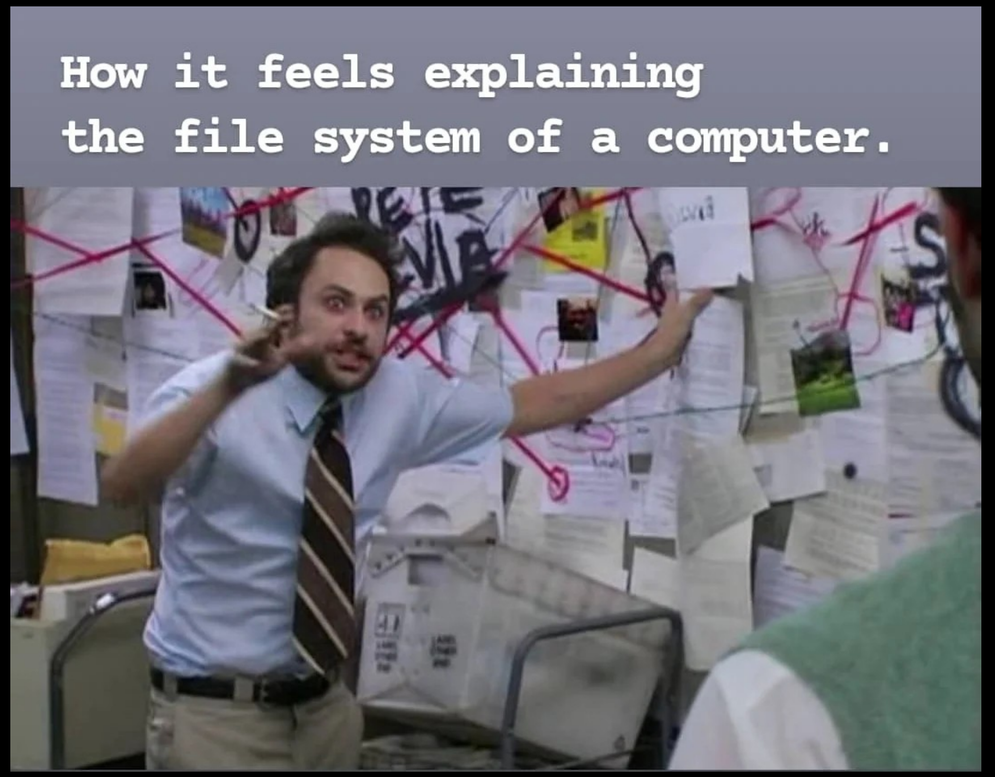
avito.tech Москва 2025

Code Fest

Эволюция файловых систем: от Web2 к Web3





Марат Зимнуров

Марат Зимнуров

Руководитель группы разработки в Avito.tech

- ***fun of:*** bsd, wow hc, genetic algorithms, functional programming
- ***education:*** ассистент кафедры ИТиЦЭ в ИГХТУ
- ***role:*** lead full-stack engineer | team/tech/dev lead engineer
- ***project:*** digitable.ru
- ***ex-worker:*** Microsoft, Adjust, Roadmunk, Xsolla, Tproger, Kontur, EPAM Anywhere, Akvelon



github: https://github.com/the-homeless-god

tg: https://github.com/the_homeless_god

linkedin: https://www.linkedin.com/in/marat-z/

habr: https://habr.com/ru/users/the_homeless_god/

Идеального решения по хранению данных НЕТ

Если, конечно, вообще ничего не хранить



Хранилища Web2: Централизация и хаос

4

2

3

1



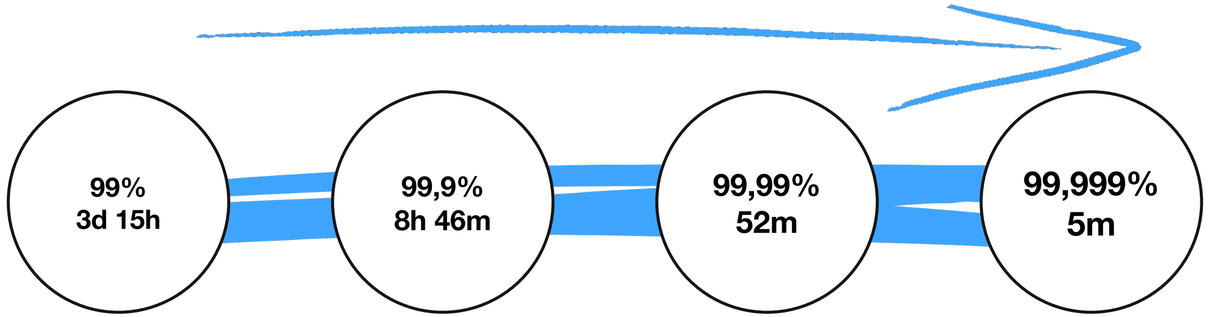
ХРАНИЛИЩА WEB2

Классический Web2-кошмар, не правда ли?

- Централизованные дата-центры
- Single point of failure
- Утечки данных
- Зависимость от единого провайдера
- Система не выдерживает нагрузку



Availability% VS Downtime per Year

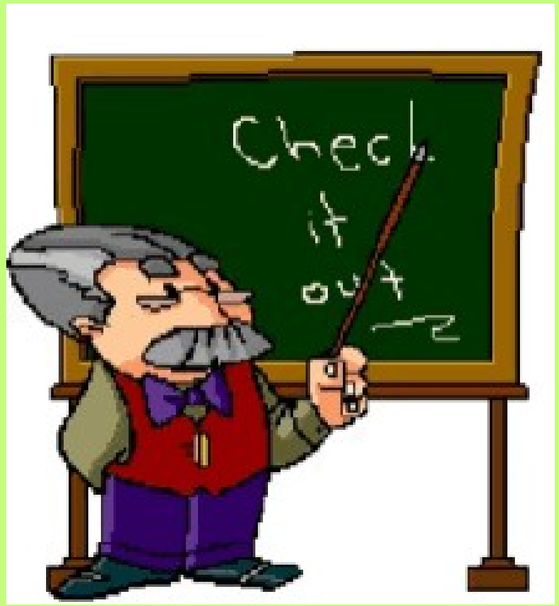


- Высокая стоимость инфраструктуры
- Масштабирование требует значительного времени
- Ограничения пропускной способности
- Сложность организации резервного копирования
- Проблемы с доступностью при сбоях оборудования



Исторический контекст

4



2

3

1

Как это вообще получилось?

01 → 02 → 03

BitTorrent – 2001

Blockchain – 2009

IPFS – 2015

1. Web3 – альтернатива централизованной модели Web2
2. Блокчейн – технологическая основа децентрализации

Такие разные файловые системы

4

2



3

1

Вспомним всех

FAT-семейство (FAT12/16/32/exFAT)

- Нет журналирования
- Сильная фрагментация
- Ограничения на размер файла
- Нет поддержки прав доступа
- Устаревшая структура

NTFS

- Высокая нагрузка на SSD из-за журналирования
- Сложность восстановления при повреждениях
- Лицензионные ограничения
- Избыточные метаданные
- Не лучший выбор для Linux-систем

ext4

- Возможна фрагментация
- Ограничения с огромными файлами
- Менее эффективна для SSD, чем современные ФС
- Нет встроенной поддержки снапшотов
- Медленное удаление большого числа файлов

ZFS / Btrfs

- Высокие требования к RAM
- Сложность администрирования
- Btrfs всё ещё имеет нестабильные функции
- ZFS не встроена в ядро Linux из-за лицензии
- Производительность ниже, чем у ext4 в некоторых задачах

Что такое IPFS

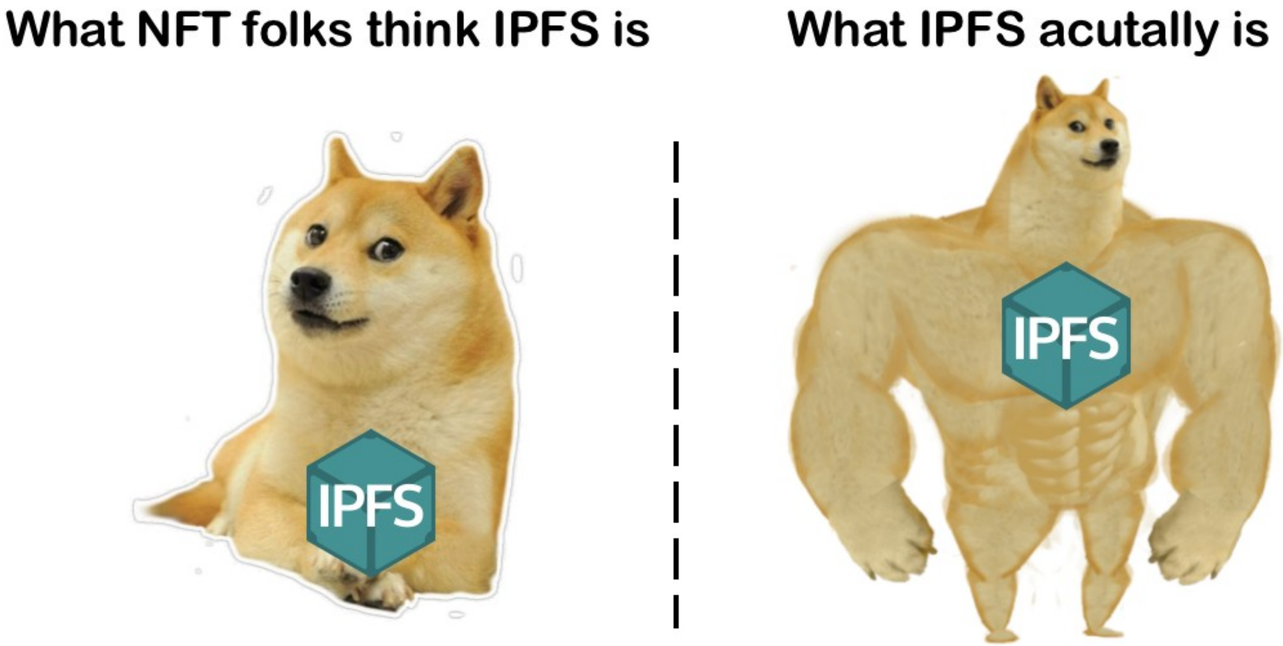
InterPlanetary File System — протокол, разработанный компанией Protocol Labs

Использует принципы:

- DHT (распределённая хеш-таблица)
- DAG (направленный ациклический граф)

Контентная адресация (CID) вместо традиционной адресации по местоположению

Функционирует по аналогии с «вечным BitTorrent + Git», обеспечивая мгновенное версионирование



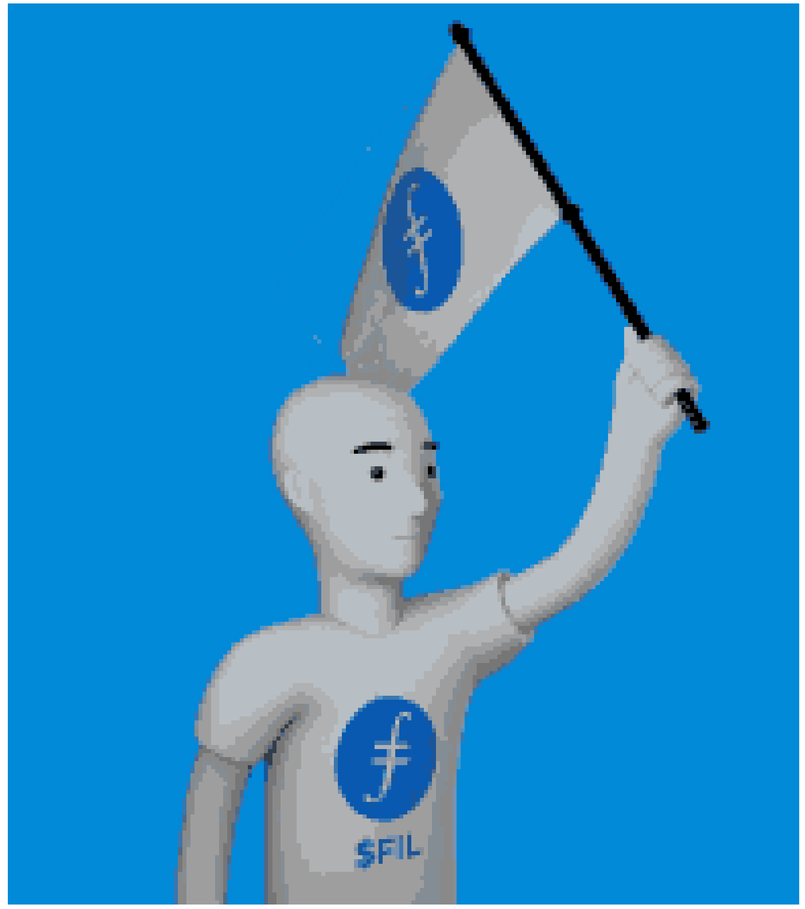
Что такое FileCoin

Filecoin — блокчейн-платформа, ориентированная на децентрализованное хранение данных со встроенными экономическими стимулами для пользователей.

Два алгоритма консенсуса: **Proof of Spacetime (PoSt) и Proof Of Replication (PoRep).**

Майнеры в сети Filecoin отвечают за **хранение и предоставление данных.**

Экономического стимул для хранения данных и других участников Filecoin – **нативная криптовалюта FIL.**

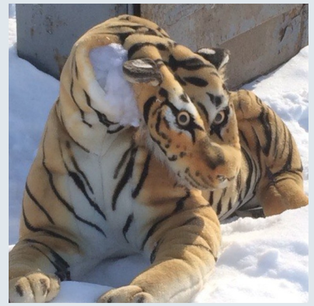


IPFS + FileCoin



- Нет единого узла, данные устойчивы к блокировкам
- Работает даже при выходе из строя части узлов
- Один файл — множество ссылок (экономия места)
- Встроенная история изменений (CID)
- Данные ближе к пользователям (низкие задержки)
- Финансовая мотивация для хранения (FileCoin)

Классические FS



- Единый контрольный узел (можно заблокировать)
- Зависит от серверов (при падении — данные недоступны)
- Одинаковые файлы хранятся многократно
- Требует ручного управления
- Данные ближе к серверам (высокие задержки)
- Нет встроенных стимулов

USER ROLE LEVEL MANAGEMENT

Традиционный подход 🤝 Web3 инновации



🔑 Keycloak

Централизованная

🌐 Web3 Integration

Подпись транзакций через MetaMask /WalletConnect

+

🤖 Роли в смарт-контрактах

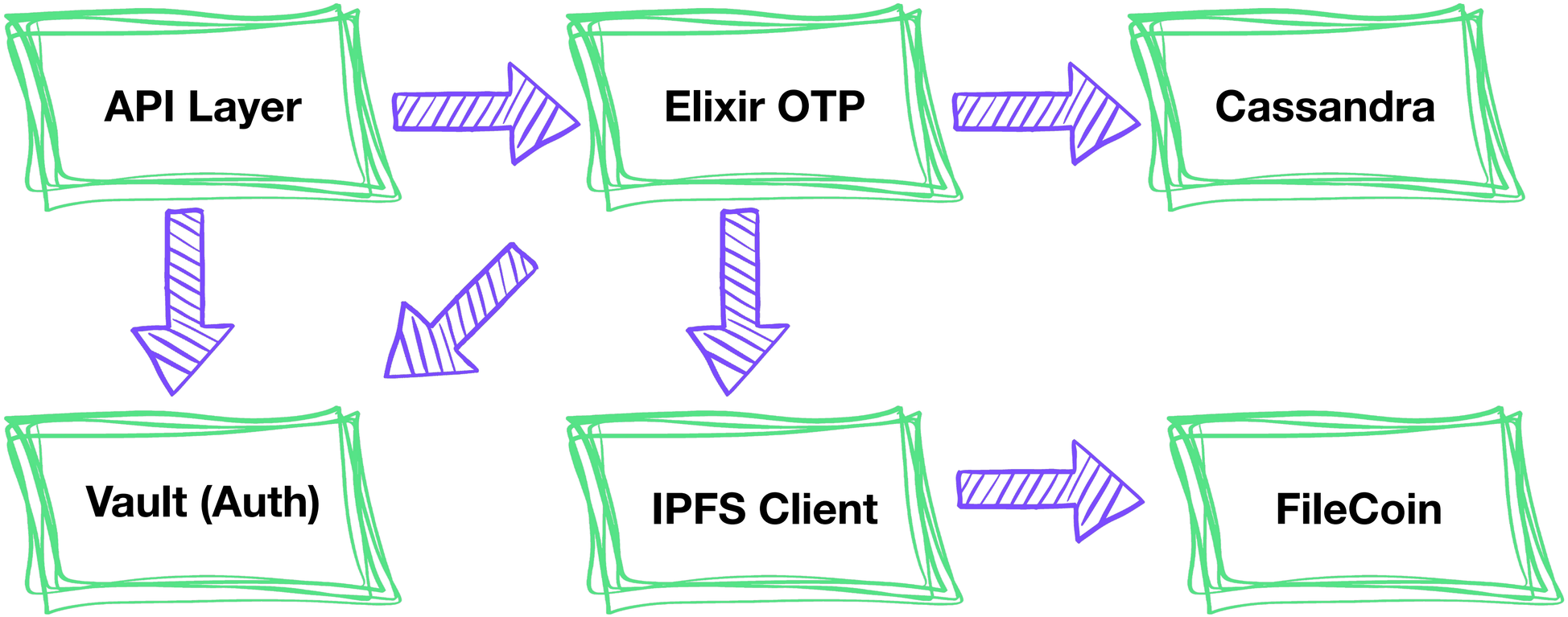
Децентрализованное управление доступом (ERC-20/721)

🕵️‍♂️ ZK-Proofs

Подтверждение прав без раскрытия данных (zk-SNARKs)



А если построить гибрид, как бы он выглядел?



тут бы написать как это круто просто вау, но мы решили оставить это так, для вайба

ELIXIR OTP: СУПЕРВИЗОРЫ КАК ХРАНИТЕЛИ ВРЕМЕНИ

Преимущества Elixir
для распределенных систем

- Отказоустойчивость (самовосстановление)
- Hot Code Reload (без downtime)
- Распределённость (кластеры)
- FP Надёжность (immutable)

```elixir
defmodule Storage.Supervisor do
  use Supervisor

  def start_link(init_arg) do
    Supervisor.start_link(__MODULE__, init_arg, name: __MODULE__)
  end

  def init(_init_arg) do
    children = [
      {StorageWorker, []},
      {IPFS.Client, []},
      {Cassandra.Repo, []}
    ]

    Supervisor.init(children, strategy: :one_for_one)
  end
end
```

Cassandra + IPFS:

Идеальный симбиоз

- **Cassandra**

  Метаданные, поиск и индексация

- **IPFS**

  Хранение больших бинарных объектов (изображение, видео, документы)

```elixir
defmodule MedicalStorage.IPFS do
  def store_image(image_data, encryption_key) do
    # Шифрование изображения перед отправкой в IPFS
    encrypted_data = MedicalStorage.Encryption.encrypt(image_data, encryption_key)
    # Добавление в IPFS и получение CID (Content Identifier)
    {:ok, cid} = IPFS.add(encrypted_data)
    # Запись метаданных в Cassandra
    Cassandra.insert("medical_images", %{
      id: UUID.generate(),
      ipfs_hash: cid,
      timestamp: DateTime.utc_now(),
      # Другие метаданные...
    })
    {:ok, cid}
  end
end
```

Cassandra: Распределенная память временных линий



Линейная масштабируемость — производительность растёт пропорционально добавлению узлов



Отсутствие единой точки отказа (SPOF) — отказ любого узла не влияет на работу системы



Гибкие уровни согласованности — настраиваемый баланс между консистентностью и доступностью



Георазпределённая репликация — автоматическое дублирование данных между дата-центрами



Оптимизированная модель данных — Single Table Design для максимальной производительности

```elixir
# Elixir пример для работы с Cassandra и IPFS
defmodule StorageWorker do
  use GenServer

  def store(data) do
    # Сохраняем в IPFS, получаем хеш
    ipfs_hash = IPFS.add(data)

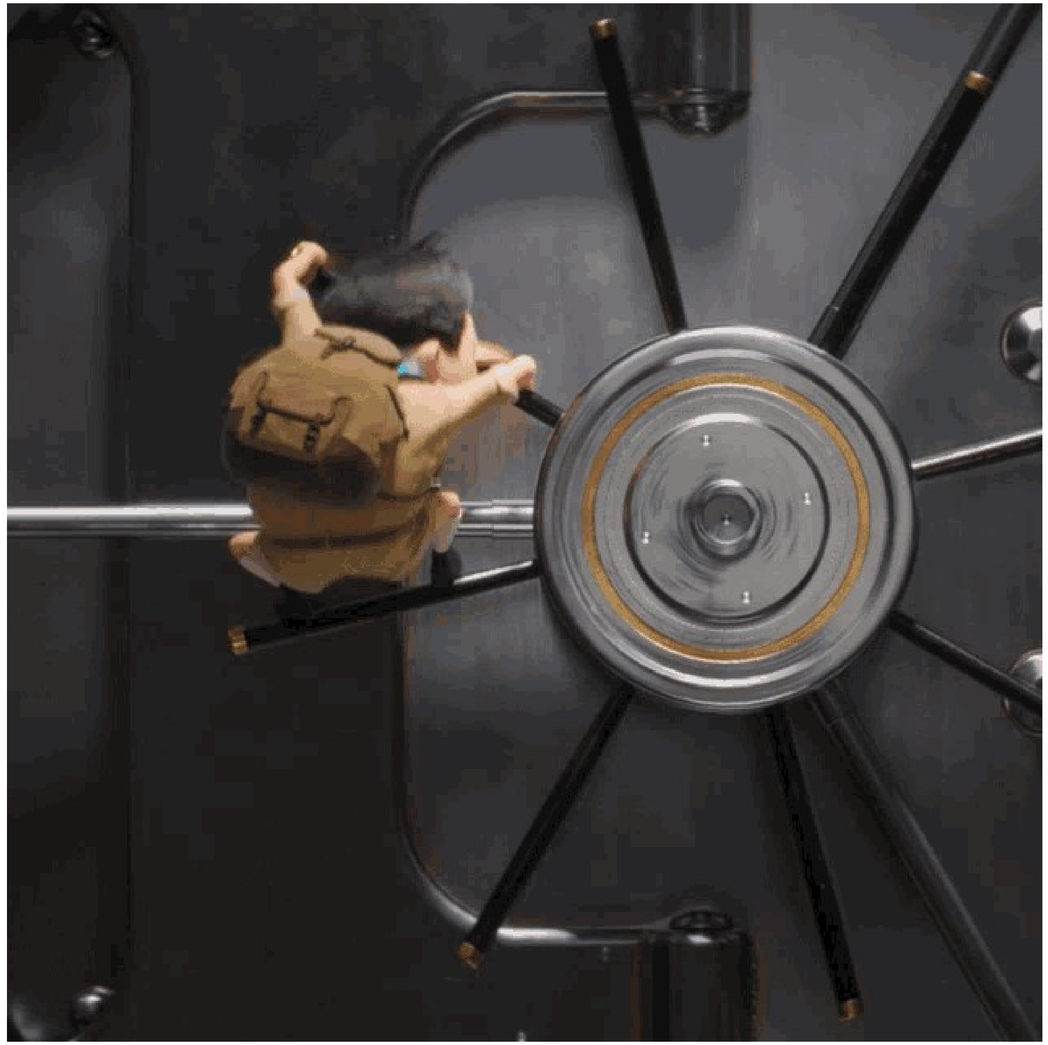
    # Записываем метаданные и хеш в Cassandra
    Cassandra.insert("files", %{
      id: UUID.generate(),
      ipfs_hash: ipfs_hash,
      timestamp: DateTime.utc_now()
    })

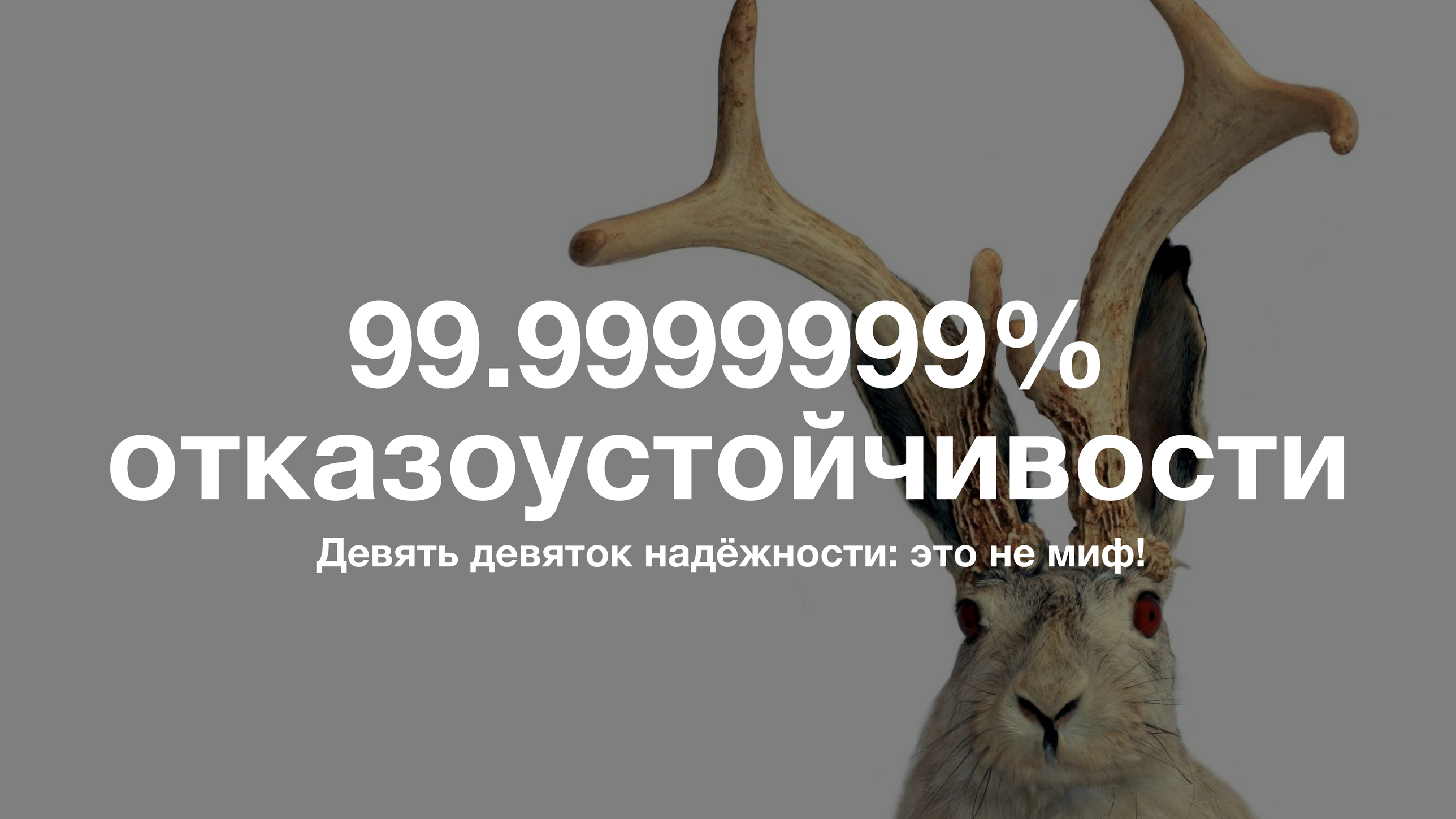
    ipfs_hash
  end
end
```

HashiCorp Vault

Защита конфиденциальных данных

- Хранение ключей → HSM, KMS
- Политики доступа → RBAC, ABAC
- Ротация → Автообновление + аудит
- Аутентификация → OAuth 2.0, JWT





99.9999999% отказоустойчивости

Девять девяток надёжности: это не миф!

99.9999999% отказоустойчивости

1. Множественная репликация данных в Cassandra (N+2)
2. Распределённое хранение файлов в IPFS
3. Самовосстановление сервисов через Elixir OTP супервизоры
4. Фиксация операций в неизменяемом блокчейн-журнале
5. Асинхронное восстановление данных после сбоев



Ну вроде бы и балдеж



~~Ну это был балдеж~~ Это не балдеж!



В чем же проблема?

ПРОБЛЕМЫ IPFS



СЛОЖНО. Сложность масштабирования для больших объёмов данных



ТРЕБОВАТЕЛЬНО. Высокие требования к пропускной способности сети



МЕДЛЕННО. Низкая скорость доступа к редко запрашиваемым данным



РИСКОВАННО. Риск потери данных без принудительного пиннинга



ОГРАНИЧЕННО. Ограниченная эффективность для частых обновлений

ОГРАНИЧЕНИЯ FILECOIN



СЛОЖНО. Сложная система доказательств



ТРЕБОВАТЕЛЬНО. Высокие требования к инфраструктуре



МЕДЛЕННО. Долгий процесс поиска



ЗАВИСИМО. Зависимость от волатильности токена



ПРОБЛЕМНО. Проблемы восстановления данных

Web3-хранилища: нерешенные задачи

Масштабируемость

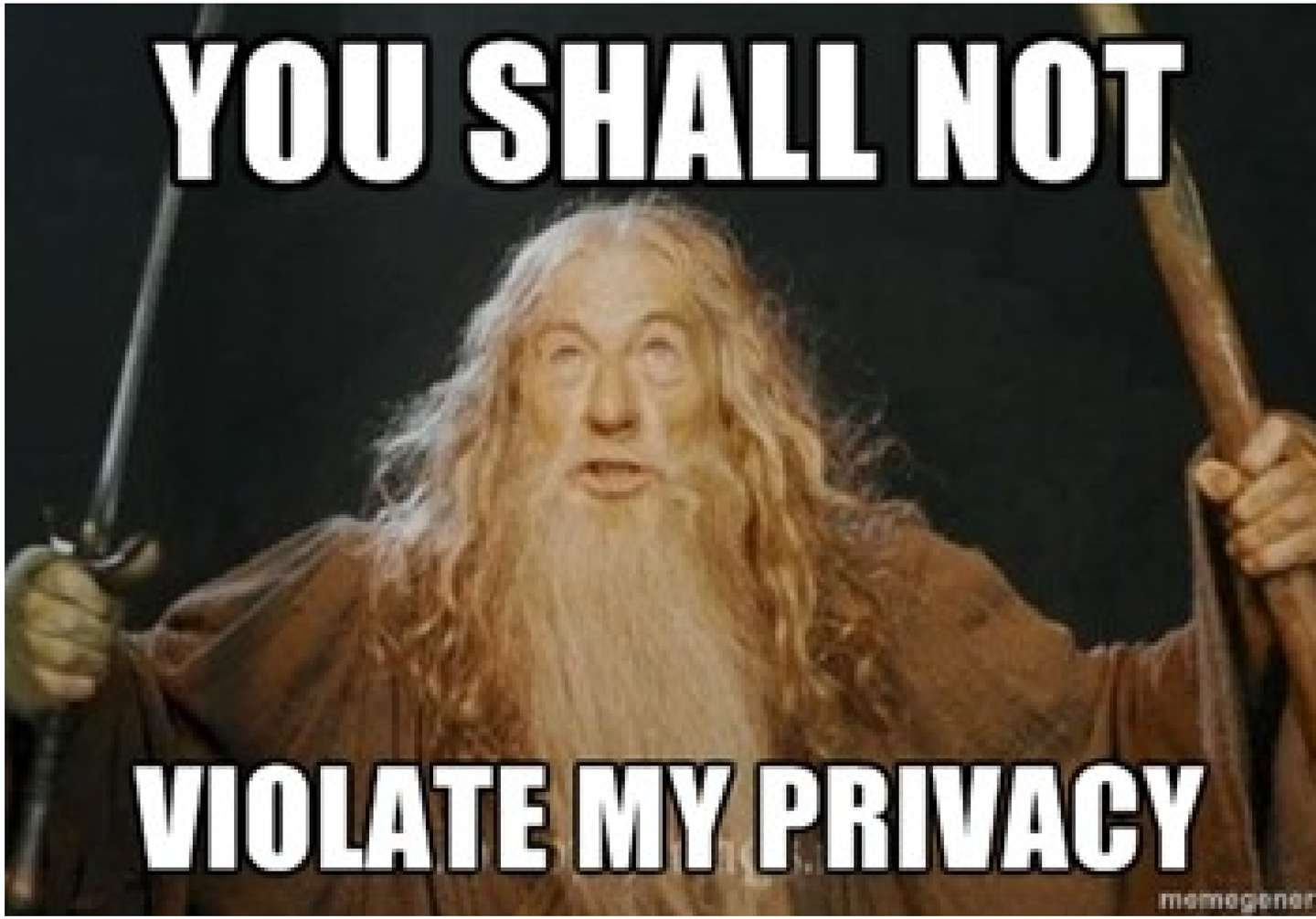
Поддержка больших объемов транзакций и данных



Web3-хранилища: нерешенные задачи

Регуляторная совместимость

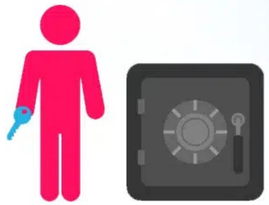
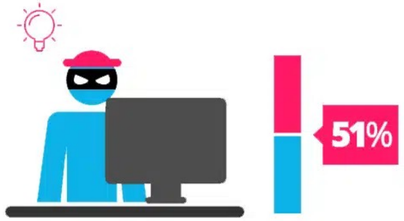
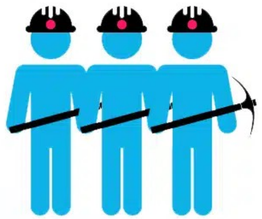
Соответствие GDPR и праву на забвение



Web3-хранилища: нерешенные задачи

Энергоэффективность

Оптимизация механизмов консенсуса

Proof of Work	vs.	Proof of Stake
		
To add each block to the chain, miners must compete to solve a difficult puzzle using their computers processing power.		There is no competition as the block creator is chosen by an algorithm based on the user's stake.
		
In order to add a malicious block, you'd have to have a computer more powerful than 51% of the network.		In order to add a malicious block, you'd have to own 51% of all the cryptocurrency on the network.
		
The first miner to solve the puzzle is given a reward for their work.		There is no reward for making a block, so the block creator takes a transaction fee.

Web3-хранилища: нерешенные задачи

Оптимальный баланс

Децентрализация vs. скорость доступа к данным



Web3-хранилища: нерешенные задачи

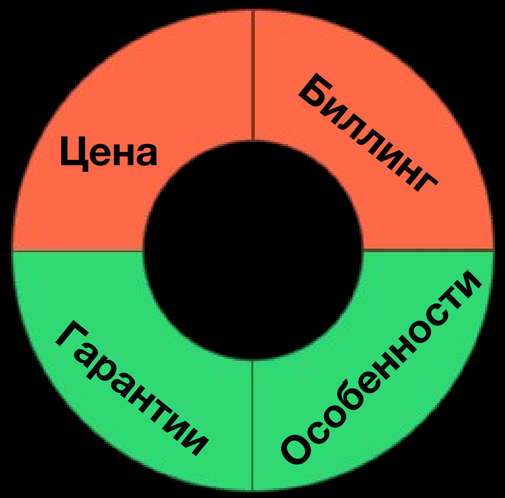
Восстановление доступа

Механизмы recovery
при утере ключей



Сравнить бы это…

Amazon S3



IPFS + Pinning Service



Filecoin



Проблемы и выгоды

	Плюсы	Минусы	Проблемы	Пример
Клиент-сервер	Простота, чёткое разделение ролей	Сервер — единая точка отказа, сложность масштабирования	Файлы на сервере — риск перегрузки	• Google, Wikipedia) • Сбербанк Online • Gmail
Микросервисы	Гибкость, независимое масштабирование	Сложность управления, eventual consistency, дублирование данных	Проблемы с транзакциями и согласованностью данных	• Netflix • Uber • Spotify • Airbnb
Peer-to-Peer (P2P)	Децентрализация, отказоустойчивость	Низкая скорость, сложность модерации	Риск атак, дублирование данных	• µTorrent, Transmission • Bitcoin • Skype (ранние версии)
Гибридная	Гибкость, адаптивность под задачи	Риск "распределённого монолита", сложность администрирования	Проблемы синхронизации между разными хранилищами	• Dropbox (клиент-сервер + P2P-синхронизация) • Discord (серверы + P2P-звонки)

А как этим всем управлять, мониторить и сопровождать?

```elixir
defmodule MedicalStorage.Supervisor do
  use Supervisor

  @moduledoc """
  Супервизор для управления компонентами медицинского хранилища.
  Запускает и контролирует все основные сервисы системы.
  """

  def start_link(init_arg) do
    Supervisor.start_link(__MODULE__, init_arg, name: __MODULE__)
  end
```

```elixir
  @impl true
  def init(_init_arg) do
    children = [
      # Компонент для управления кэшем изображений
      {MedicalStorage.Cache, []},

      # Клиент Cassandra для метаданных
      {MedicalStorage.Cassandra, []},

      # Клиент IPFS для долговременного хранения
      {MedicalStorage.IPFS, []},

      # Модуль шифрования данных
      {MedicalStorage.Encryption, []},

      # API для интеграции с внешними системами
      {MedicalStorage.API, []}
    ]

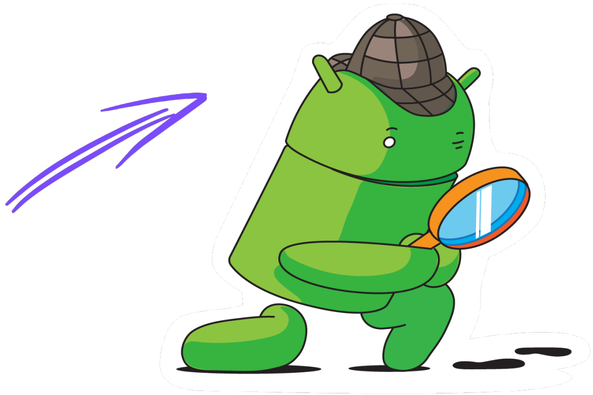
    # Стратегия перезапуска: если падает один процесс,
    # перезапускается только он, остальные продолжают работать
    Supervisor.init(children, strategy: :one_for_one)
  end
end
```

Приземление на реальность: путь к адаптации



Постепенное внедрение

- Гибридная архитектура
- Тестирование в dev-среде



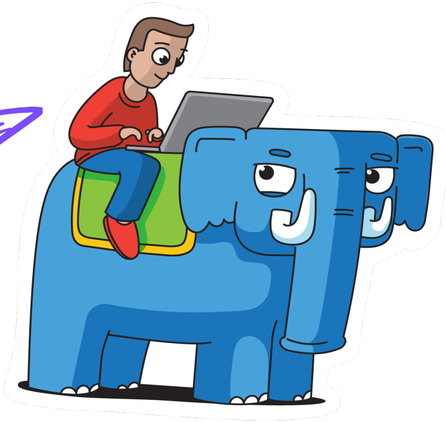
Стратегия развёртывания

- IPFS как репликация
- Внедри через архивы и бэкапы



Гарантии доступности

- Пиннинг критичных данных
- Контроль репликации



Архитектурные требования

- Учитывай задержки
- Оптимизируй кэширование

Марат Зимнуров

Руководитель группы разработки





/the-homeless-god



@the_homeless_god

Идеальной системы нет, решайте задачи бизнеса ресурсами, которые предоставляет вам бизнес

Потому что это всё может быть выброшено в помойку через год

Потому что это всё может быть не нужно через год

Потому что регуляторные требования могут меняться раз от разу и это может стоить переделки всей архитектуры

