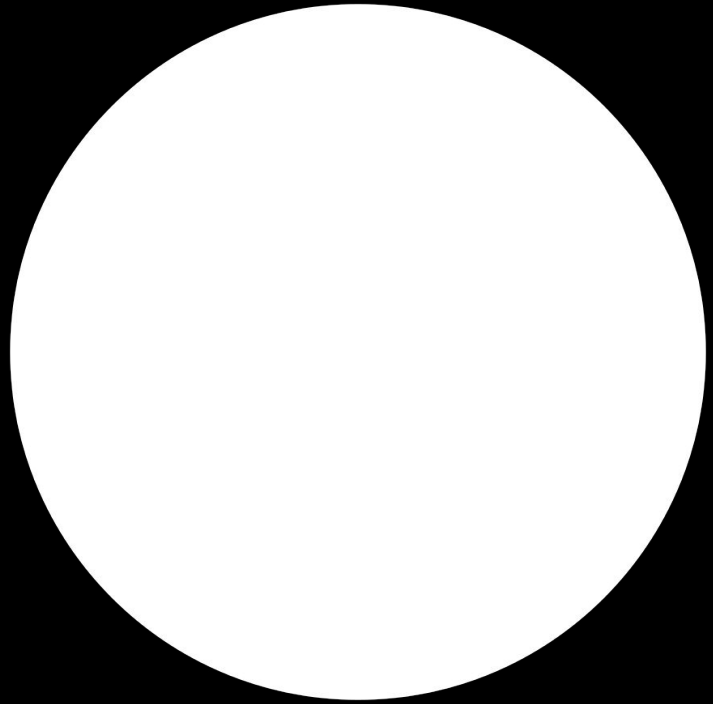


Metarhia

Metarhia

Технологический стек
для распределенных высоконагруженных
приложений и баз данных

- 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

Основные идеи

- Унификация компонентов
- Однородность узлов
- Отказ от обратной совместимости
- Открытый код
- Независимость от вендора
- Архитектурные решения
- Организована подготовка кадров

Требования к стеку

- Распределенность (межсерверная, клиентская, оффлайн)
  В том числе распределенное хранение с обеспечением интеграции в любое время
  Каждый госорган может хранить свою часть данных, при общей целостности данных
- Высоконагруженность (оптимизация, линейная масштабируемость)
  Поток запросов, кол-во соединений, взаимодействие между соединениями
- Интерактивность (двустороннее взаимодействие реального времени)
- Безопасность (глубоко встроена в механизмы хранения, передачи и обработки)
  Невозможно сверху “посолить” систему безопасностью
- Гибкость (динамическа перенастройки и непрерывное обновление)
  Миграции, доступ к истории, зеркалирование вместо резервного копирования

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

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++ and Rust (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

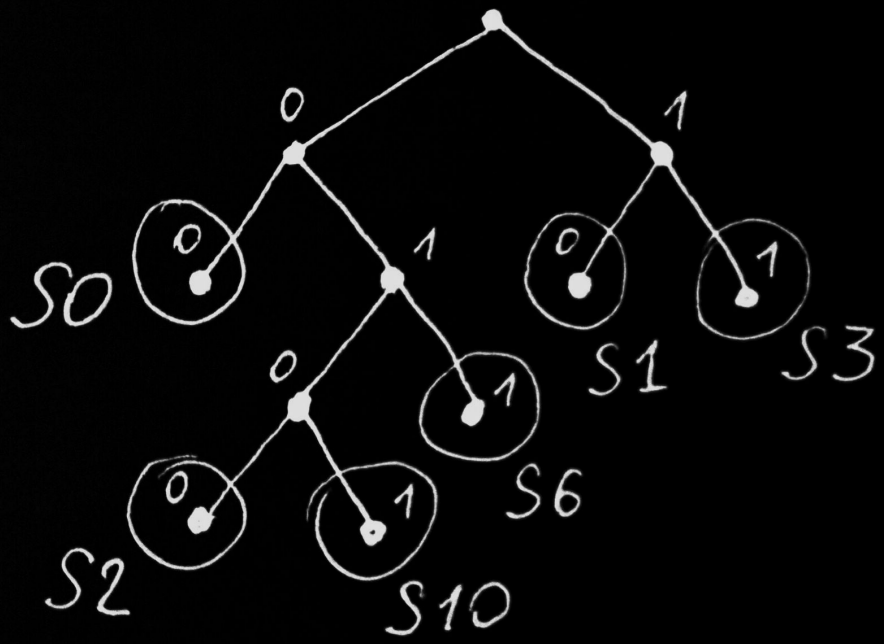
База данных

- Все инстансы составляют одну базу данных
- Сквозная бесконечная идентификация объектов
- Бинарное дерево серверов и клонирование
- Использование оперативной памяти
- Специальный язык запросов и API
- Реактивные и связанные курсоры

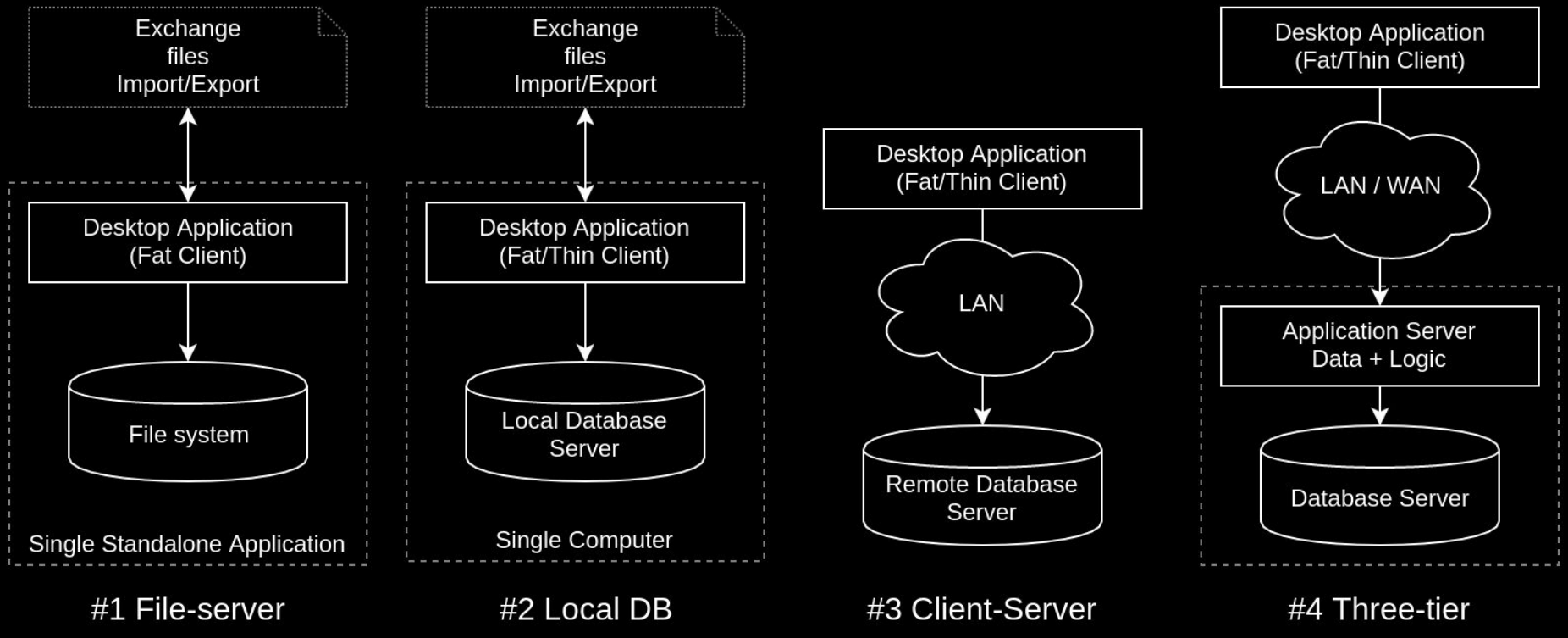
Направления исследований

- Репозиторий категорий (типов)
- Непрерывные миграции
- Ленивые курсоры и отложенные запросы
- Потоки событий в БД
- Операции с множествами, CRDT, OT
- Масштабирование и распределенные запросы
- Распределенные безопасные индексы и хеши
- Длительные транзакции

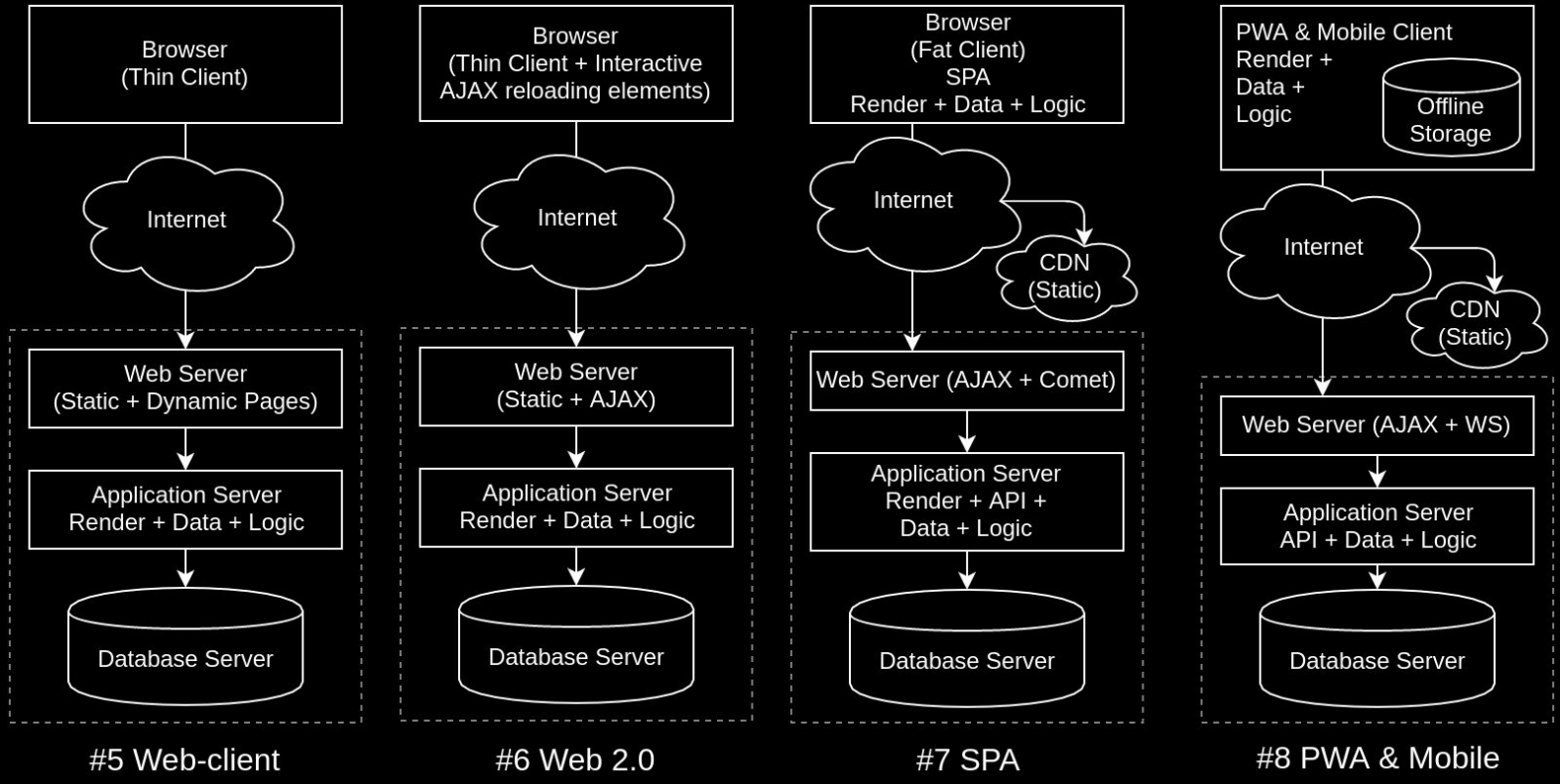
Бинарное дерево серверов и идентификаторов



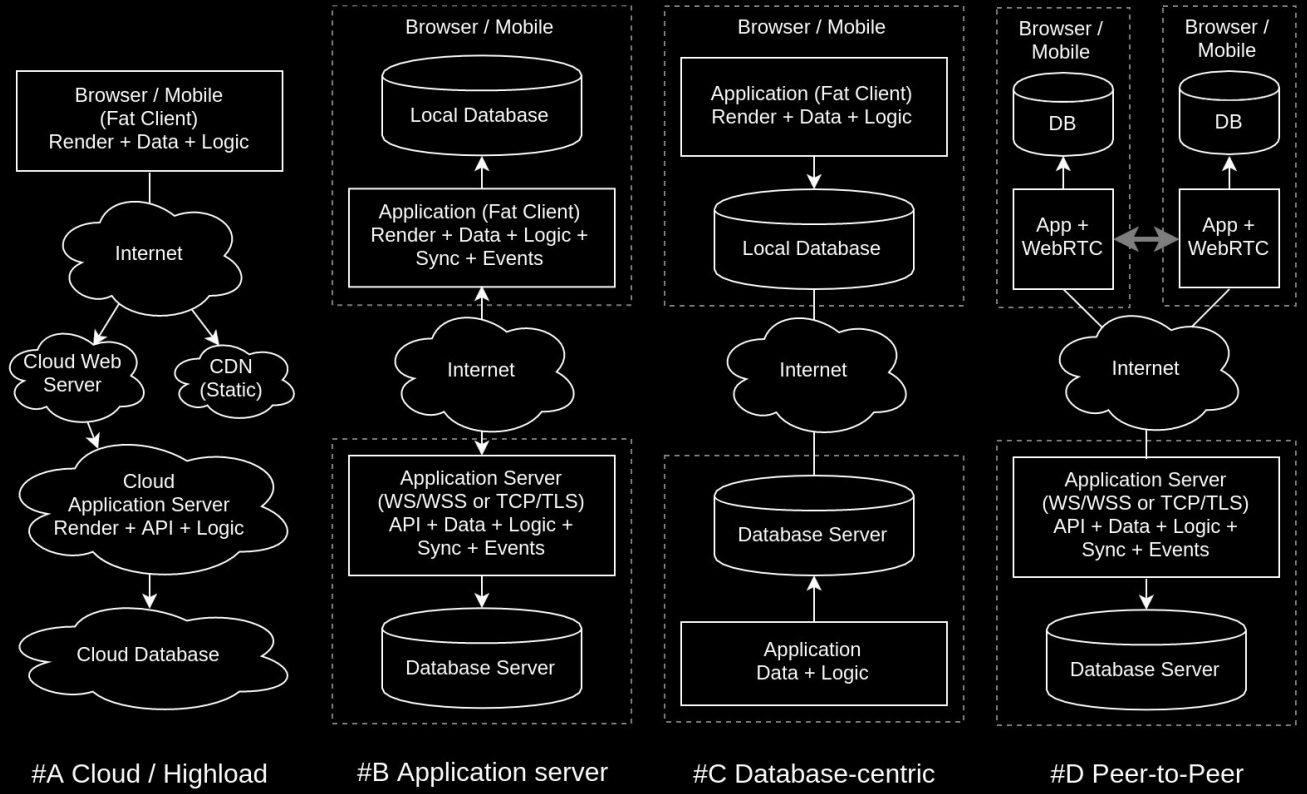
Early days


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



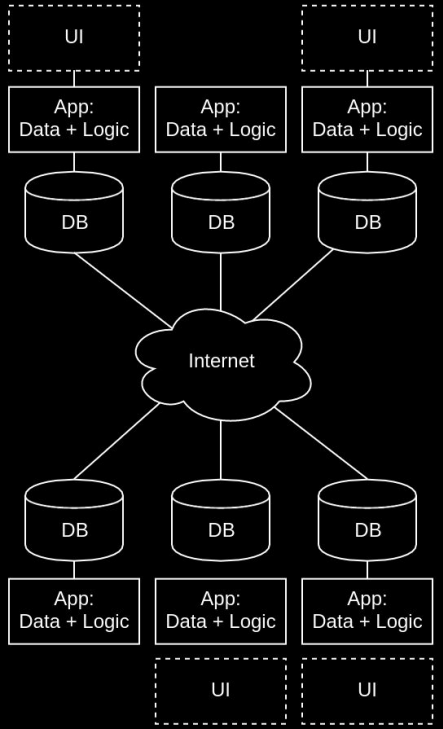
#A Cloud / Highload

#B Application server

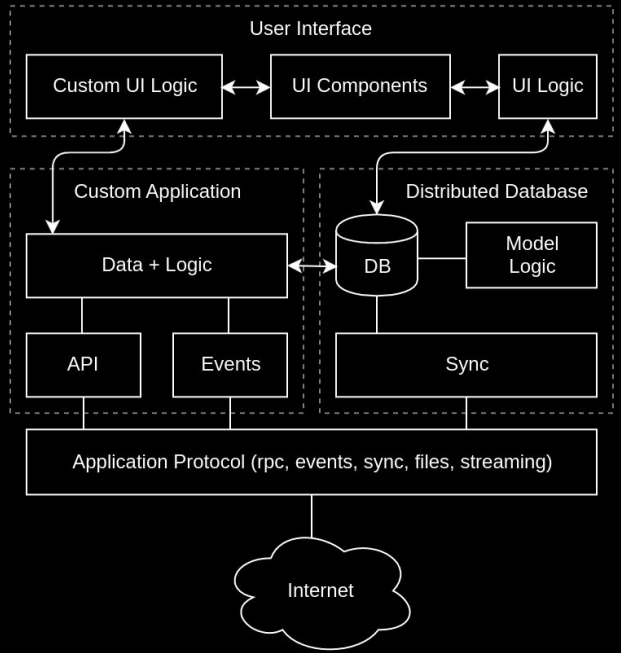
#C Database-centric

#D Peer-to-Peer

Future...



Database-centric P2P network

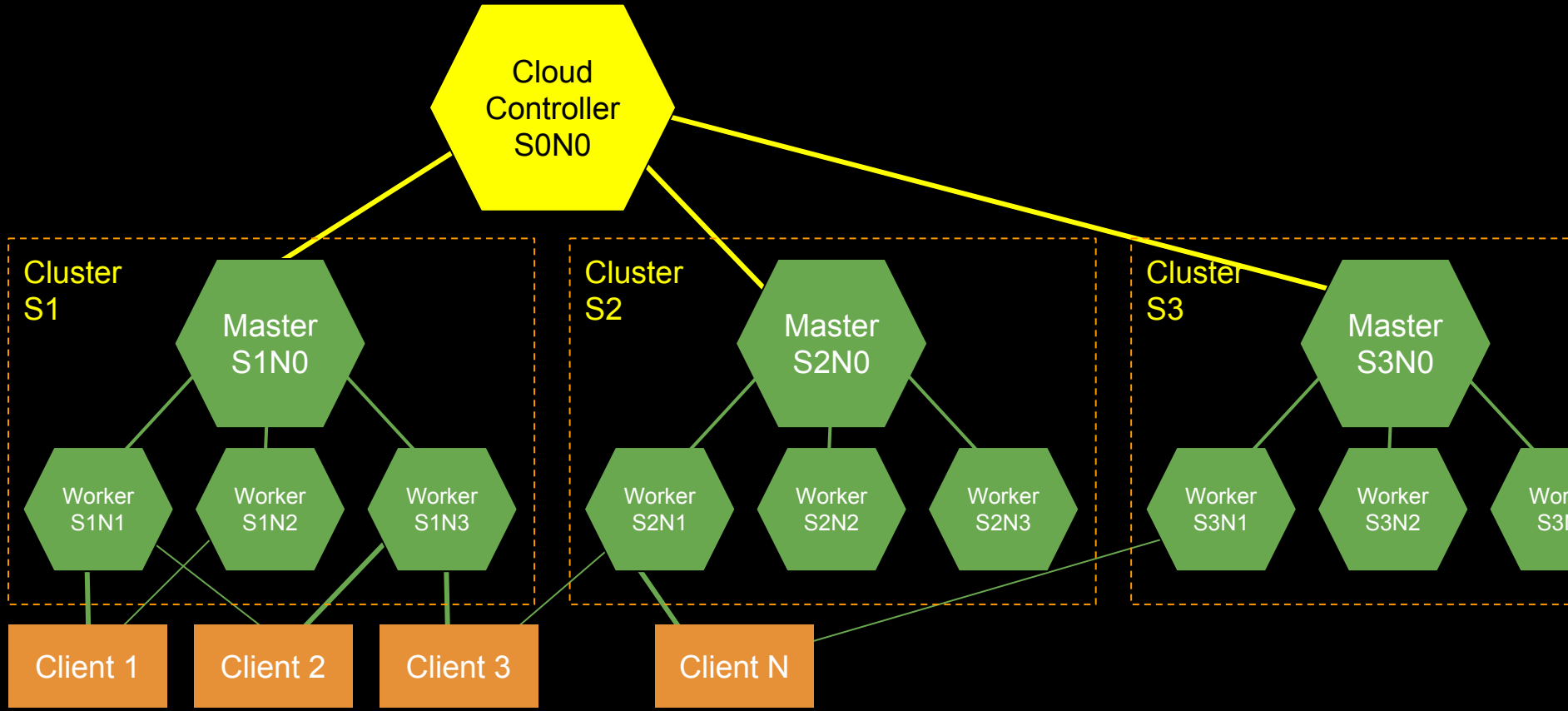


Hybrid Application

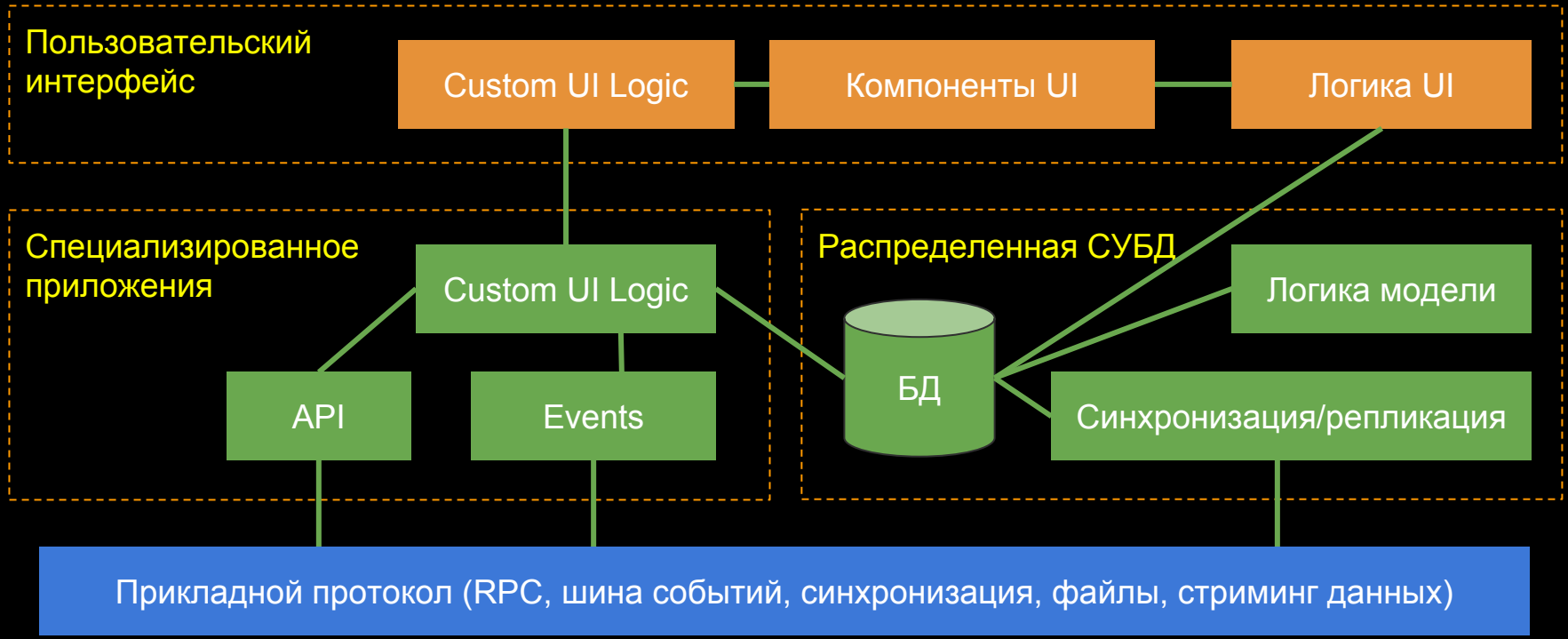
Реактивные запросы к БД Globalstorage

```
gs.select({ category: 'Person', name: 'Marcus' })
  .filter(person => person.isImperor())
  .projection({
    name: ['name', toUpperCase],
    age: ['birth', toAge],
    place: ['address', getCity, getLatLon],
  })
  .order('age')
  .on('update', data => api.log(data.toString()))
  .clone().group('age')
  .map(x => x.count + ' imperors of age ' + x.age)
  .pipe(destinationCursor)
  .fetch((err, data) => Maybe(data)
    .map(api.jstp.emit('imperorsReportUpdated', data))
  )
  .clone().count().fetch((err, count) => Maybe(count)
    .map(api.log('Recalculated records: ' + count))
  );
```

Архитектура приватного облака ЕИС

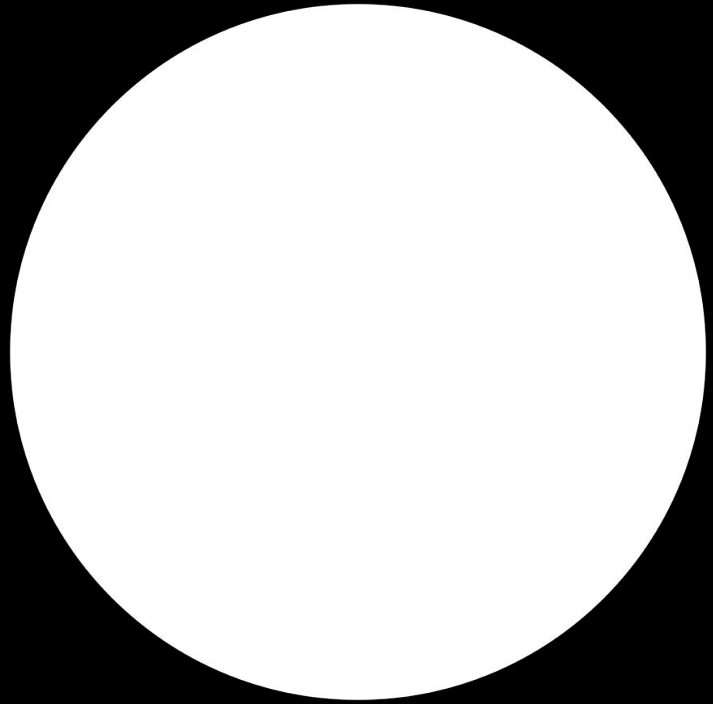


Архитектура приложения в стеке Metarhia



Features

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



Metarhia