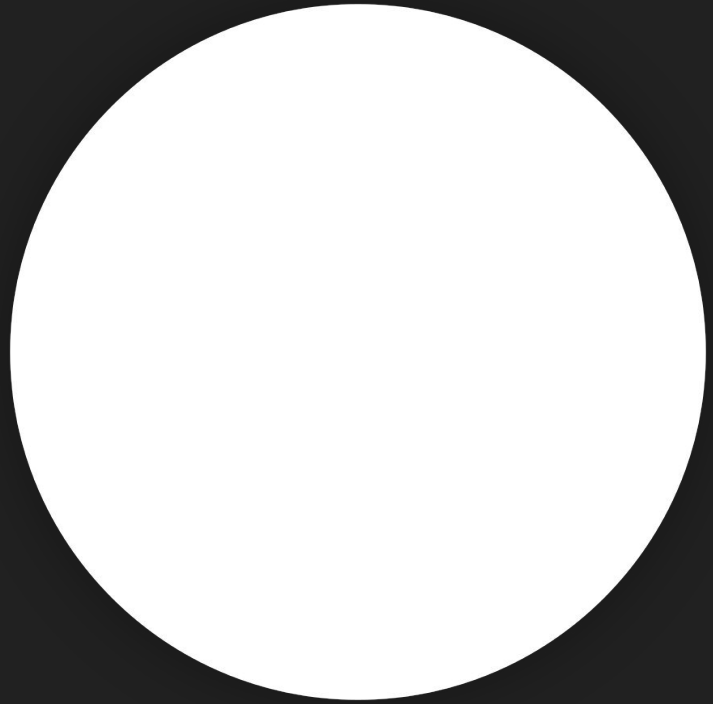


Metarhia

Metarhia

Technology Stack for Highload Applications

Introduction

If we need applied software for following fields:

- Business (corporate information systems including ERP, BI & BPM)
- Economics (including Supply Chain Management, CRM, etc.)
- Commerce and trading (including Real-Time and High-Frequency Trading)
- Enterprise (Knowledge Management, CAD/CAM, Databases)
- Education (University IS, remote learning and e-Learning)
- Medicine (MDS, management, Diagnostics, Equipment control)

- Governance (Databases, Public services, Collaboration)
- Social networking (Instant messaging, Content publishing)
- Communications (Voice/video calls and mail, file exchange)
- Mass media and Highload interactive TV (second screen platform)
- Clusterware for massively parallel distributed cluster/cloud applications
- Production Automation, Telemetry and Internet of Things
- Fog/dew computing (use client devices for distributed data processing)

etc.

We need applications to be at the same time:

- Distributed (multi-server, multi-client, C/S)
- Highload (billions of concurrent users)
- Interactive (bi-directional in real time)
- Secure (both anonymity and reliability)
- Flexible (continuous self-updates)

The Concept

We need following components for applied software:

- Server-side runtime (to execute business-logic)
- Client-side runtime (for b-logic and user interface)
- Inter-process communication with:
  - RPC, MQ, streaming and introspection
  - Client-server, Server-server and P2P interactions
- DBMS designed to be unified, single and globally distributed
- New naming and identification conventions

Metarhia is a new Technological Stack

Experimental

… and alternative.

It should not support backward compatibility.

It can find right way...

It can avoid success at all costs :)

Open Source

It’s free and no vendor dependent.

Anybody can contribute.

It opens opportunity to be used in critical and key projects.

Unistack

Unified and...

Using architectural solutions;

It solves task by design, before they became problems.

Metarhia components

- Impress Application Server for building dedicated private clouds with Node.js
  Ready to use, require docs. https://github.com/metarhia/Impress
- JSTP protocol: RPC, Event BUS, DB Sync, Binary streaming, Multiplexing
  Ready to use, require start guide. https://github.com/metarhia/JSTP
- Includes SDK for mobile and desktop: JavaScript, Java, Swift, C++, Haskell, Python, Objective-C, PHP, GoLang, C# and other languages
- GlobalStorage distributed reactive in-memory Database Management System
  Work in progress, requires funding https://github.com/metarhia/GlobalStorage
- Metasync (library for async I/O and parallel asynchronous programming)
  Work in progress. https://github.com/metarhia/MetaSync
- Console new generation browser (concept to be implemented 2017-2019)
  Planning, requires funding. https://github.com/metarhia/ConsoleMetarhia

Used/related Technologies:

- CentOS for better results
  (or any Linux/Unix/MacOS/Windows server)
- We use TCP/TLS transport and WebSockets (WS/WSS) for web implementation
- Virtual machine V8, libuv
  (async I/O library)
- C++ (for high optimized code) and
  JavaScript (for complex and dynamic logic)

JavaScript Transfer Protocol

JSTP is a family of data formats, libraries and implementations for different languages and platforms based on several assumptions:

- Simple and well-known format. We can take data serialization format already available for multiple platforms. This format is JSON5 compatible and a subset of JavaScript itself.
- Interactive communications and Real-time bidirectional interactions.
  - RPC (Remote Procedure Calls)
  - MQ (Message Queuing) or Event BUS implementation
  - Bi-directional calls and event
  - Establish long-time connection
  - Both statefull (RPC, MQ, DB Sync) and staleless (REST) approaches for API
  - Types of interaction: Client-Server, Server-Server, Client-Client

JavaScript Transfer Protocol

- Reactive data processing.
- Focused on optimization and highload.
  - Avoid HTTP/HTTPS overhead, use TCP/TLS instead
  - Minimization of data transformations. Single data format for:
    - Persistent storage in database,
    - Representation in server memory
    - Serialization and protocol
    - Representation in client memory
    - Intersystem data exchange
- Use metadata everywhere: for schema model, for data representation minimization, for flexibility.

GlobalStorage DBMS

- Minimize IPC (Interprocess communication)
  - Built-in database to work in process of Application server
  - Built-in database to work locally in Client-side process
- Minimize data repacking and transform
- Use metadata and metamodel
- Work online and offline with sync
- Calculations on insert principle (for speedup queries)

GlobalStorage DBMS

- Use database-centric principle with Optimistic sync, OT, Patch based or CRDT approach
- Use reactive data processing (all data in DB are connected with event-function chains)
- Distributed queries execution (executed where data placed, transmitted and combined where data needed)
- Automatic indexing: no need to define indexing manually, DBMS will build optimal index based on statistics

GlobalStorage DBMS

It is hybrid DBMS API, now we have in-memory DBMS and no own persistent storage implementation but we use MongoDB for persistent storage and going to create PostgreSQL and Oracle storage providers. It is not critical for in-memory database so we also want to create own persistent storage engine based on filesystem or raw partition.

Why?

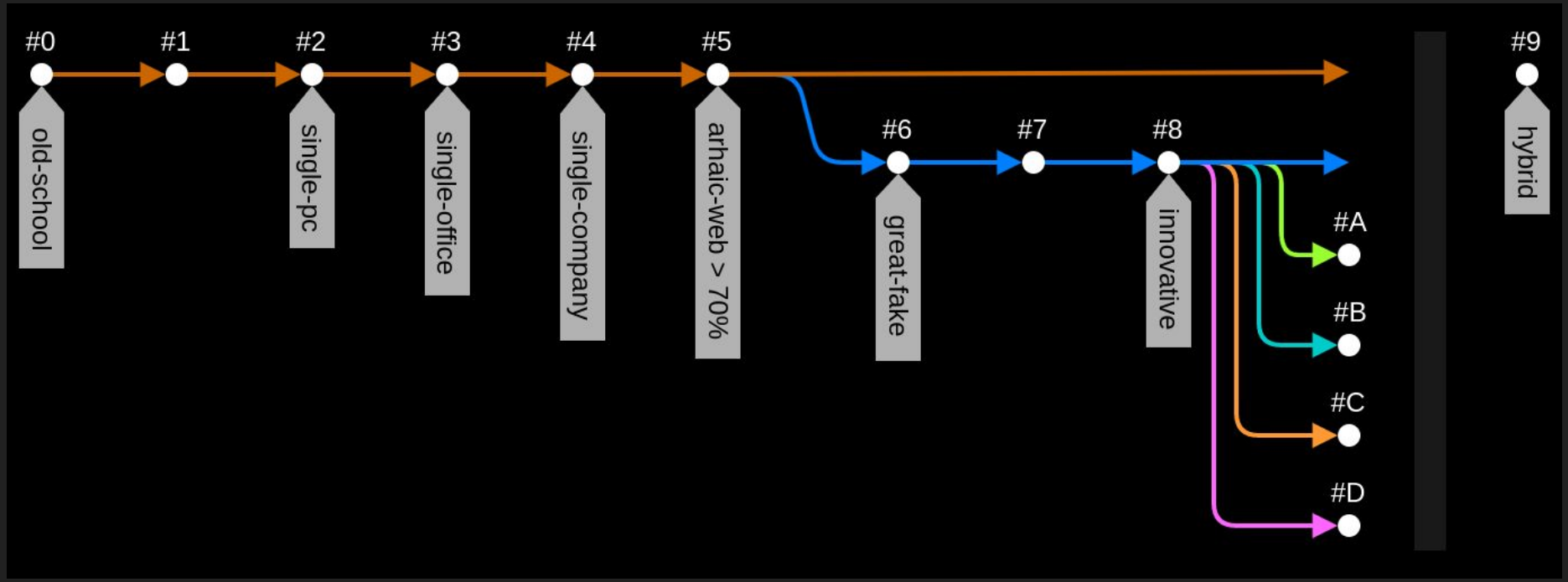
Problems of
the Software Engineering

See articles here

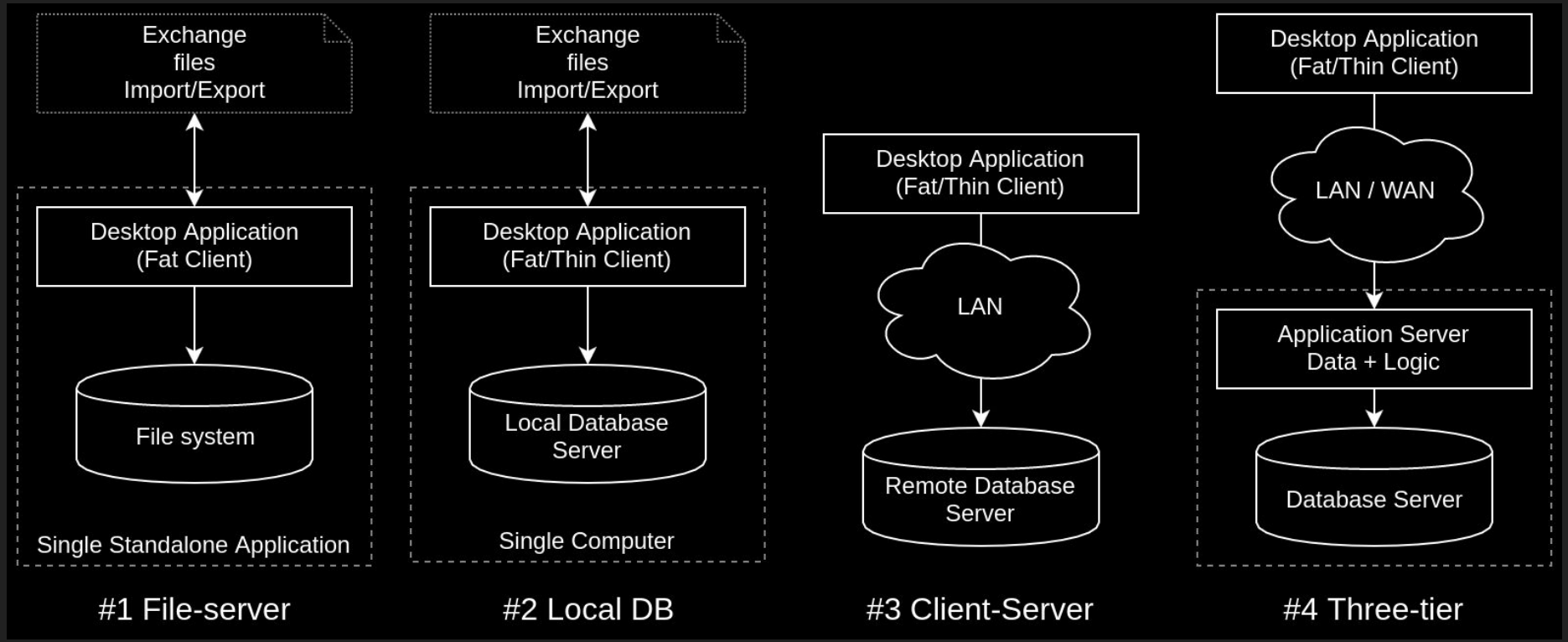
Node.js & JavaScript after the Web death
https://habrahabr.ru/post/306584/

Evolution of applications or where we go
https://habrahabr.ru/post/326016/

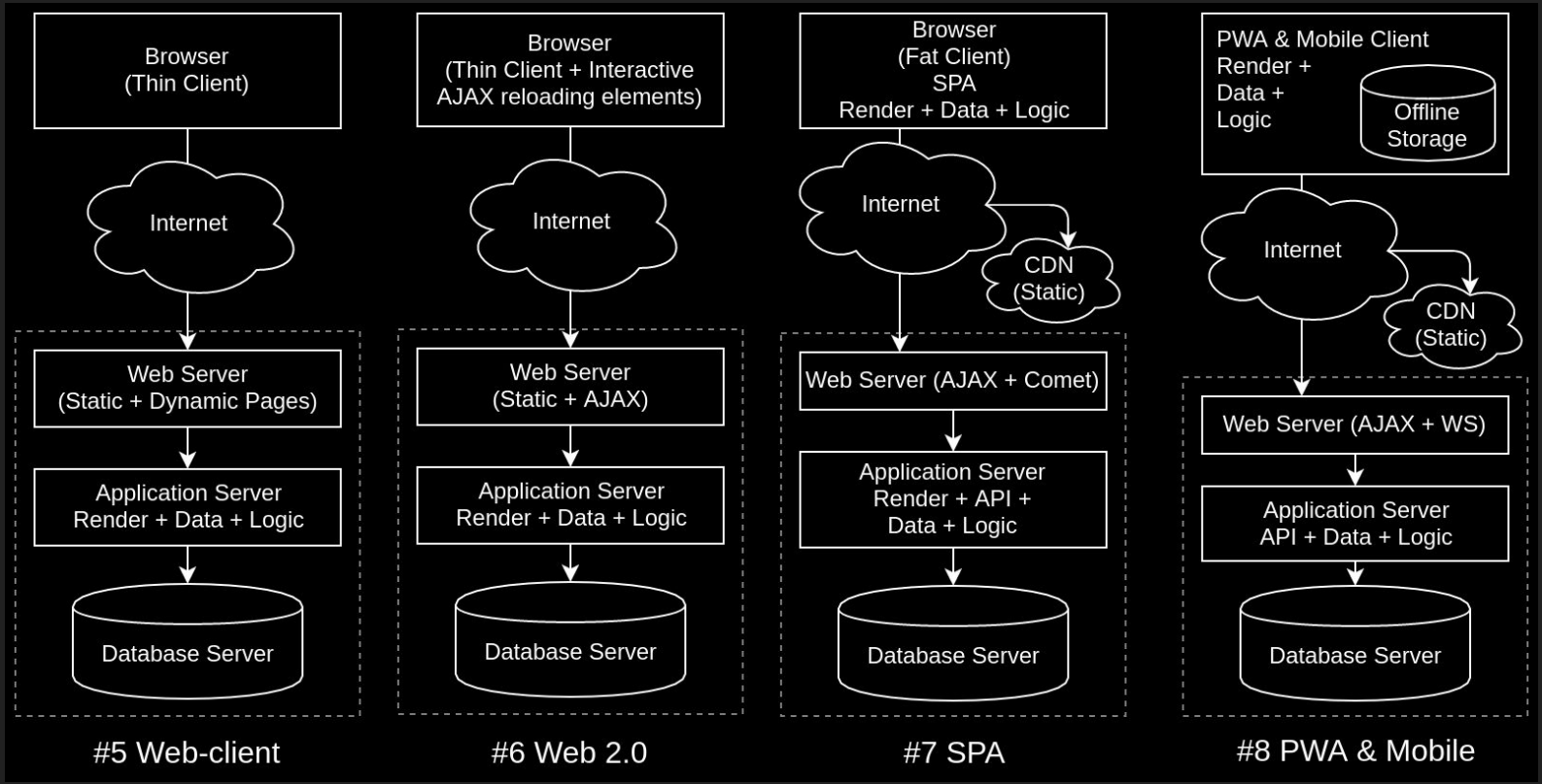
Evolution milestones



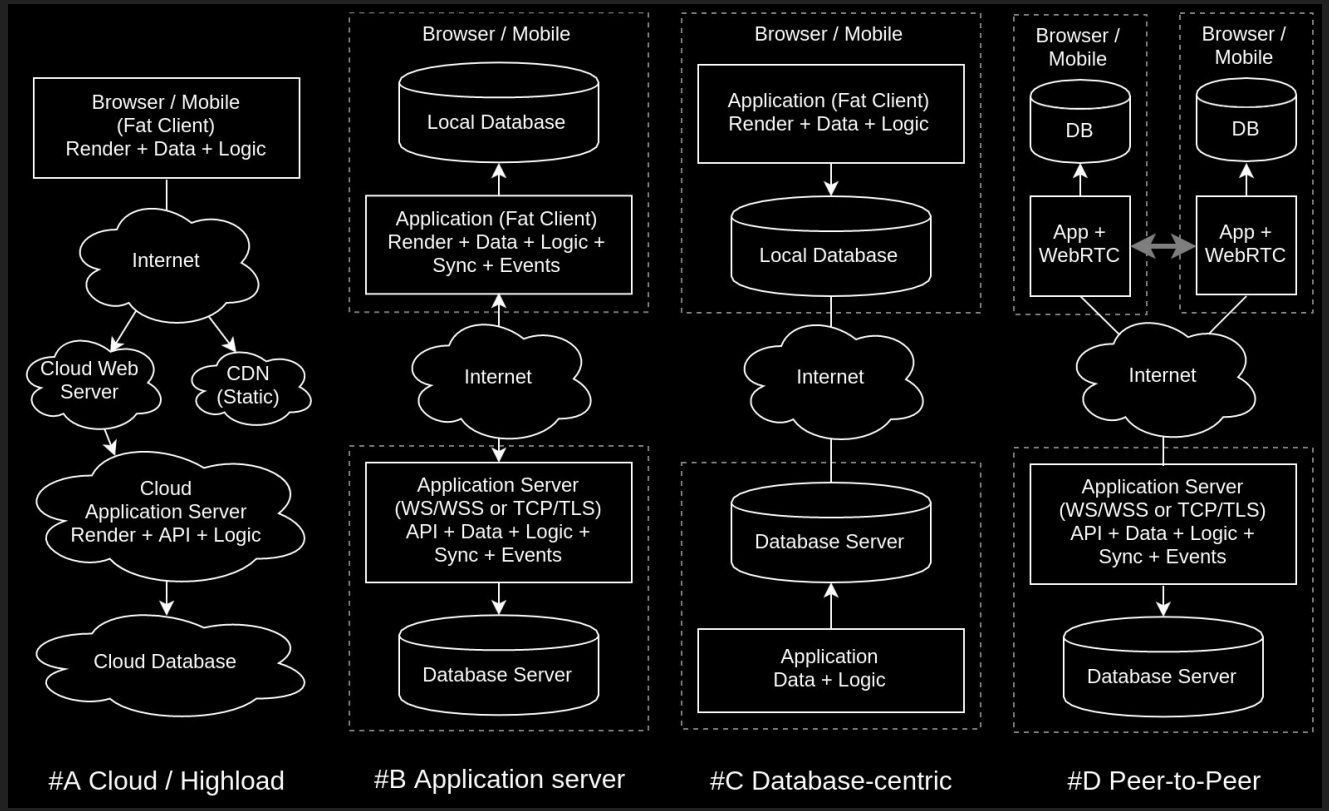
Early days


Exchange files Import/Export
Desktop Application (Fat Client)
File system
Single Standalone Application
#1 File-server
Exchange files Import/Export
Desktop Application (Fat/Thin Client)
Local Database Server
Single Computer
#2 Local DB
Desktop Application (Fat/Thin Client)
LAN
Remote Database Server
#3 Client-Server
Desktop Application (Fat/Thin Client)
LAN / WAN
Application Server Data + Logic
Database Server
#4 Three-tier

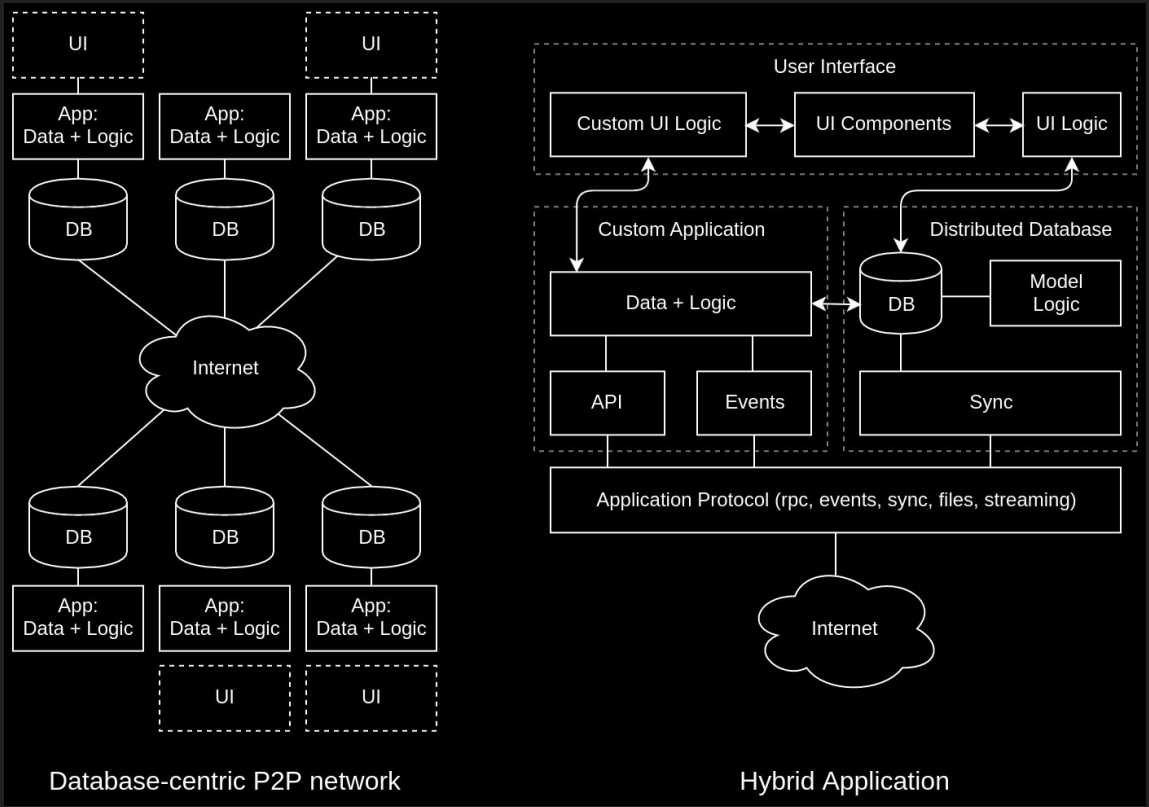
Nowadays


Browser
(Thin Client)
Internet
Web Server
(Static + Dynamic Pages)
Application Server
Render + Data + Logic
Database Server
#5 Web-client
Browser
(Thin Client + Interactive
AJAX reloading elements)
Internet
Web Server
(Static + AJAX)
Application Server
Render + Data + Logic
Database Server
#6 Web 2.0
Browser
(Fat Client)
SPA
Render + Data + Logic
Internet
CDN
(Static)
Web Server (AJAX + Comet)
Application Server
Render + API +
Data + Logic
Database Server
#7 SPA
PWA & Mobile Client
Render +
Data +
Logic
Offline
Storage
Internet
CDN
(Static)
Web Server (AJAX + WS)
Application Server
API + Data + Logic
Database Server
#8 PWA & Mobile

Alternatives



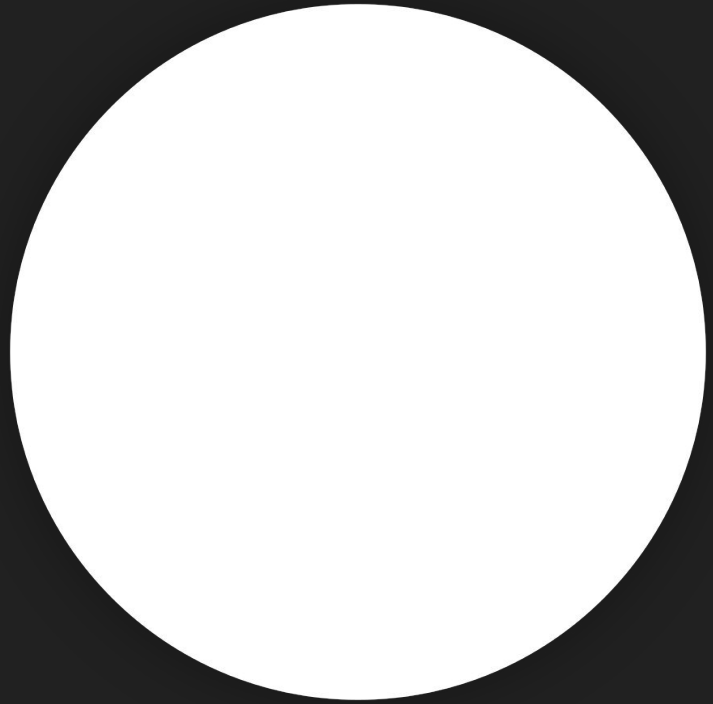
Future...


UI
UI
App:
Data + Logic
App:
Data + Logic
App:
Data + Logic
DB
DB
DB
Internet
DB
DB
DB
App:
Data + Logic
App:
Data + Logic
App:
Data + Logic
UI
UI
Database-centric P2P network
User Interface
Custom UI Logic
UI Components
UI Logic
Custom Application
Distributed Database
Data + Logic
DB
Model
Logic
API
Events
Sync
Application Protocol (rpc, events, sync, files, streaming)
Internet
Hybrid Application

Features

- Data Synchronization
- Offline
- Interactivity
- Low latency / availability
- Highload
- High connectivity
- High interconnection
- Scalability
- Big data
- Big memory & in-memory DB
- Integration flexibility

Questions



Metarhia