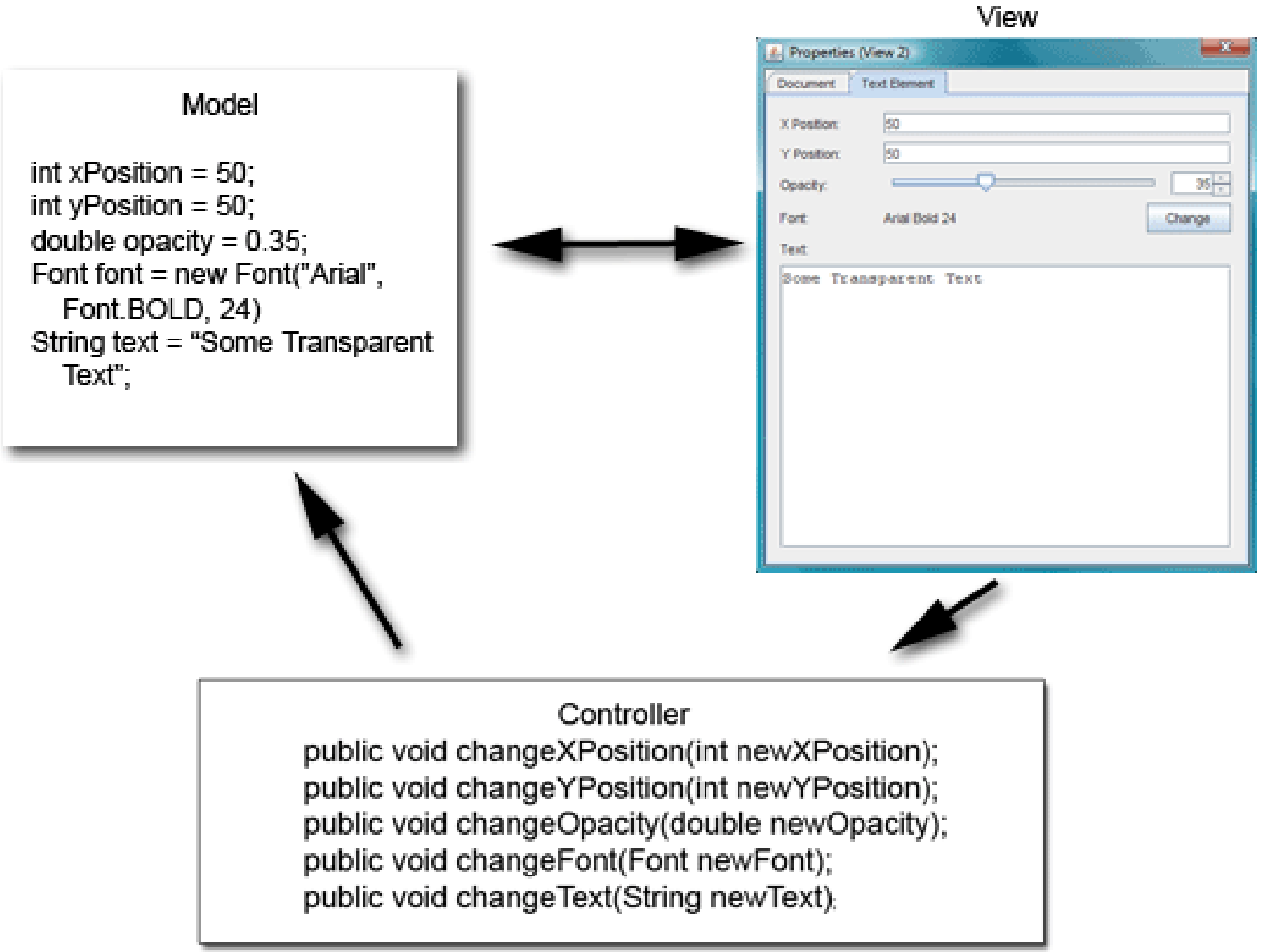
Más capas


Applications
Home
Dialer
SMS/MMS
IM
Browser
Camera
Alarm
Calculator
Contacts
Voice Dial
Email
Calendar
Media Player
Photo Album
Clock
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
...
Libraries
Surface Manager
Media Framework
SQLite
WebKit
Libc
OpenGL|ES
Audio Manager
FreeType
SSL
...
Android Runtime
Core Libraries
Dalvik Virtual Machine
Hardware Abstraction Layer
Graphics
Audio
Camera
Bluetooth
GPS
Radio (RIL)
WiFi
...
Linux Kernel
Display Driver
Camera Driver
Bluetooth Driver
Shared Memory Driver
Binder (IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Model View Controller (MVC)

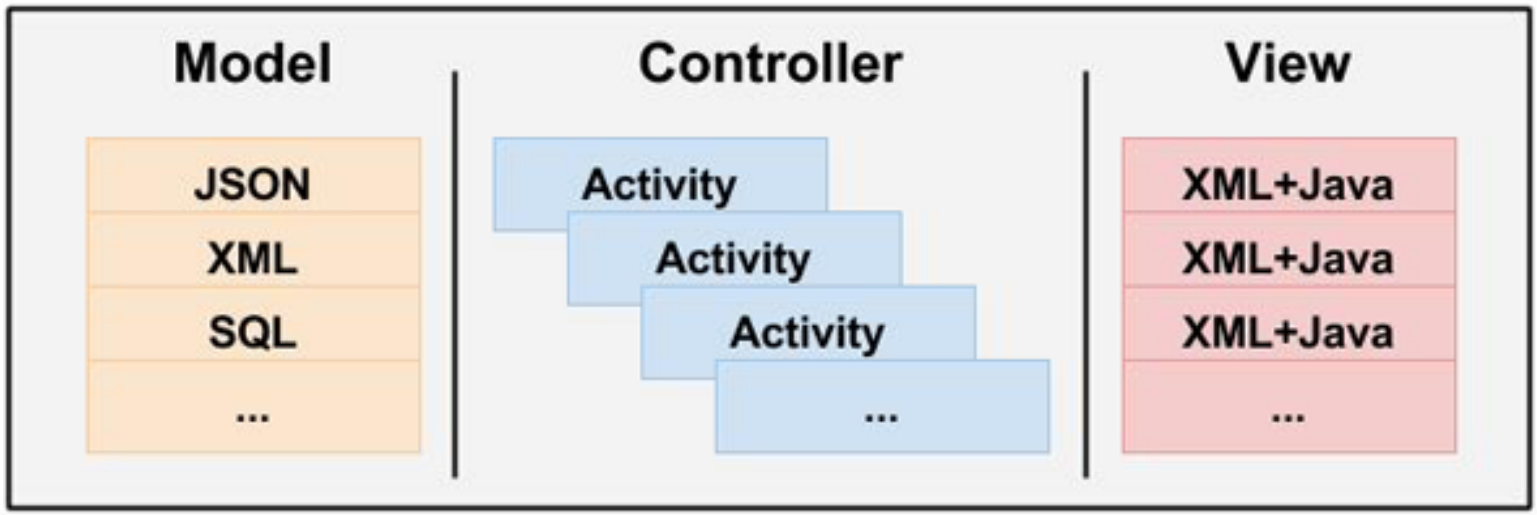


Interfaces gráficas en Java

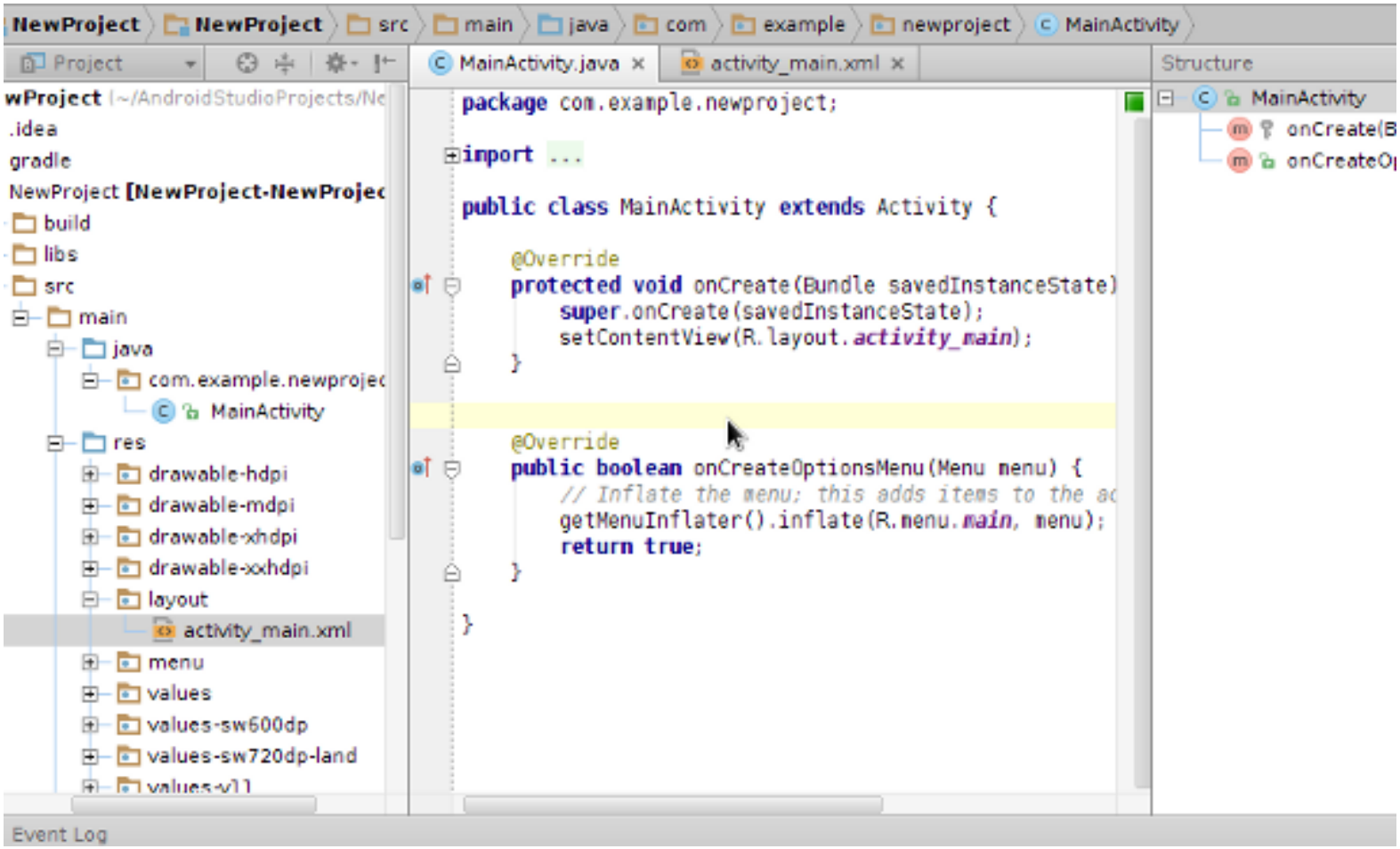


Aplicaciones en Android

Una visión un poco simplista de aplicaciones en Android:



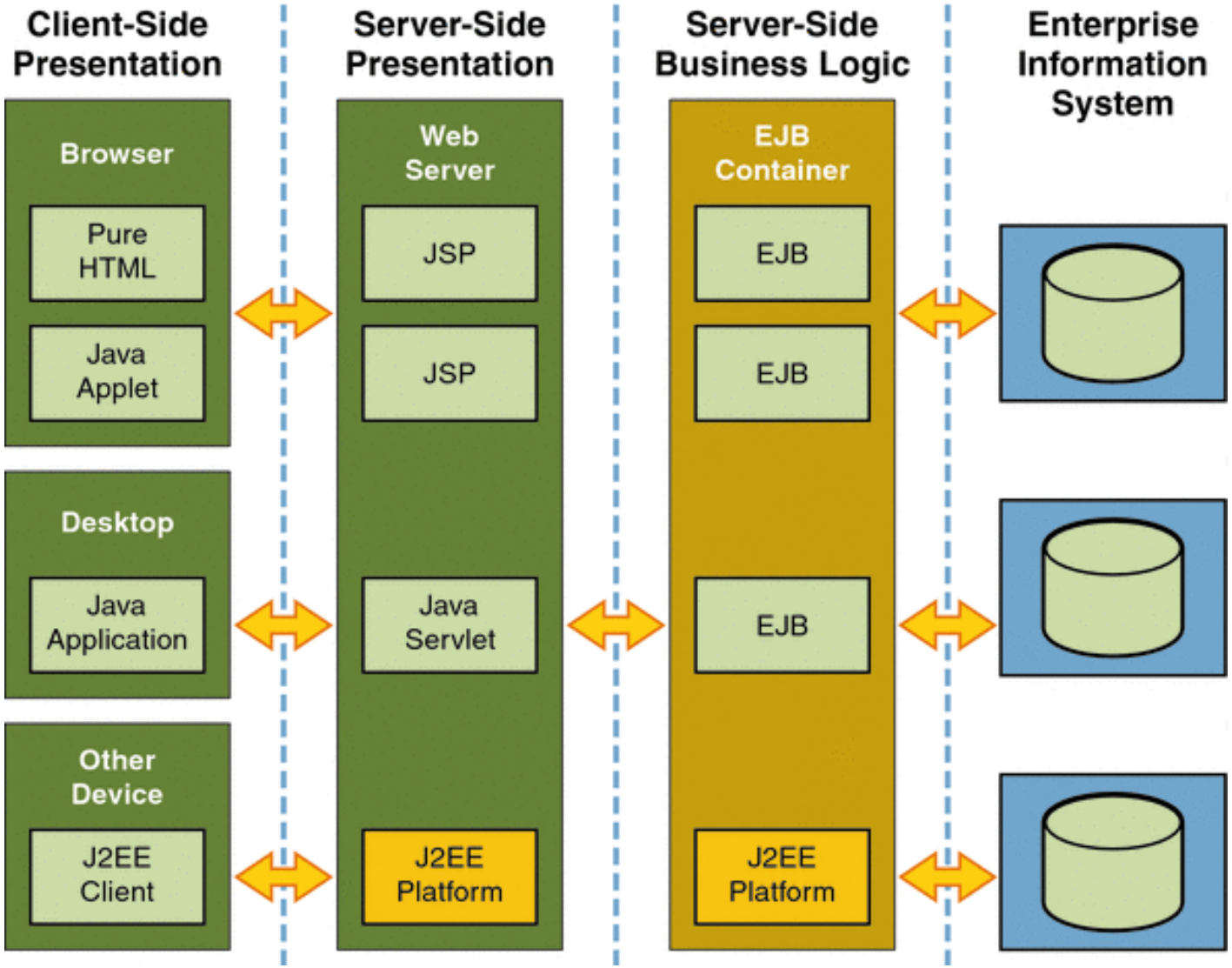
Aplicaciones en Android



Arquitecturas de Objetos Distribuidos

- No hay distinción entre clientes y servidores
- Cada entidad distribuida es un objeto que provee servicios a otros objetos y recibe servicios de otros objetos
- La comunicación entre objetos es mediante un sistema middleware que se llama un corredor de solicitudes a objetos
- Arquitecturas de referencia de objetos distribuidos
  - CORBA, J2EE, COM (.NET)

J2EE: Java 2 Enterprise Edition

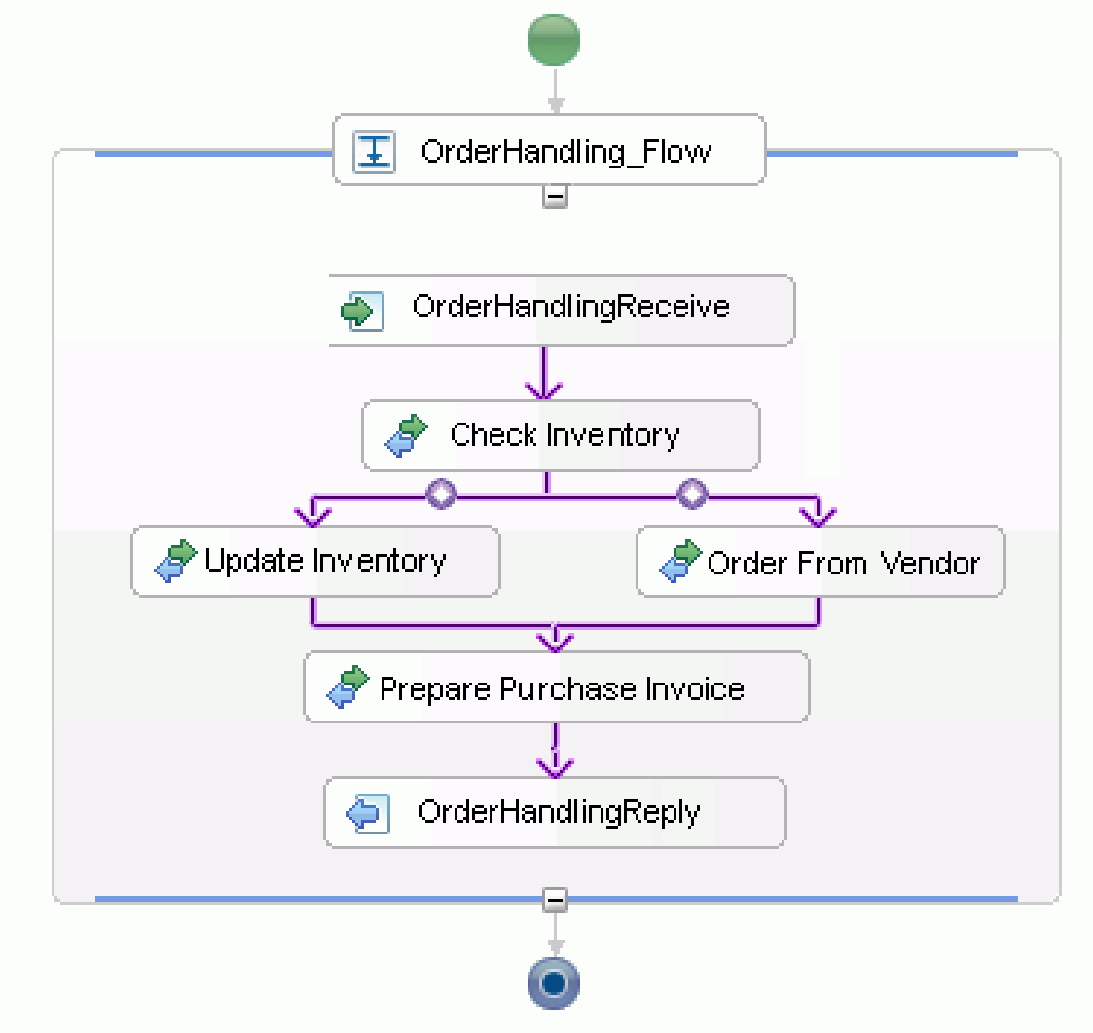


Arquitecturas Orientadas a Servicios

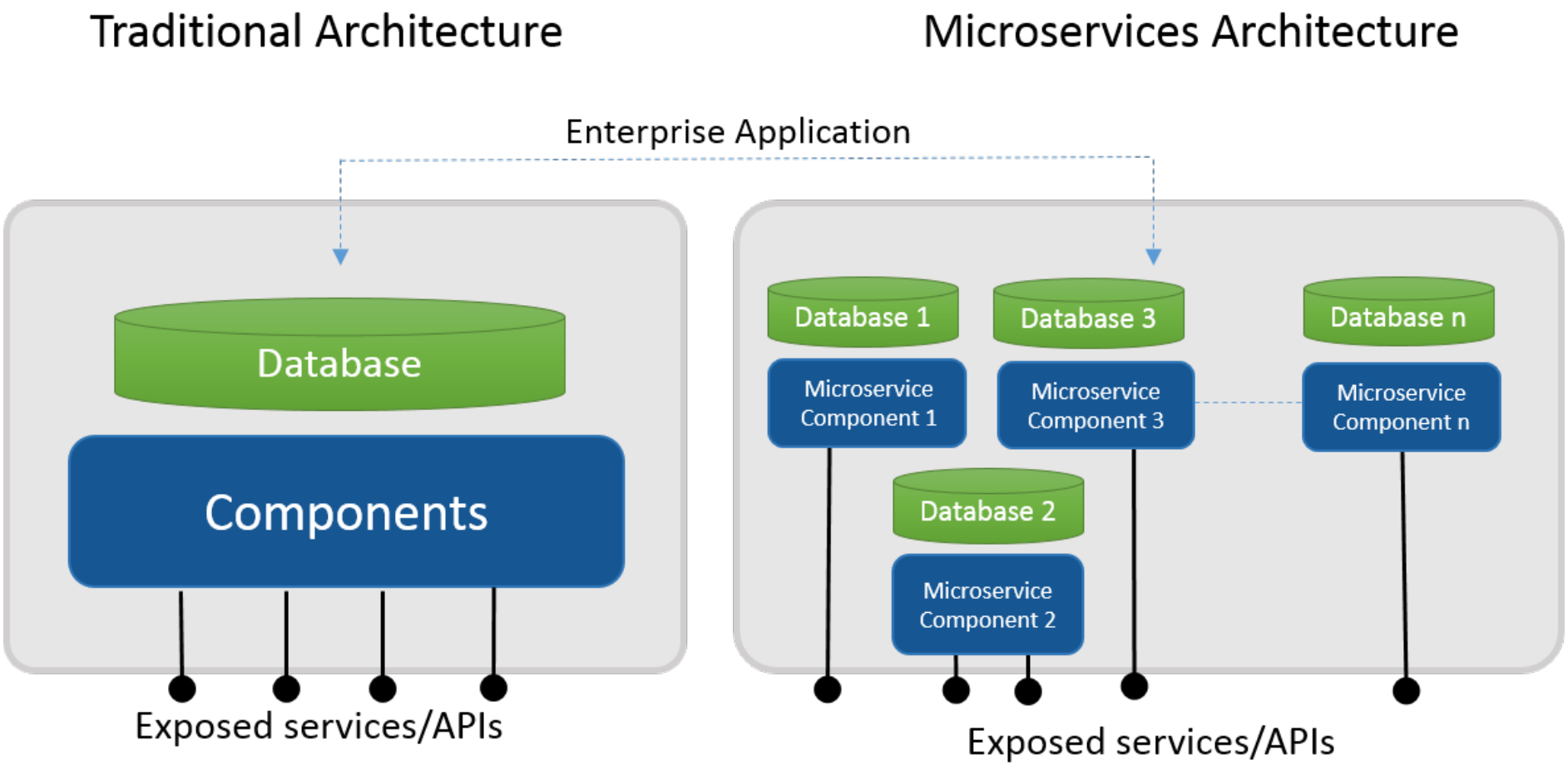
- Están basadas en la noción de servicios provistos en forma externa (servicios web)
- Un servicio web es una forma estándar de hacer que una componente reutilizable esté disponible en la Internet
  - Un servicio de declaración de impuestos podría ayudar a los usuarios a llenar sus formularios de declaración y enviarlos a impuestos internos
  - Google provee varios servicios web para incorporar la data de mapas que tienen: `https://developers.google.com/maps/web-services/overview`

Web Services y SOA

Aplicaciones BPEL: orquestación de servicios Web

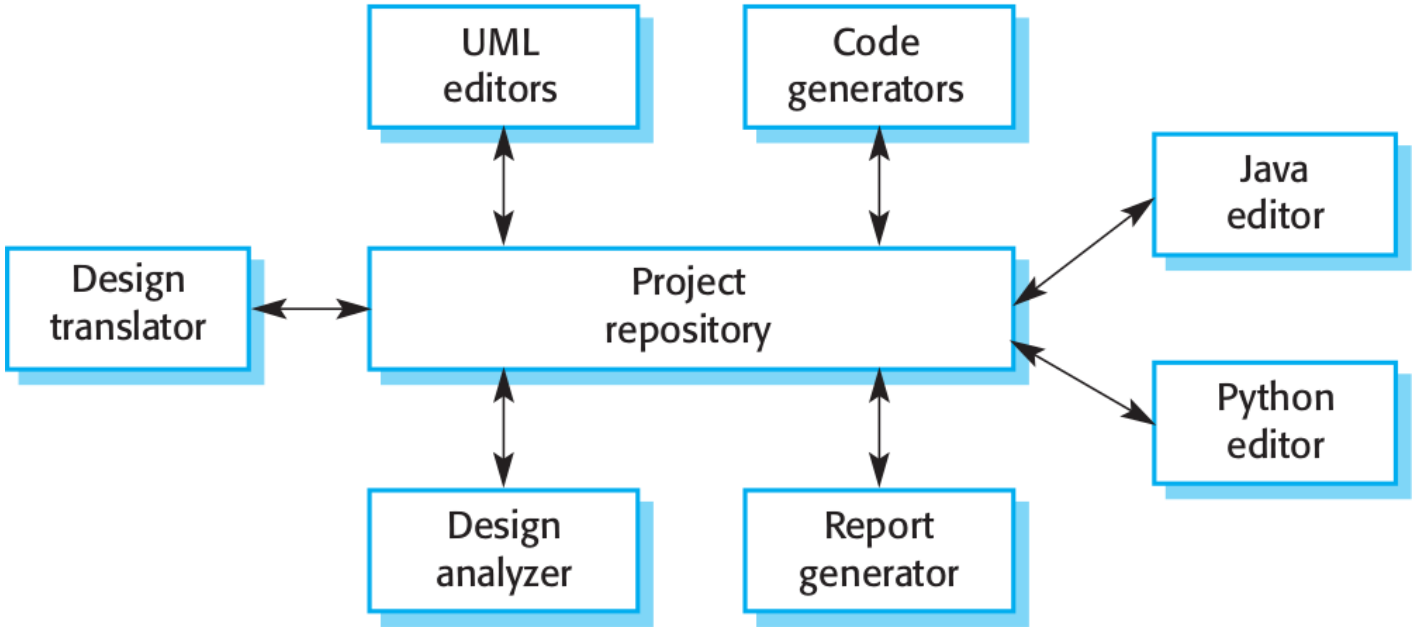


Migración hacia microservicios



Basada en repositorios

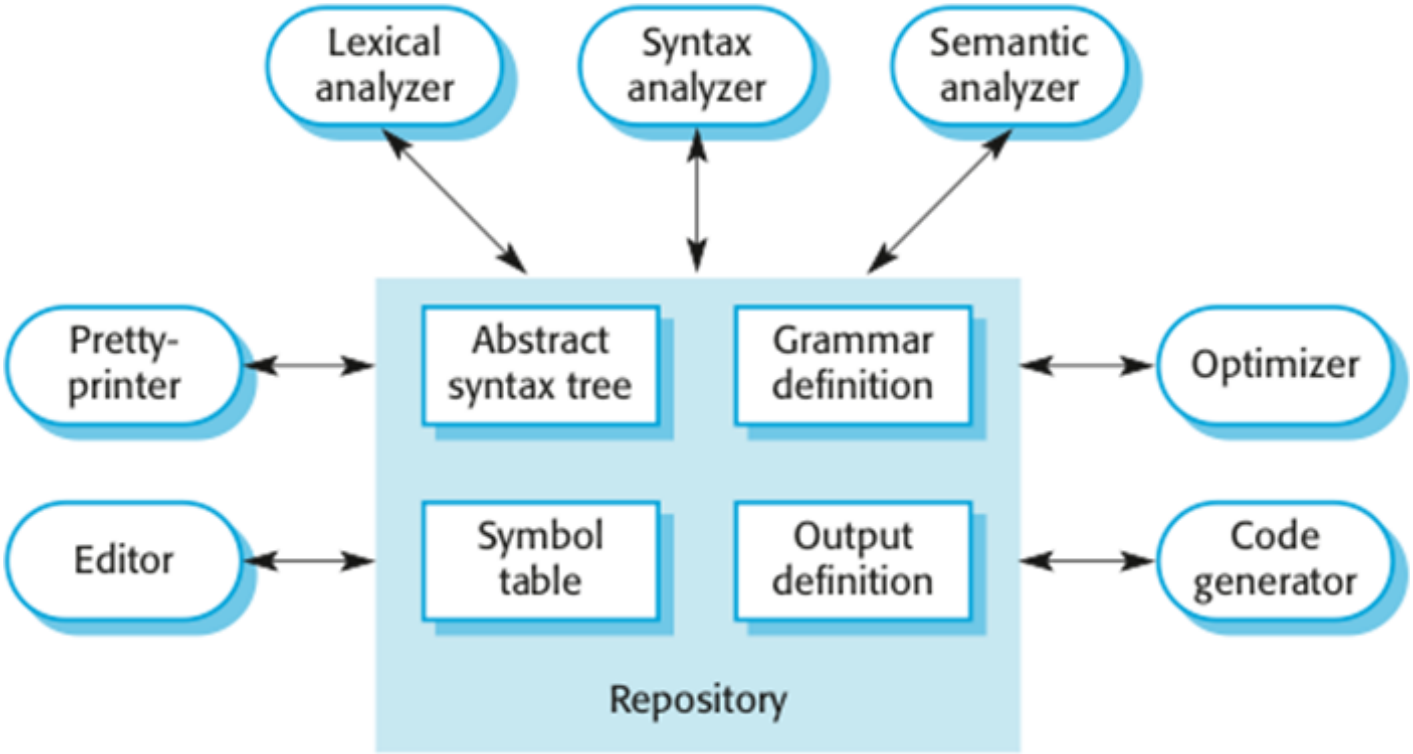
Esta arquitectura se usa cuando los componentes deben compartir grandes cantidades de datos:



Ojo que es frágil con respecto a cambios en los datos compartidos.

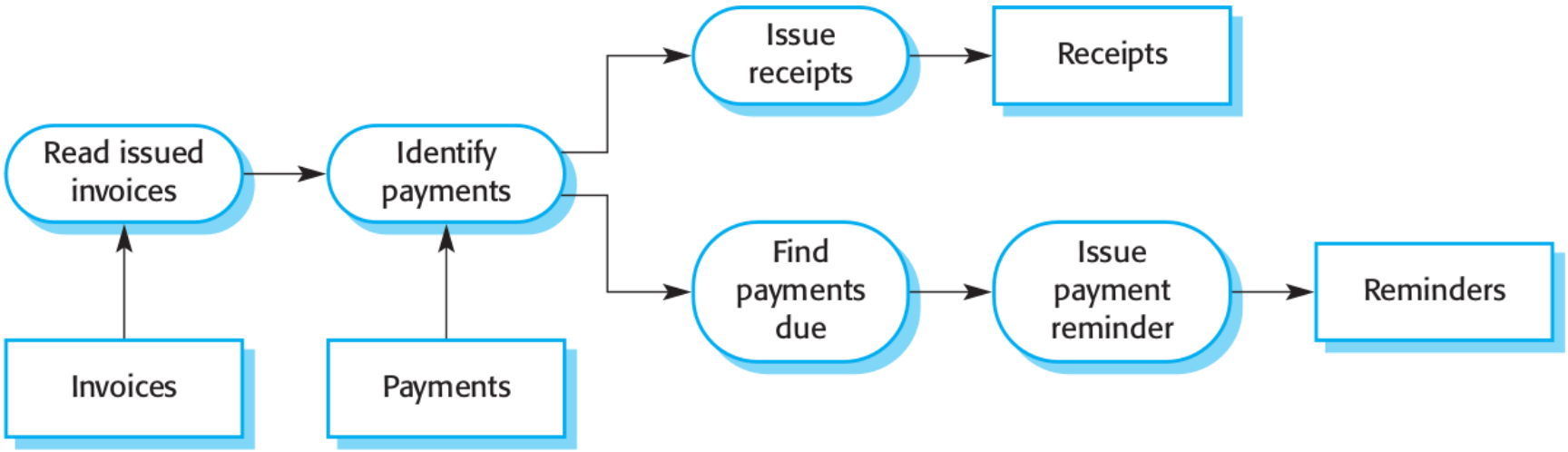
Ejemplo

Arquitectura de repositorio para un conjunto de herramientas para procesar lenguajes



Pipe & filter

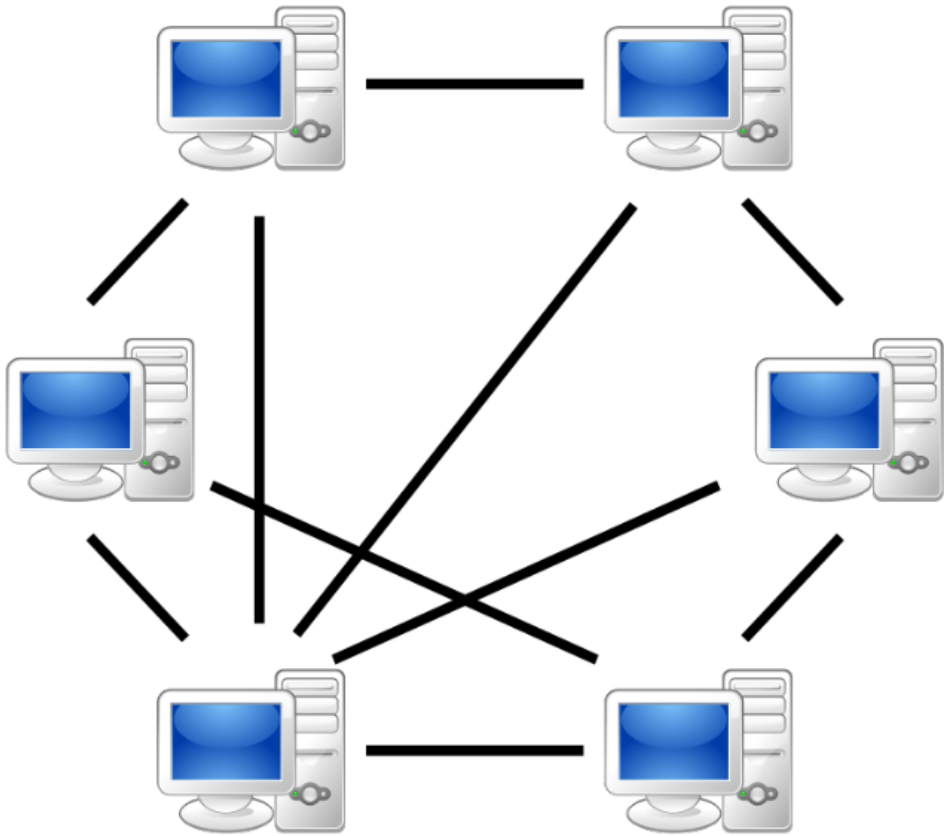
Cada filtro transforma los datos antes de pasarselos al siguiente filtro:



También es frágil con respecto a cambios en los datos.

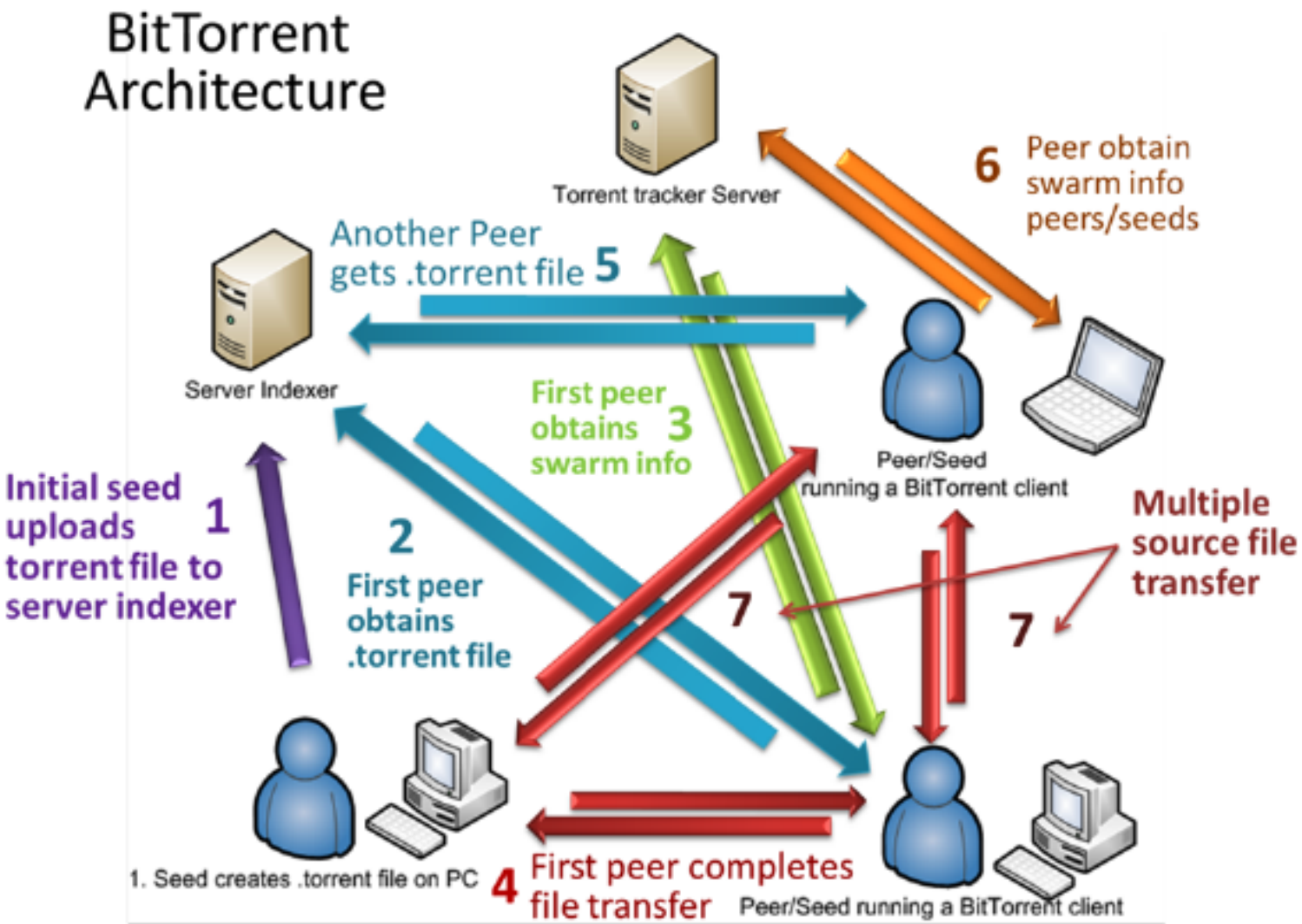
Peer-to-peer

Peers se coordinan entre si, sin un control centralizado:



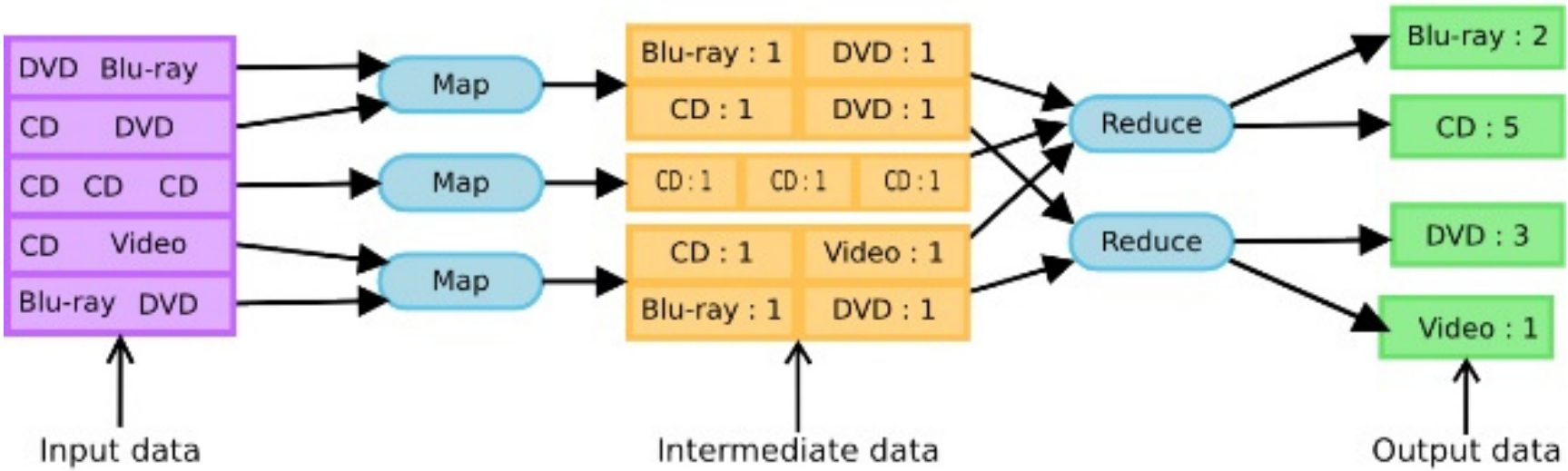
Cada peer provee y consume recursos (CPU, almacenamiento, etc.)

Peer-to-peer


BitTorrent Architecture
Torrent tracker Server
6 Peer obtain swarm info peers/seeds
Another Peer gets .torrent file 5
Server Indexer
First peer obtains 3 swarm info
Peer/Seed running a BitTorrent client
Initial seed uploads 1 torrent file to server indexer
2 First peer obtains .torrent file
7
7
Multiple source file transfer
1. Seed creates .torrent file on PC
4 First peer completes file transfer
Peer/Seed running a BitTorrent client

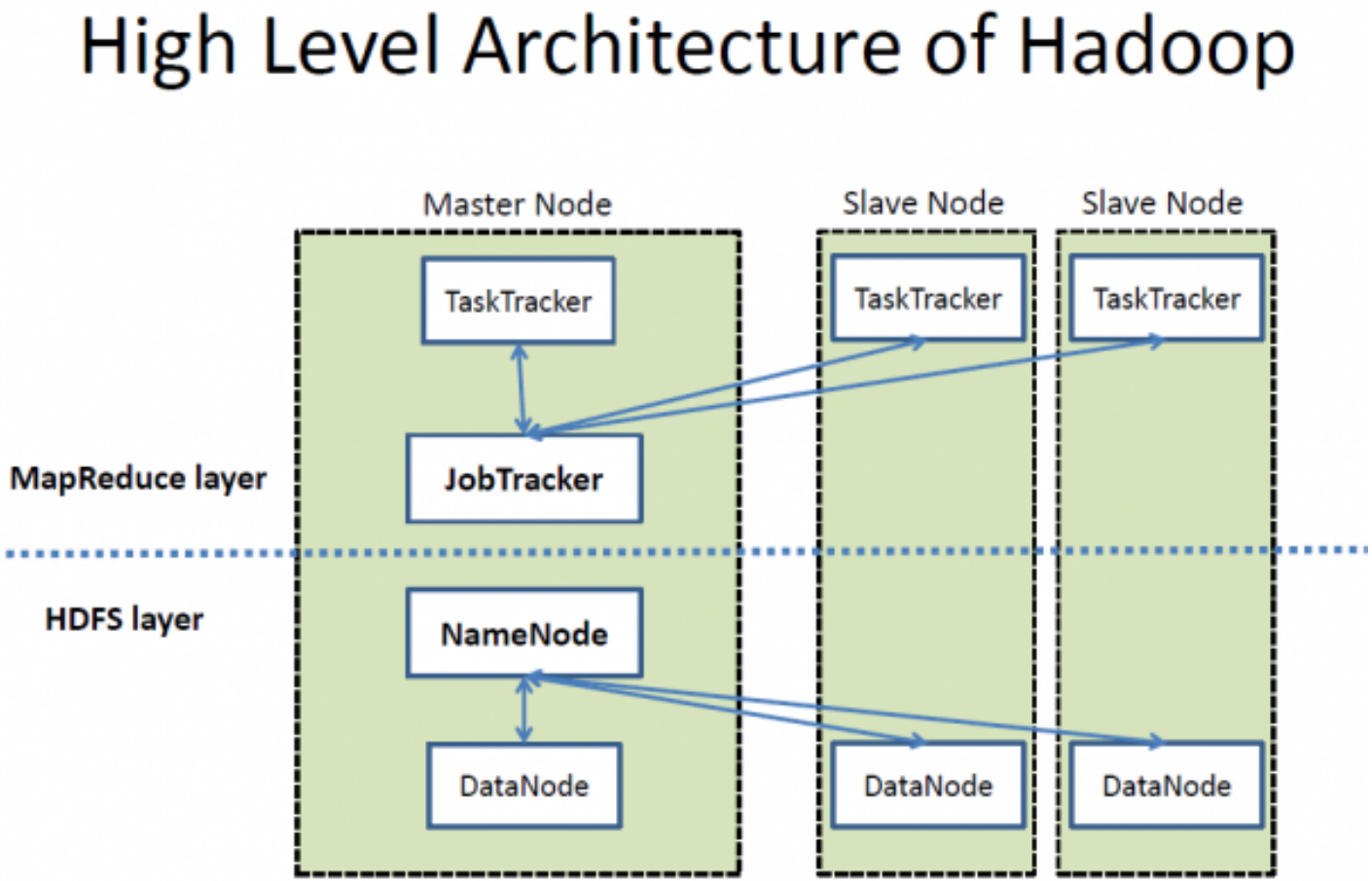
Map/Reduce

Usado cuando podemos dividir una tarea grande en tareas paralelas:



Crear tantos map y reduce nodes como sea necesario.

Hadoop



HDFS = Hadoop Distributed File System

Diseño Arquitectural

- Hay muchas opciones arquitecturales
- . . . y podemos ocupar patrones para satisfacer ciertos requisitos
- ¿Cómo determinar qué arquitectura elegir?

Diseño Arquitectural

- Hay muchas opciones arquitecturales
- . . . y podemos ocupar patrones para satisfacer ciertos requisitos
- ¿Cómo determinar qué arquitectura elegir?
  - Generación de arquitecturas alternativas
    - Necesitamos conocer tecnología existente
    - Saber sus beneficios y desventajas, sus límites
  - Evaluar arquitecturas
    - Para poder comparar
    - Evaluación depende de stakeholders
  - Describir arquitecturas
    - y el por qué se toma cada decisión arquitectural y se descarta otras (rationale)