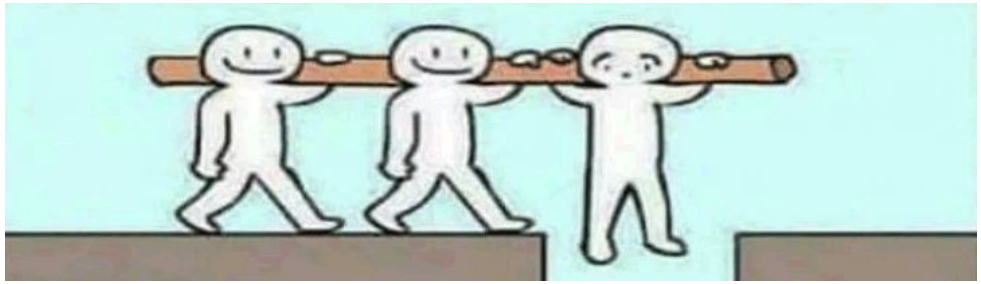


p2p-FS

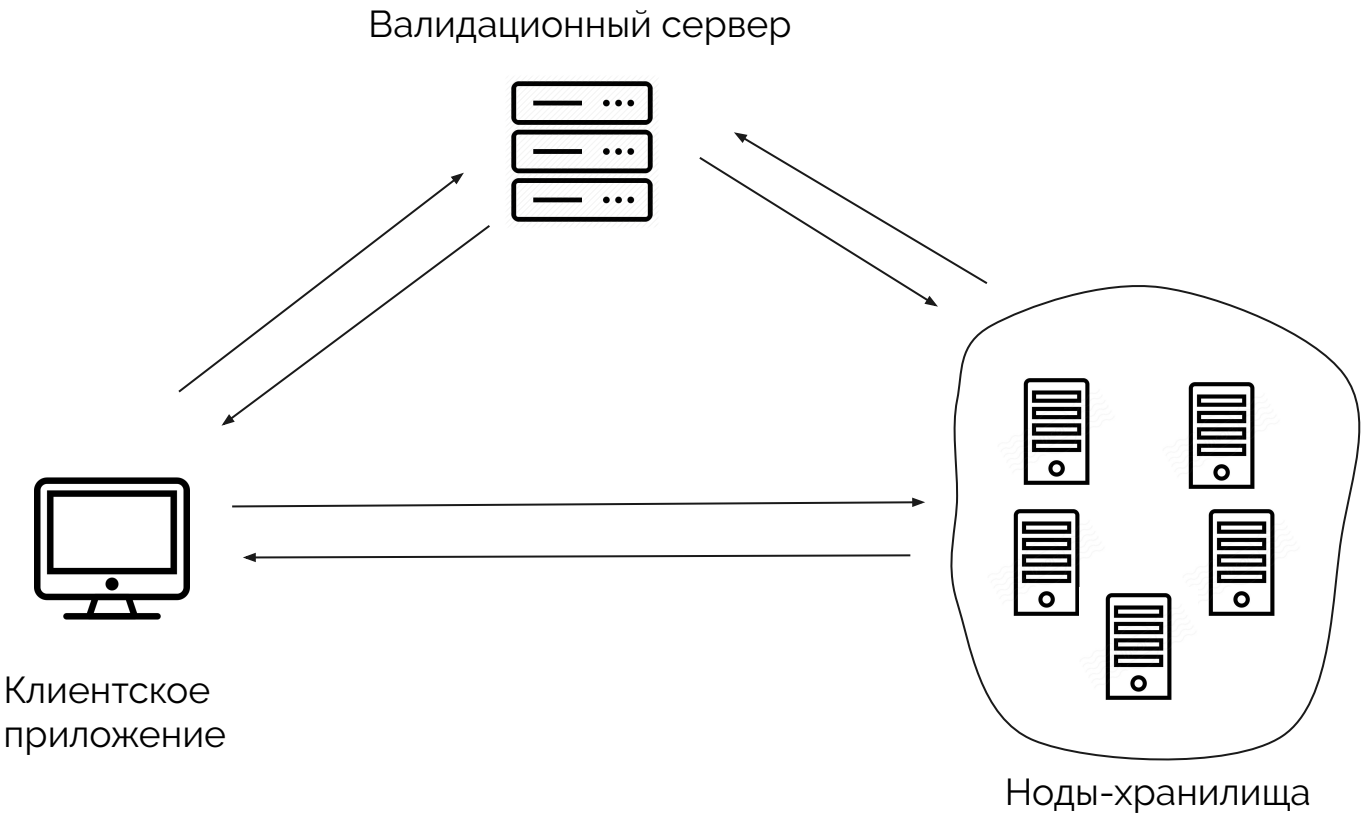
Распределенное хранение файлов

Что это такое?

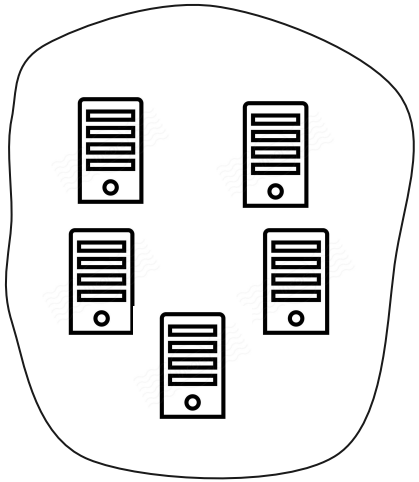
Файловое хранилище, которое:

- Хранит файлы распределённо
- Устойчиво к сбоям и падениям нод
- Поддерживает приватность файлов
- Ограничивает используемое юзерами пространство

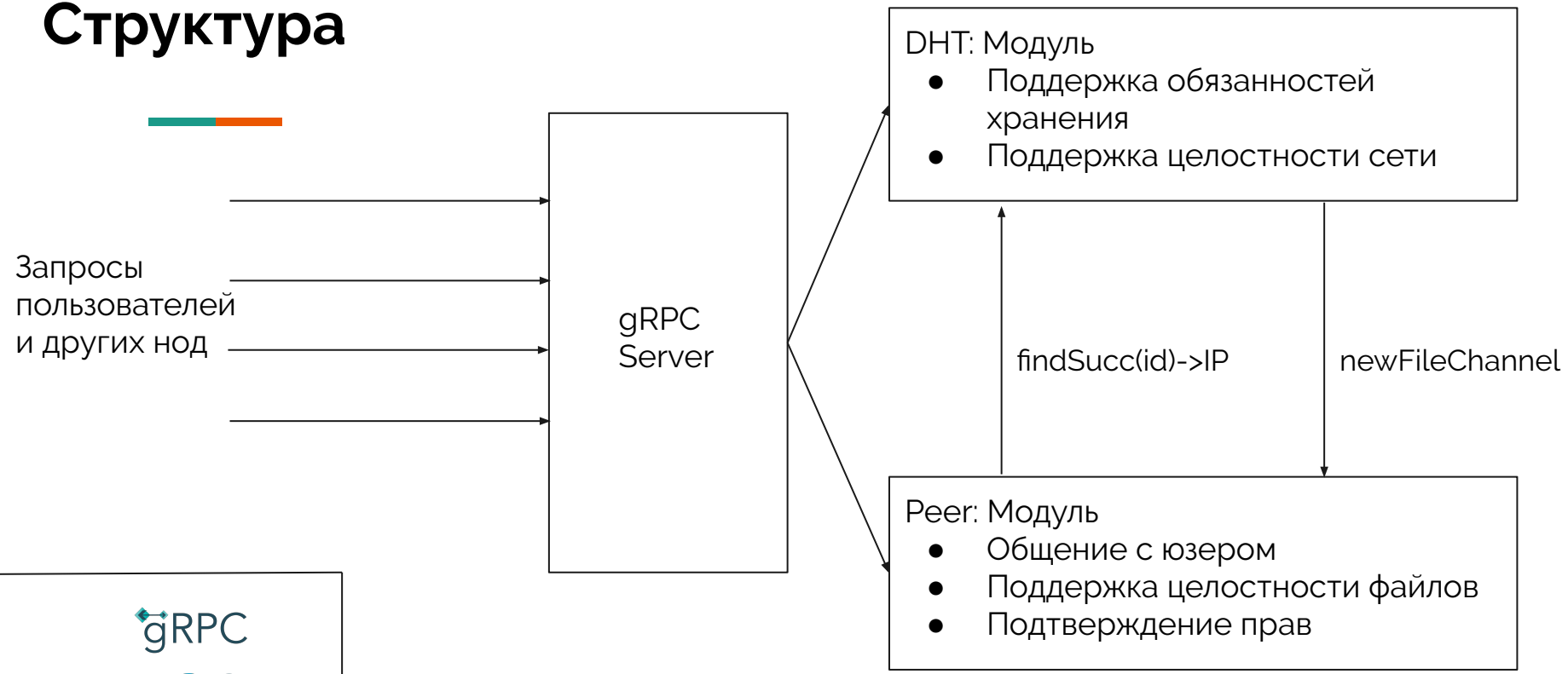
Архитектура



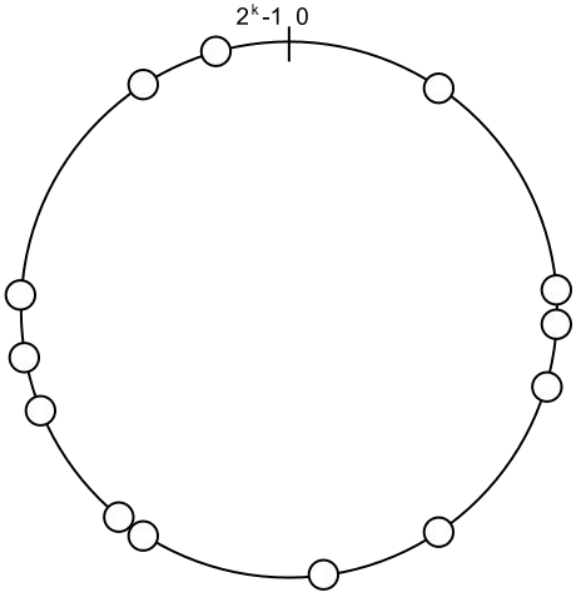
Ноды



Структура


DHT: Модуль
Поддержка обязанностей хранения
Поддержка целостности сети
Запросы пользователей и других нод
gRPC Server
findSucc(id)->IP
newFileChannel
Peer: Модуль
Общение с юзером
Поддержка целостности файлов
Подтверждение прав
Стэк:
gRPC
GO
protobuf
Protocol Buffers

DHT



DHT: Chord

HASH(file_name)

HASH(ip)

→ id

Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

Ion Stoica; Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan†
MIT Laboratory for Computer Science
chord@lcs.mit.edu
http://pdos.lcs.mit.edu/chord/

Abstract

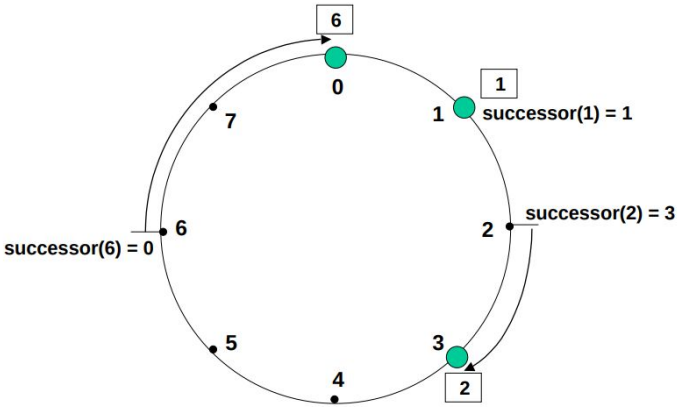
A fundamental problem that confronts peer-to-peer applications is to efficiently locate the node that stores a particular data item. This paper presents *Chord*, a distributed lookup protocol that addresses this problem. Chord provides support for just one operation: given a key, it maps the key onto a node. Data location can be easily implemented on top of Chord by associating a key with each data item, and storing the key/data item pair at the node to which the key maps. Chord adapts efficiently as nodes join and leave the system, and can answer queries even if the system is continuously changing. Results from theoretical analysis, simulations, and experiments show that Chord is scalable, with communication cost and the state maintained by each node scaling logarithmically with the number of Chord nodes.

1. Introduction

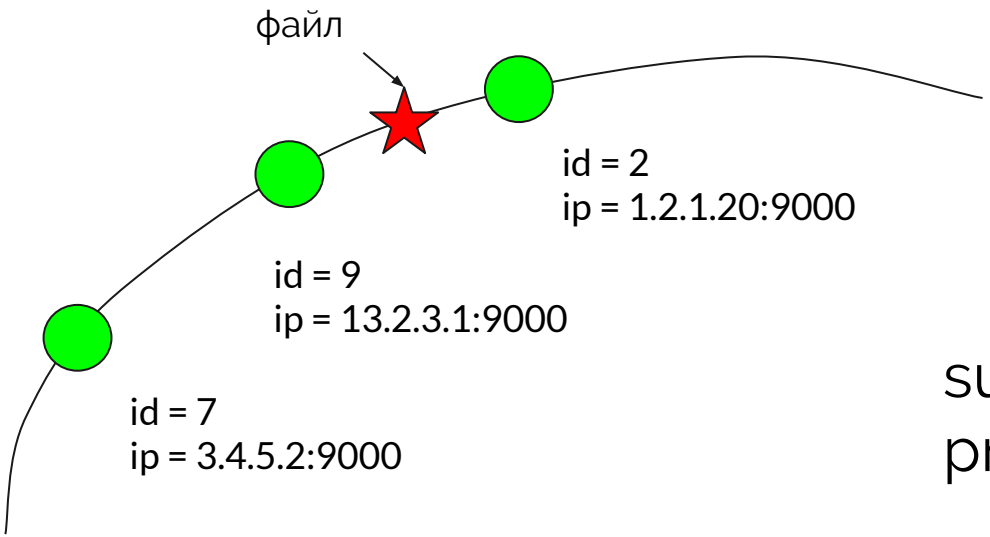
Peer-to-peer systems and applications are distributed systems without any centralized control or hierarchical organization, where the software running at each node is equivalent in functionality. and involves relatively little movement of keys when nodes join and leave the system.

Previous work on consistent hashing assumed that nodes were aware of most other nodes in the system, making it impractical to scale to large number of nodes. In contrast, each Chord node needs "routing" information about only a few other nodes. Because the routing table is distributed, a node resolves the hash function by communicating with a few other nodes. In the steady state, in an N-node system, each node maintains information only about $O(\log N)$ other nodes, and resolves all lookups via $O(\log N)$ messages to other nodes. Chord maintains its routing information as nodes join and leave the system; with high probability each such event results in no more than $O(\log^2 N)$ messages.

Three features that distinguish Chord from many other peer-to-peer lookup protocols are its simplicity, provable correctness, and provable performance. Chord is simple, routing a key through a sequence of $O(\log N)$ other nodes toward the destination. A Chord node requires information about $O(\log N)$ other nodes for *efficient* routing, but performance degrades gracefully when that information is out of date. This is important in practice because nodes will



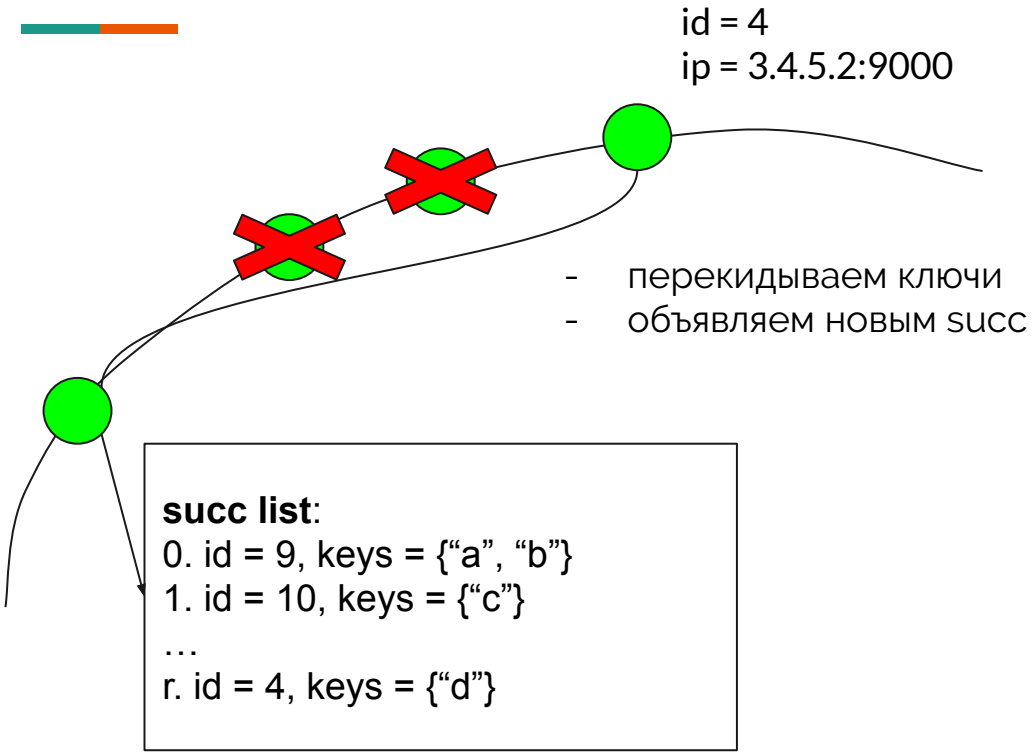
DHT: Поиск



succ
pred

→ Поиск за O(n)

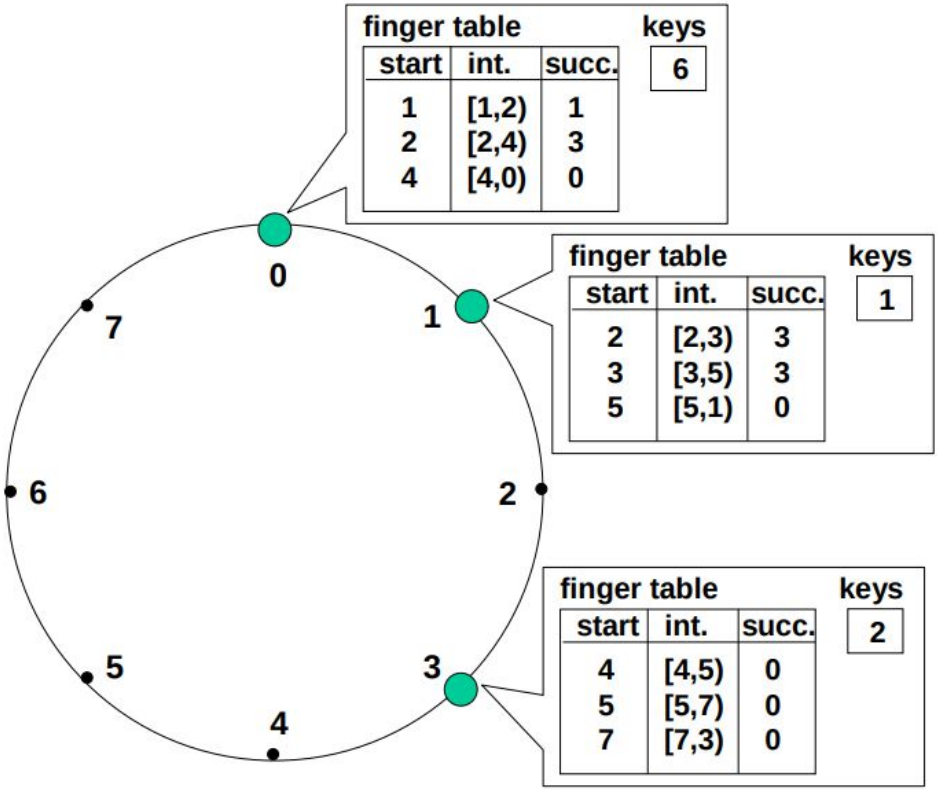
DHT: Падение нод



$$p^r \leq \epsilon$$

p - вероятность ноды упасть за время t
eps - вероятность поломки систем

DHT: Ускорение

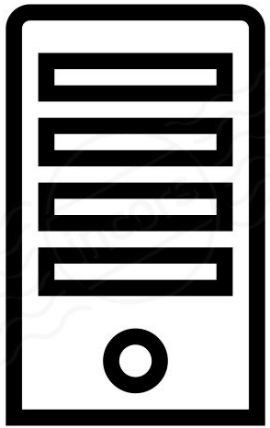


i-ая запись в таблице

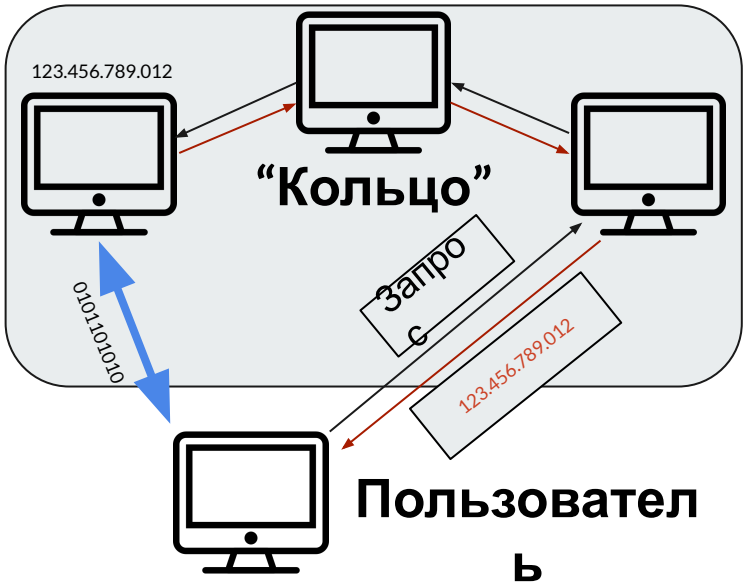
$$s = successor(n + 2^{i-1})$$

O(log(n))

Peer



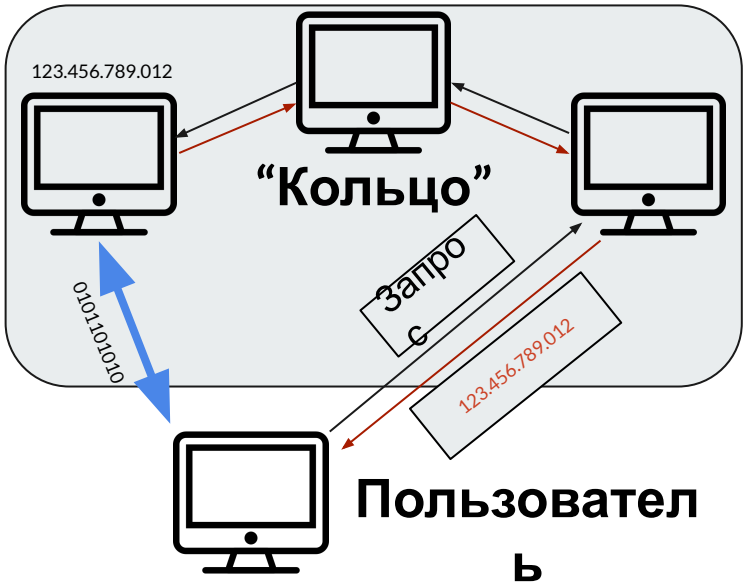
Репликация



D	D	D	D	P
D	D	D	D	P

8,2 репликация

Репликация



D	D	D	D	P
D	D	D	D	P

8,2 репликация

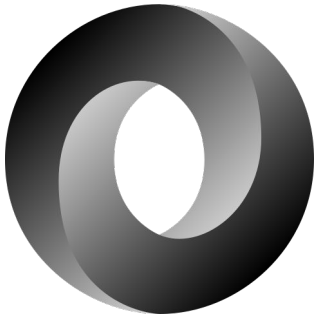
Юнит тесты

Покрытие кода: 70%

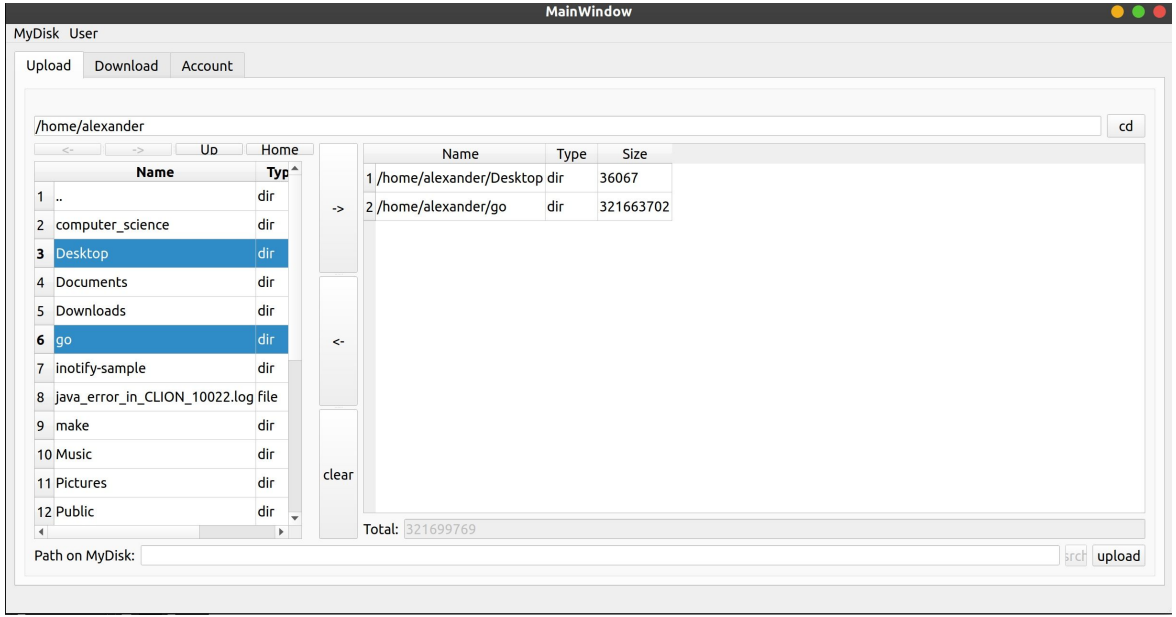
Клиент

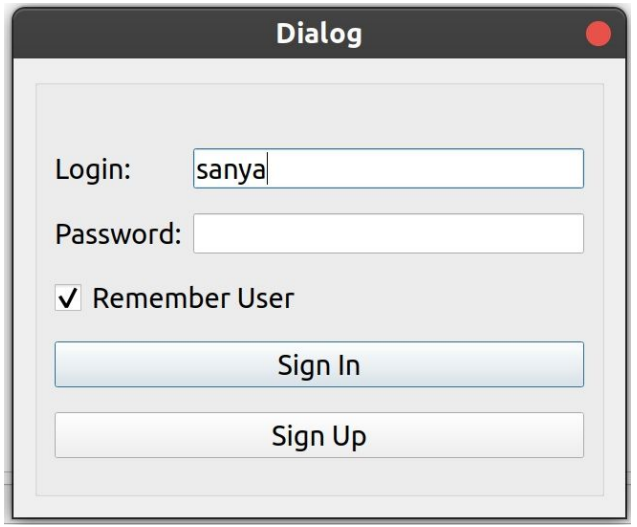
Разработка клиентской части приложения

- Консольное приложение: C/C++
- Динамические библиотеки загрузки и выгрузки файлов: C++
- Динамическая библиотека установки процессов-демонов (не вошла в окончательную сборку): C
- GUI: C++, QT (Widgets, Network, JSON)

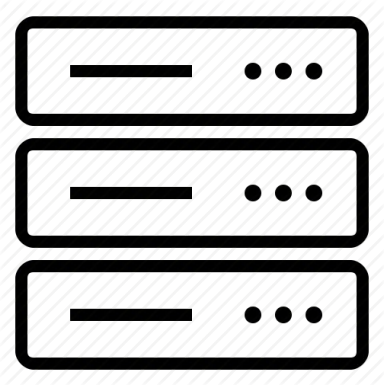


Graphical user interface



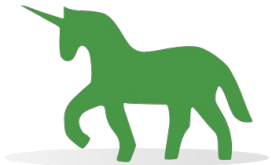


Сервер



Разработка серверной части

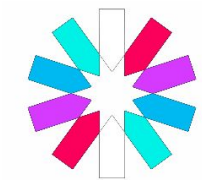
- Backend: Python + Flask
- Database: PostgreSQL
- Worker model: Gunicorn
- Authentication: JWT



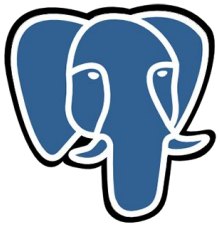
gunicorn

Flask

web development,
one drop at a time



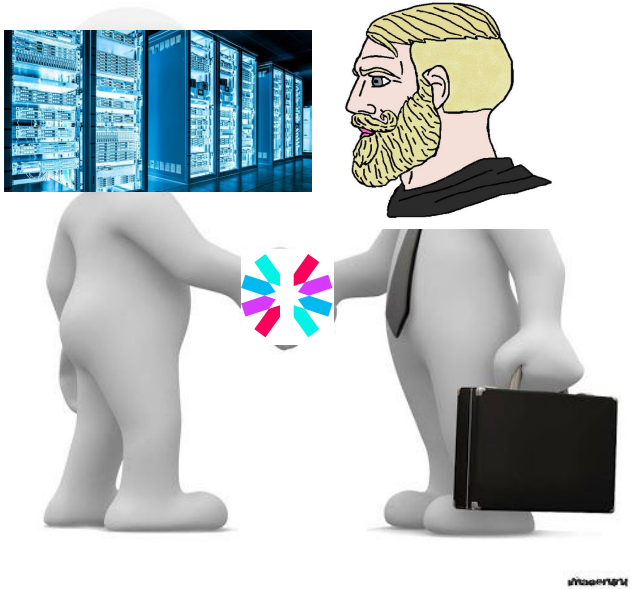
JWT

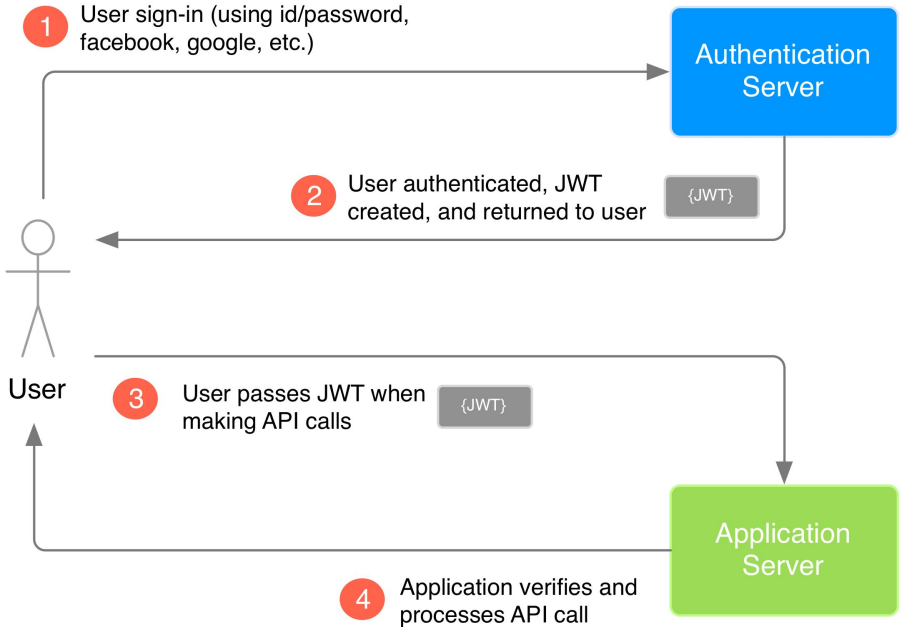


PostgreSQL

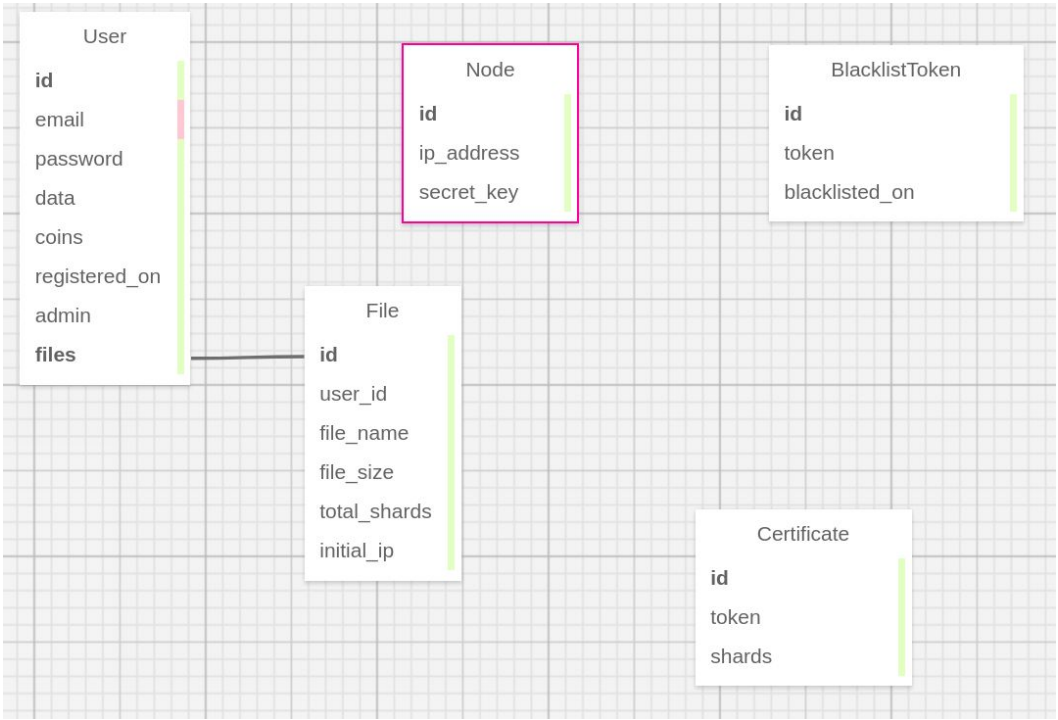
python™

Сертификаты





Немного про Database



Команда по спасению ваших файлов

Ващилко
Александр

Кулешов
Илья

Персиянов
Михаил

Манайнен
Максим

ФУПМ. 2020.


ВНИМАНИЕ!
СПАСИБО ЗА ВНИМАНИЕ!