точка банк

КЕШИРОВАНИЕ ДАННЫХ НА МАКСИМАЛКАХ

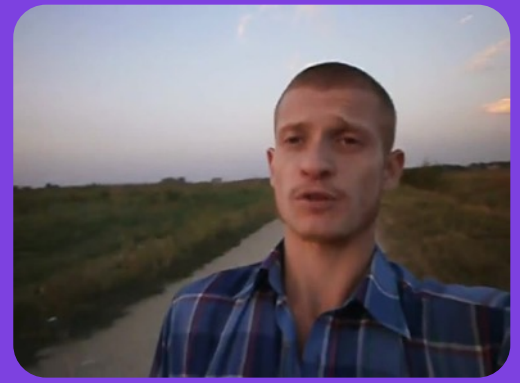
Как построить кеш для энтерпрайза

Даниил Рублёв

И швец, и жнец, и на дуде игрец

точка банк

КЕШИРОВАНИЕ НА ФРОНТЕ

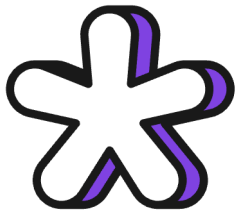


ЧТО НАМ СТОИТ КЕШ ПОСТРОИТЬ

точка банк

ЧТО БУДЕТ, ЧЕГО НЕ БУДЕТ

точка банк

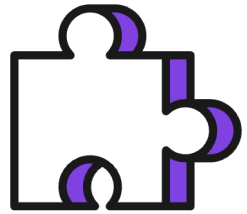


Подводные камни

И как о них споткнуться

Решения и альтернативы

Что мы не выбрали,
но мы не вы



Из чего состоит система кеширования

C4 и бонус

МЕТРИКИ

точка банк

Сначала анализ, а потом разработка

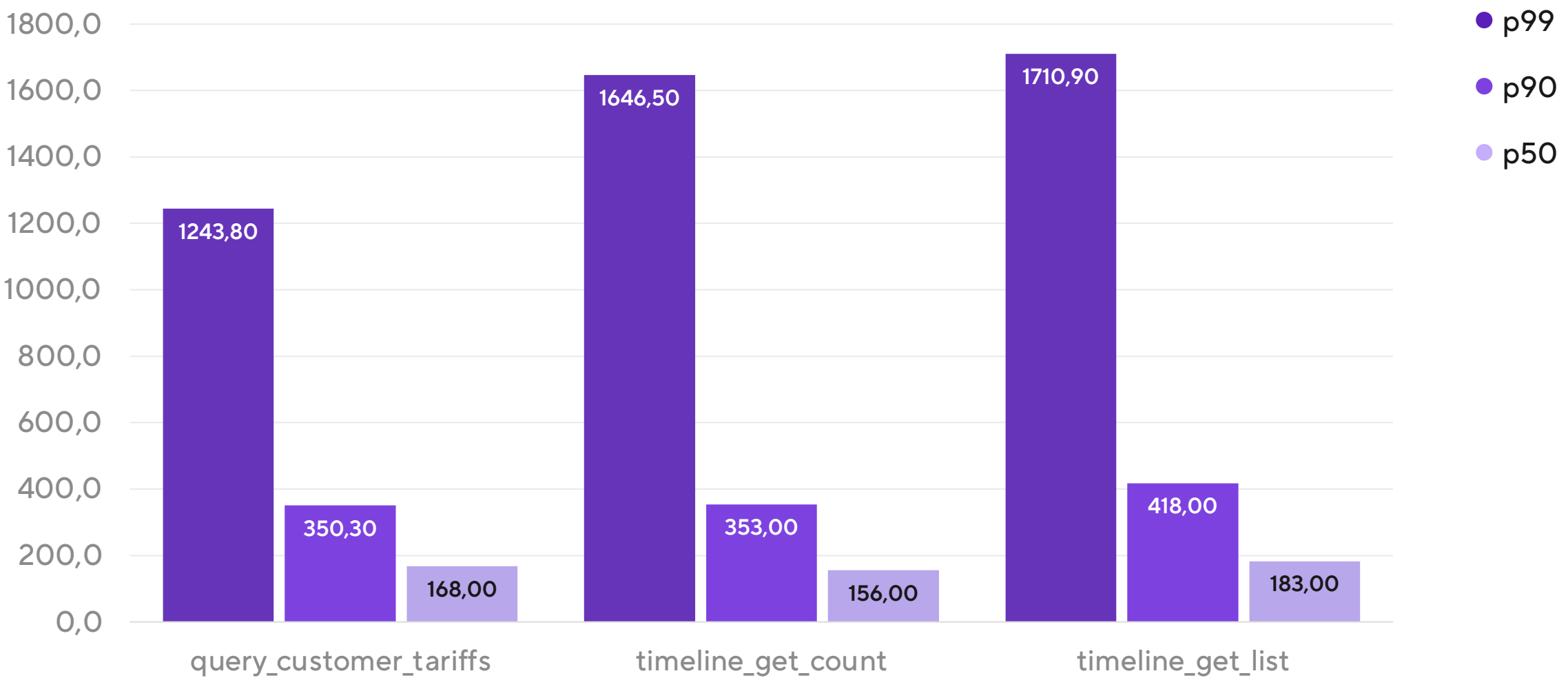
МЕТРИКИ

точка банк

Сначала анализ, а потом разработка

КАК БЫЛО

точка банк



РАЗБИРАЕМ КЕШ НА ЧАСТИ

точка банк

РАЗБИРАЕМ КЕШ НА ЧАСТИ

точка банк

```
// service-worker.js

self.addEventListener('fetch', (event) => {
  if (storage.has(event.request.url)) {
    event.respondWith(storage.get(event.request.url));
  } else {
    event.respondWith(fetch(event.request).then(response => {
      lru.evictIfNeeded();
      storage.set(event.request.url, response.clone());

      return response;
    }));
  }
});
```

РАЗБИРАЕМ КЕШ НА ЧАСТИ

точка банк

```
// service-worker.js

self.addEventListener('fetch', (event) => {
  if (storage.has(event.request.url)) {
    event.respondWith(storage.get(event.request.url));
  } else {
    event.respondWith(fetch(event.request).then(response => {
      lru.evictIfNeeded();
      storage.set(event.request.url, response.clone());

      return response;
    }));
  }
});
```

РАЗБИРАЕМ КЕШ НА ЧАСТИ

точка банк

```
// service-worker.js

self.addEventListener('fetch', (event) => {
  if (storage.has(event.request.url)) {
    event.respondWith(storage.get(event.request.url));
  } else {
    event.respondWith(fetch(event.request).then(response => {
      lru.evictIfNeeded();
      storage.set(event.request.url, response.clone());

      return response;
    }));
  }
});
```

РАЗБИРАЕМ КЕШ НА ЧАСТИ

точка банк

```
// service-worker.js

self.addEventListener('fetch', (event) => {
  if (storage.has(event.request.url)) {
    event.respondWith(storage.get(event.request.url));
  } else {
    event.respondWith(fetch(event.request).then(response => {
      lru.evictIfNeeded();
      storage.set(event.request.url, response.clone());

      return response;
    }));
  }
});
```

КТО? ЧТО? КАК?

точка банк

- Кто отвечает за кеш?
- Что за данные он собирается кешировать?
- Как он будет за ними ходить?

точка банк

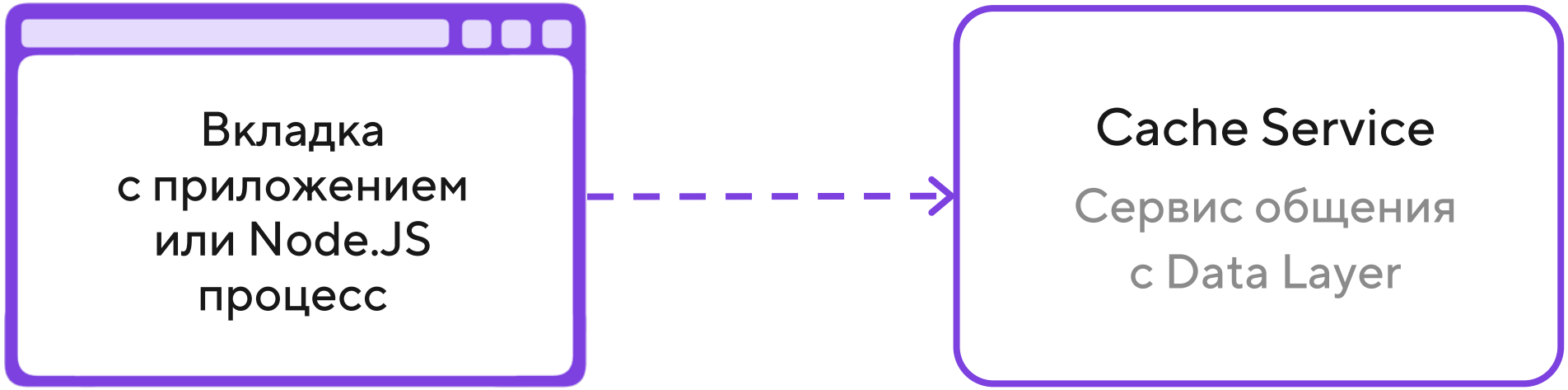
ВЗЯТЬ ДАННЫЕ ИЗ КЕША ДОЛЖНО БЫТЬ БЫСТРЕЕ, ЧЕМ СХОДИТЬ ЗА НИМИ НА БЭК

КТО?

точка банк

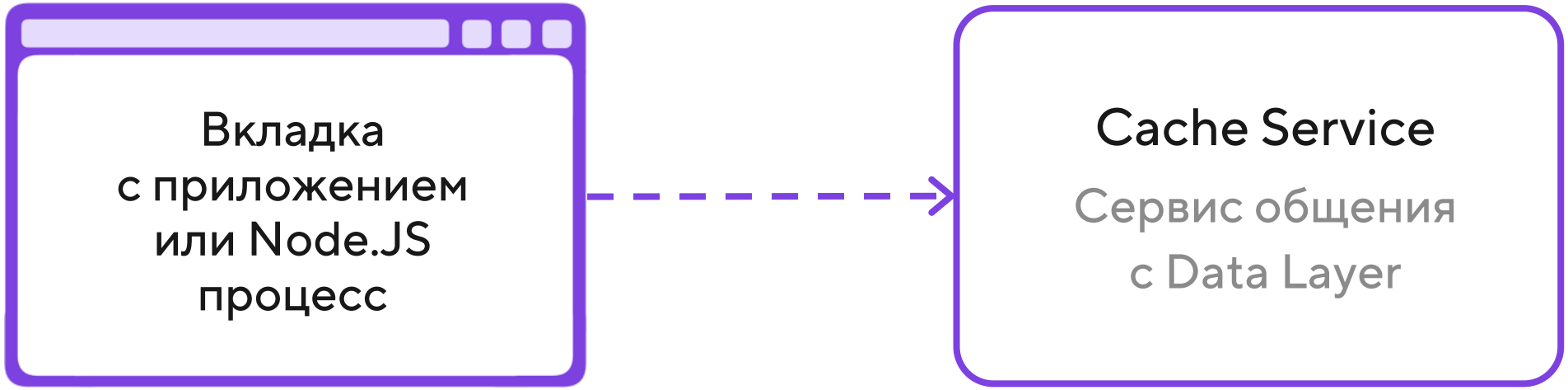
КТО?

точка банк



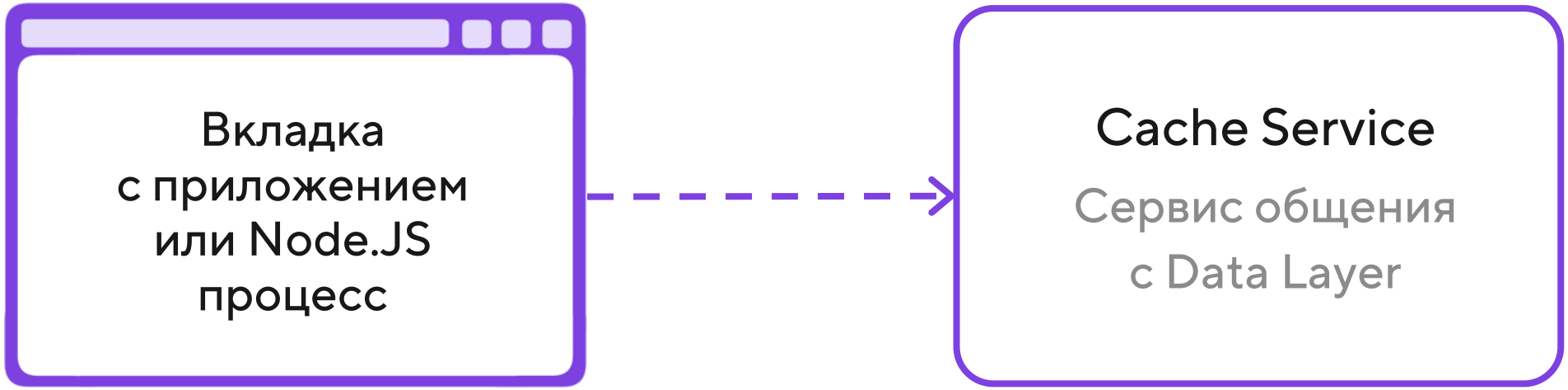
КТО?

точка банк



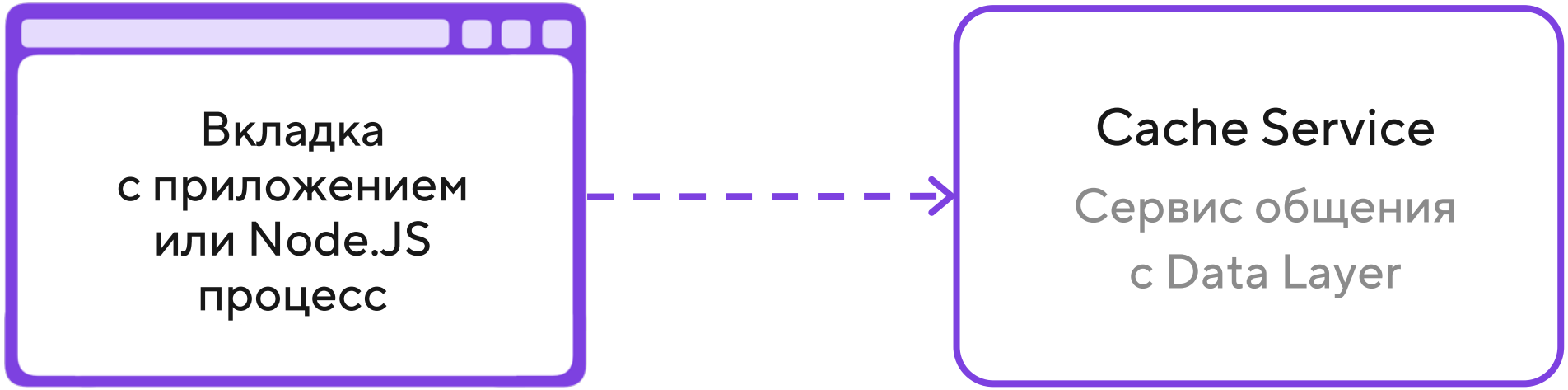
КТО?

точка банк



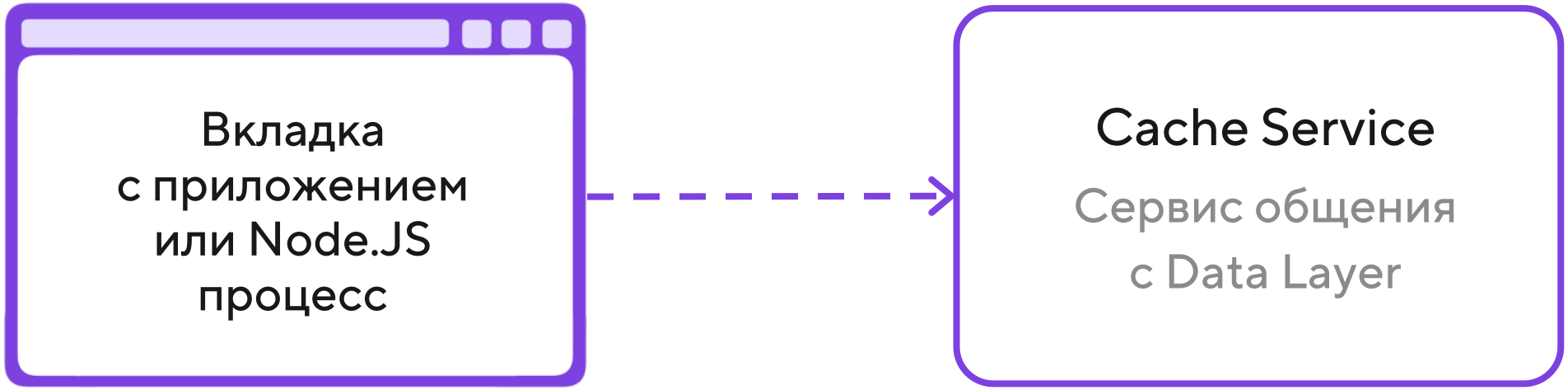
КТО?

точка банк



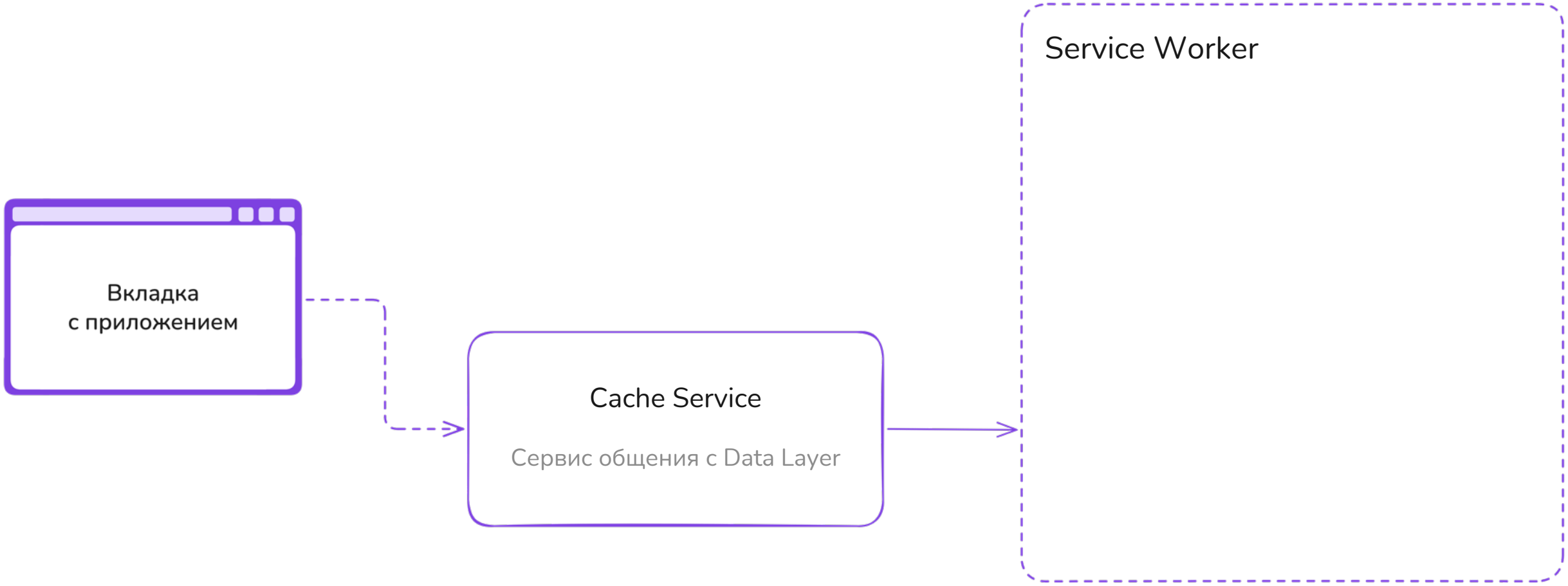
КТО?

точка банк



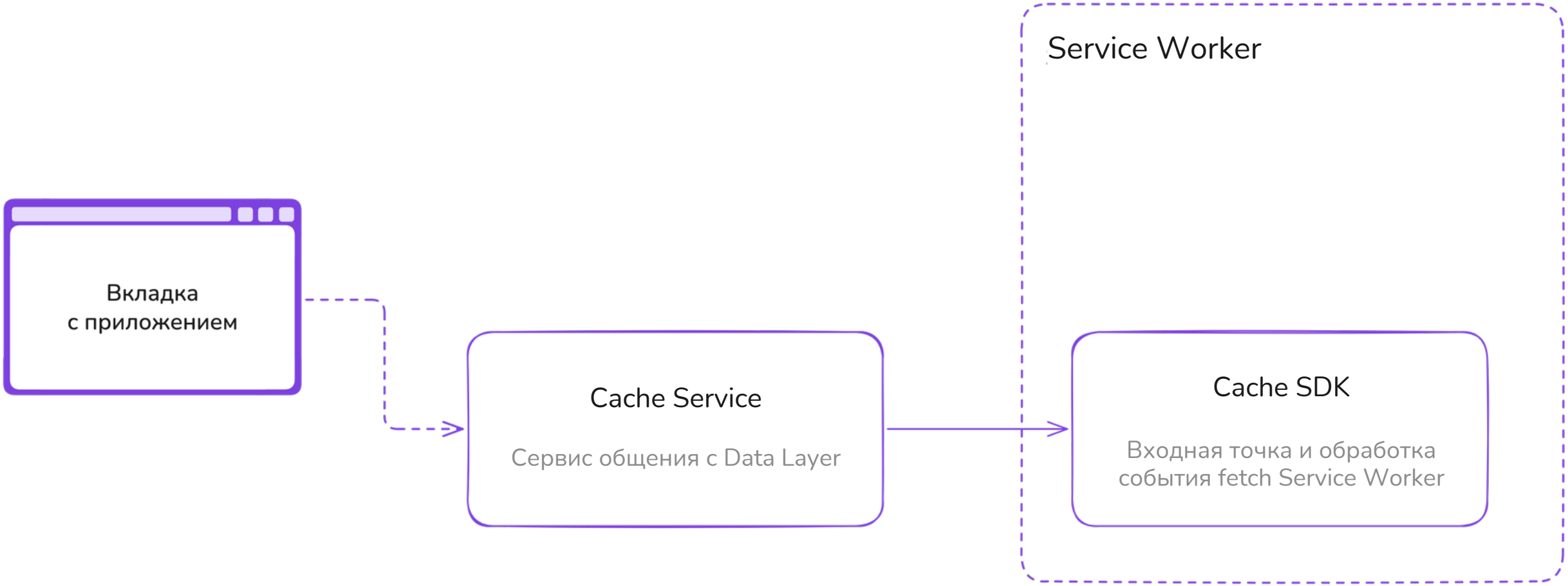
КТО?

точка банк



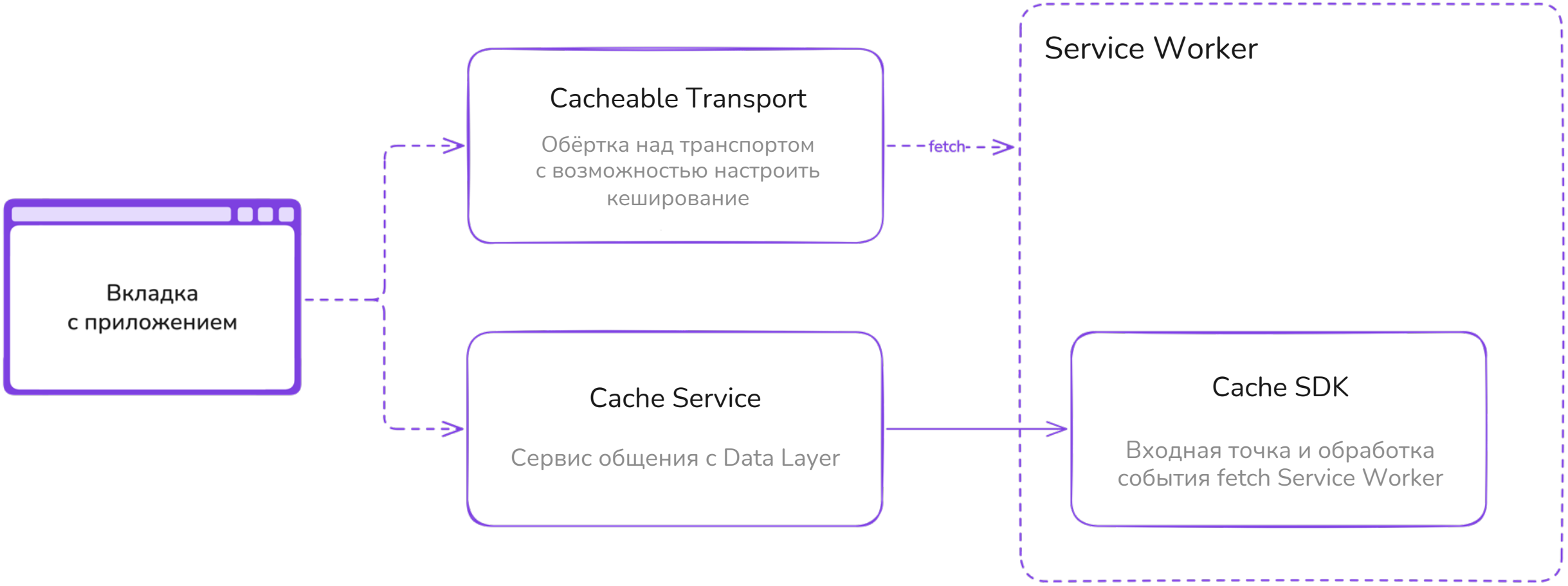
КТО?

точка банк



КТО?

точка банк

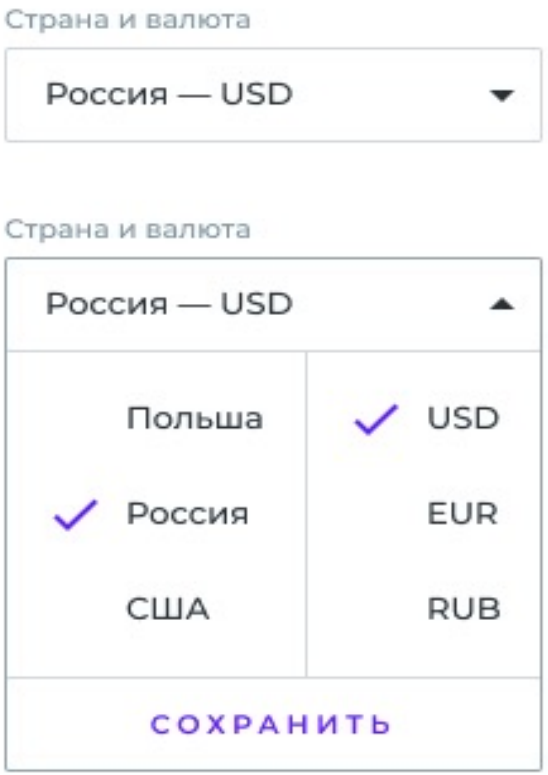


ЧТО?

точка банк

ЧТО?

точка банк



ЧТО?

точка банк

Страна и валюта

Россия — USD

Страна и валюта

Россия — USD

Польша	✓ USD
✓ Россия	EUR
США	RUB

СОХРАНИТЬ

ЧТО?

точка банк

Страна и валюта

Россия — USD

Страна и валюта

Россия — USD

Польша	✓ USD
✓ Россия	EUR
США	RUB

СОХРАНИТЬ

ЧТО?

точка банк

Какими должны быть данные:

ЧТО?

Какими должны быть данные:

Меняются не моментально

Не должно быть полей, которые могут меняться на каждый запрос, по типу цены или времени

ЧТО?

точка банк

Какими должны быть данные:

Меняются не моментально

Не должно быть полей, которые могут меняться на каждый запрос, по типу цены или времени

Нужными

Нет смысла кешировать то, к чему потом не обращаются

ЧТО?

Какими должны быть данные:

Меняются не моментально

Не должно быть полей, которые могут меняться на каждый запрос, по типу цены или времени

Нужными

Нет смысла кешировать то, к чему потом не обращаются

Без фильтров

Если данные запрашиваются только с фильтрами, будет много промахов по кешу, либо он будет забит

ГДЕ?

точка банк

ГДЕ?

точка банк

1

IndexedDB

Для большинства

ГДЕ?

точка банк

1

IndexedDB

Для большинства

2

LocalStorage

Просто, но мало

ГДЕ?

точка банк

1. **IndexedDB**
   Для большинства
2. **LocalStorage**
   Просто, но мало
3. **In-memory**
   Быстро, но тоже мало

ГДЕ?

точка банк

1

IndexedDB

Для большинства

2

LocalStorage

Просто, но мало

3

In-memory

Быстро, но тоже мало

4

Cache Storage API

«Из коробки»,
но не гибко

ГДЕ?

точка банк

1

IndexedDB

Для большинства

2

LocalStorage

Просто, но мало

3

In-memory

Быстро, но тоже мало

4

Cache Storage API

«Из коробки»,
но не гибко

5

OPFS/SQLite

Для меньшинства

ГДЕ?

точка банк

1

IndexedDB

Для большинства

2

LocalStorage

Просто, но мало

3

In-memory

Быстро, но тоже мало

4

Cache Storage API

«Из коробки»,
но не гибко

5

OPFS/SQLite

Для меньшинства

6

BFF + key/value

Если слабые
клиенты

ГДЕ?

точка банк

1

IndexedDB

Для большинства

2

LocalStorage

Просто, но мало

3

In-memory

Быстро, но тоже мало

4

Cache Storage API

«Из коробки»,
но не гибко

5

OPFS/SQLite

Для меньшинства

6

BFF + key/value

Если слабые
клиенты

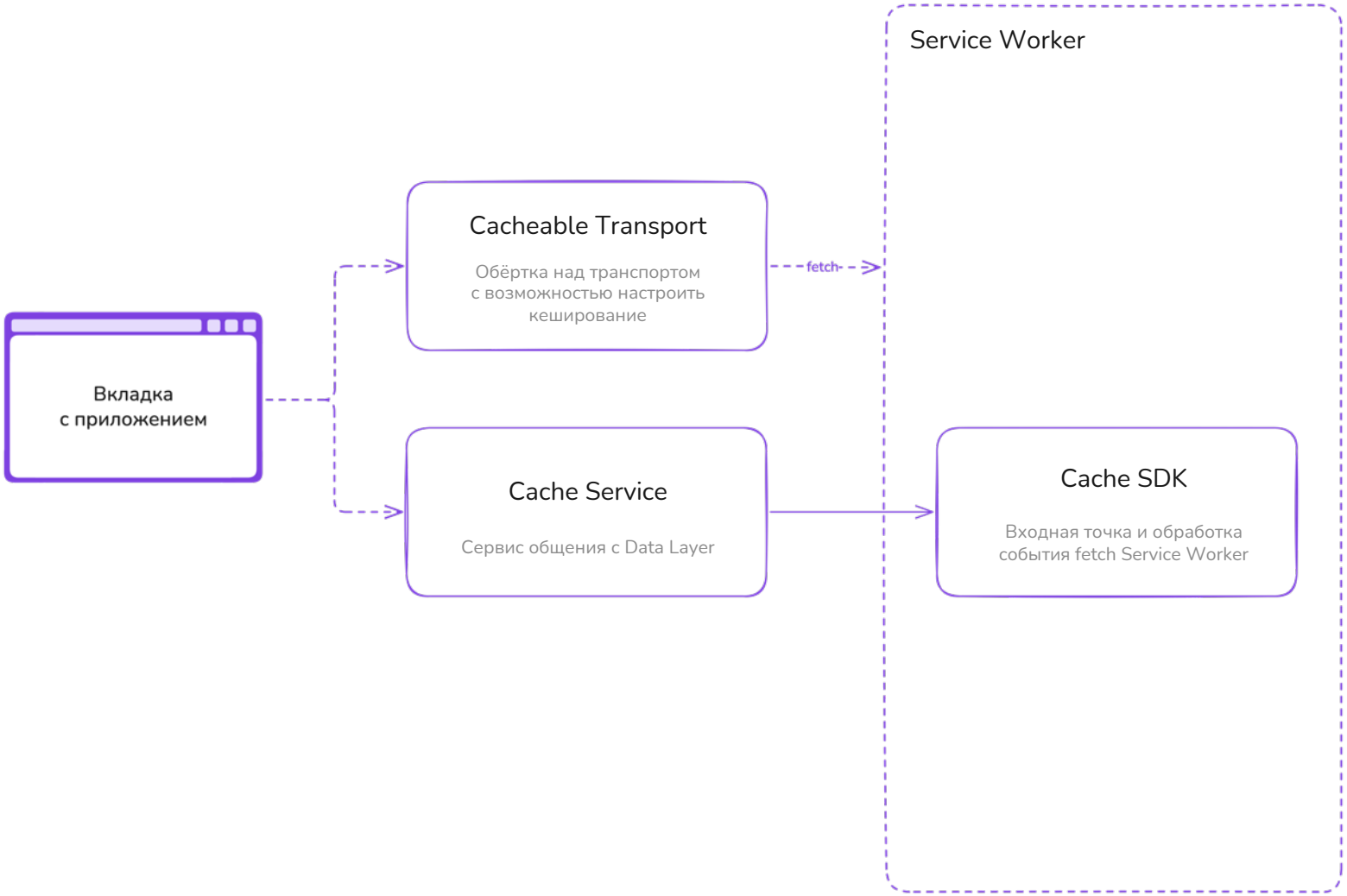
7

Bridge

Круто для WebView

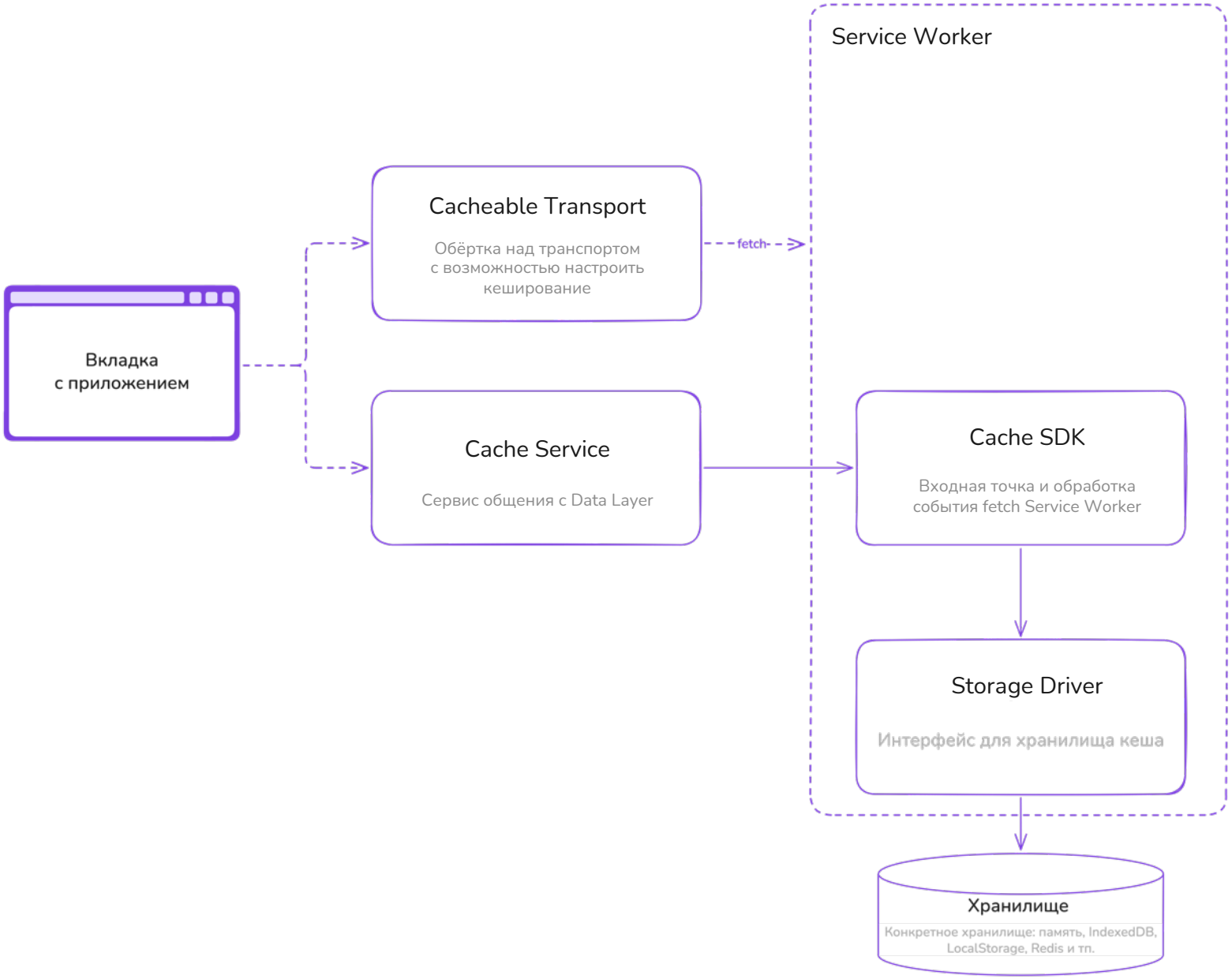
ГДЕ?

точка банк



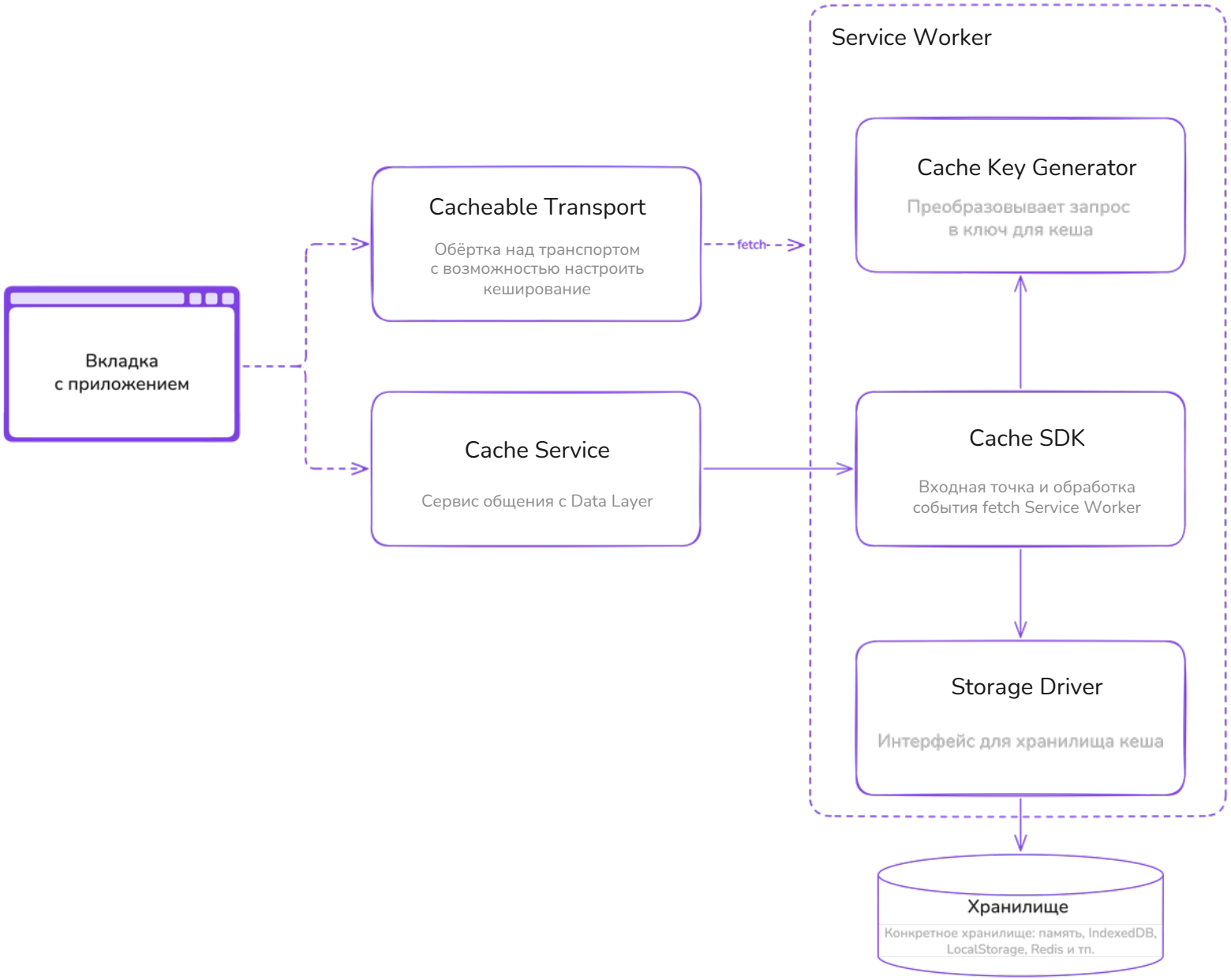
ГДЕ?

точка банк



ГДЕ?

точка банк



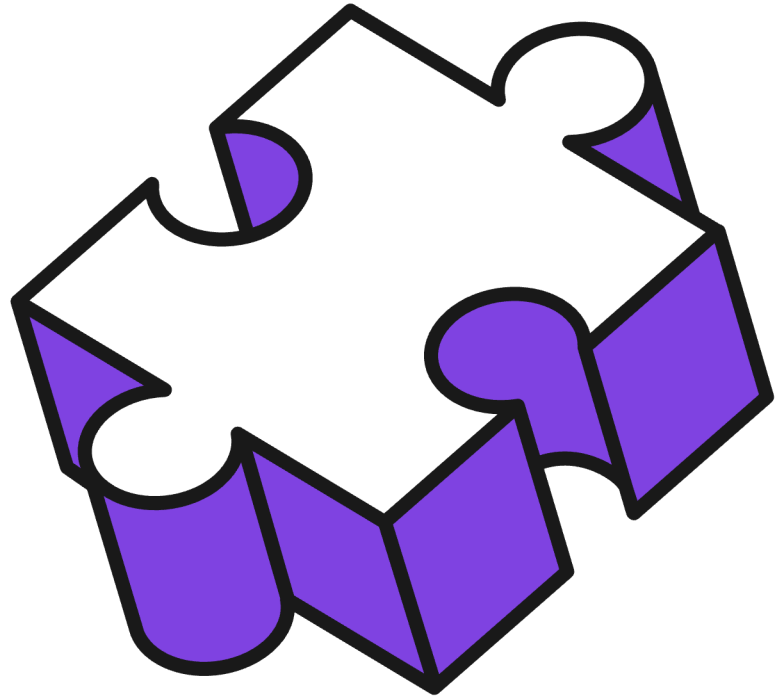
ЛИМИТЫ

точка банк

Учимся удалять записи

МЕТОДЫ ВЫТЕСНЕНИЯ КЕША

Это методы управления устаревшими данными в кеше. Они заключаются в их удалении или обновлении, когда исходные данные в базе или системе изменились. Основная цель — обеспечить согласованность (свежесть) данных: удалять старые записи, чтобы пользователи не видели неактуальную информацию.



МЕТОДЫ ВЫТЕСНЕНИЯ КЕША

точка банк

LRU / LFU

Удаляют наиболее редко или часто используемый элемент

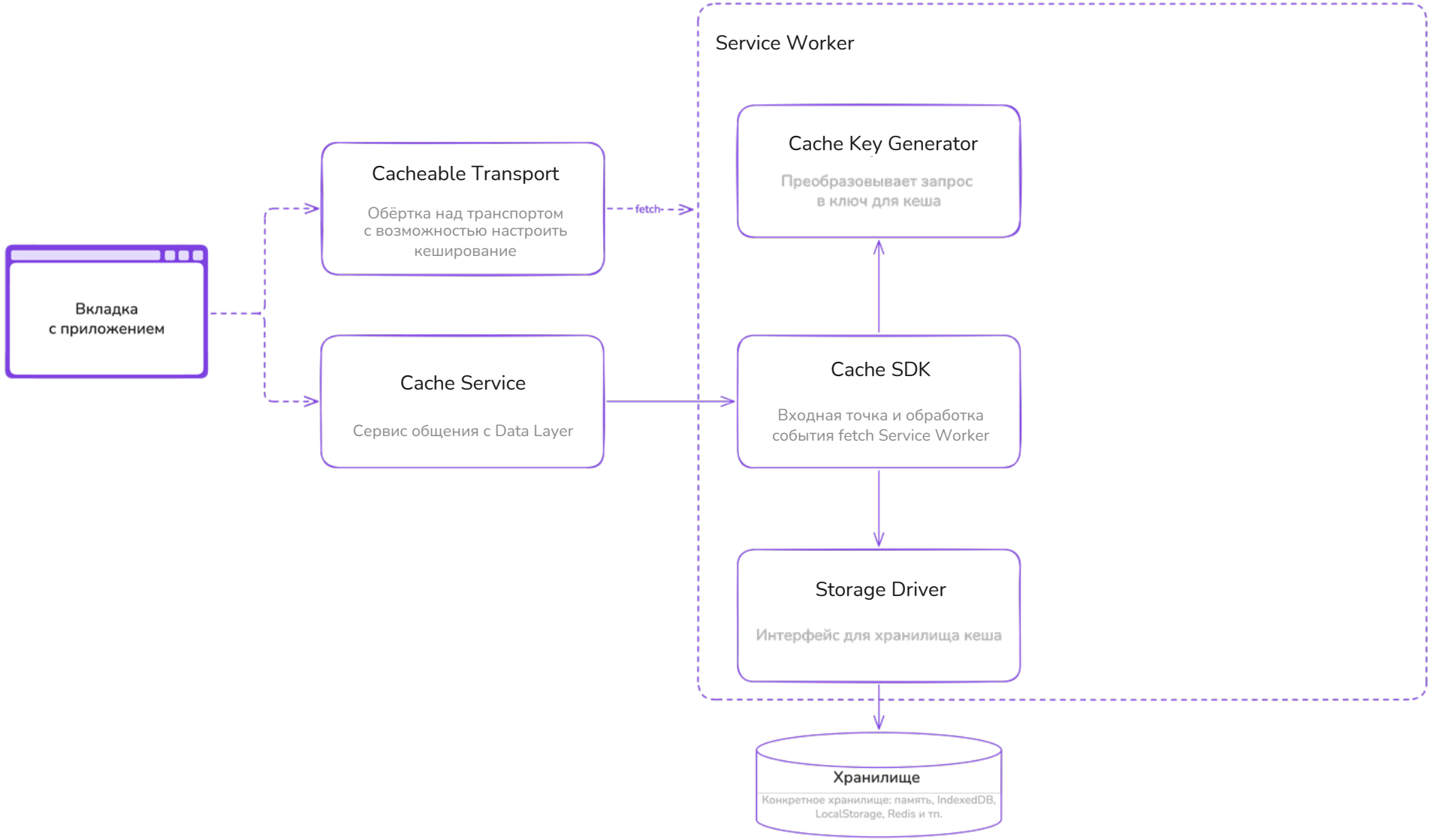
FIFO / LIFO

Удаляют первый или последний добавленный элемент

MRU, RR, 2Q и другие

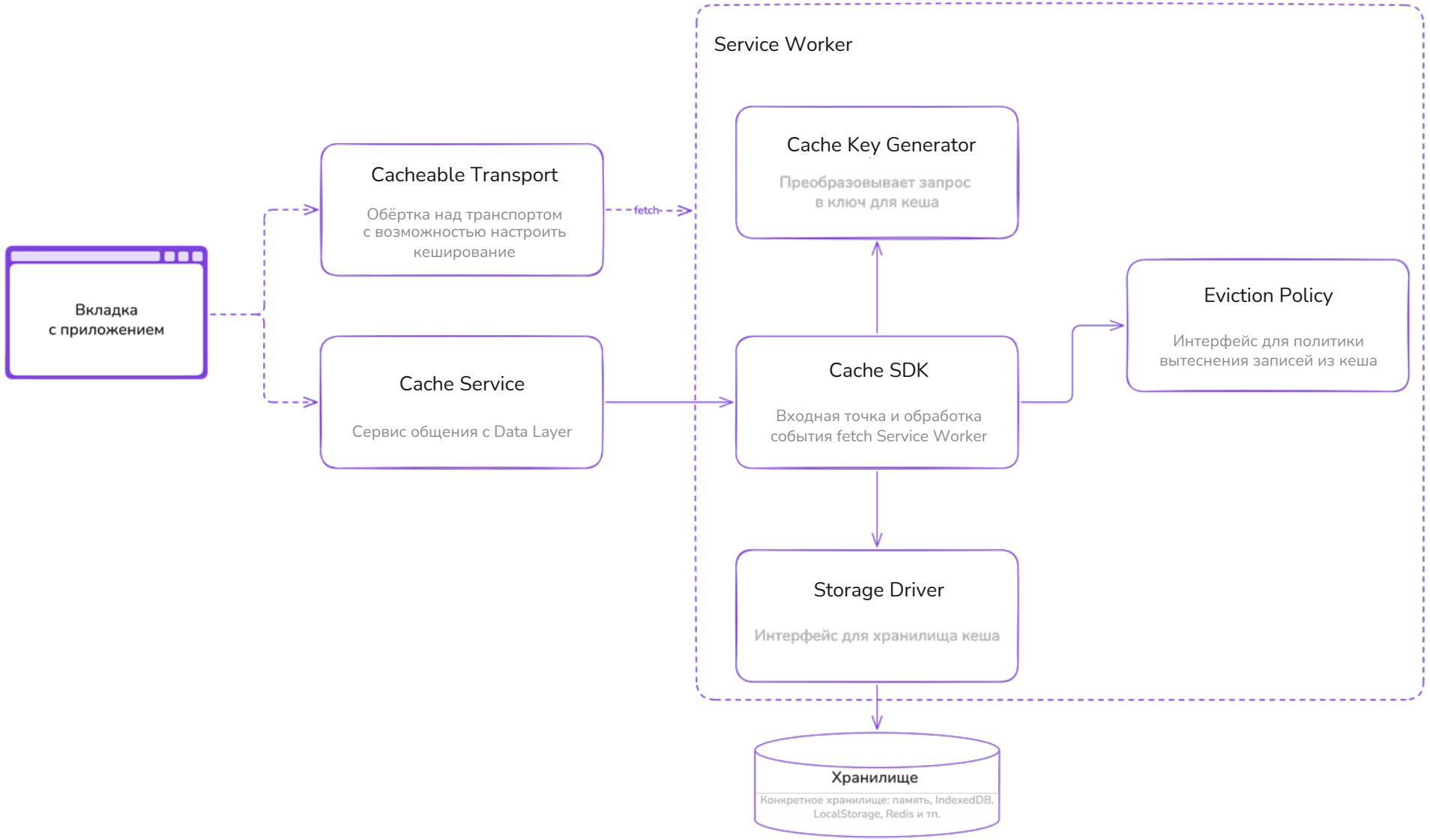
ВЫТЕСНЯЕМ КЕШ

точка банк



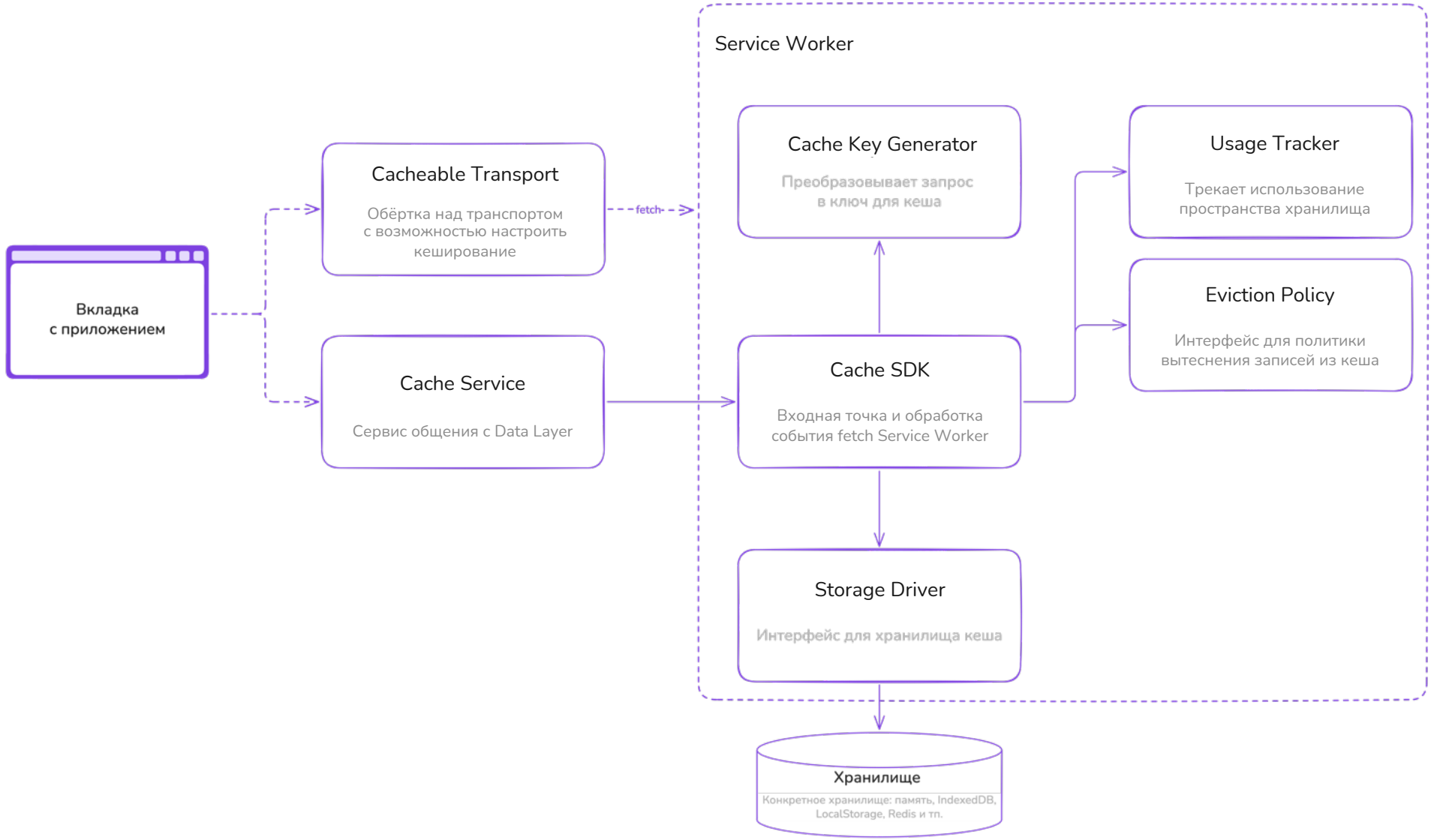
ВЫТЕСНЯЕМ КЕШ

точка банк



ВЫТЕСНЯЕМ КЕШ

точка банк



КОГДА УДАЛЯТЬ?

точка банк

КОГДА УДАЛЯТЬ?

точка банк

В простоях

Удаляем в фоне, когда не заняты

КОГДА УДАЛЯТЬ?

точка банк

В простоях

Удаляем в фоне, когда не заняты

Не по одной записи, а пачками

Особенно профитно, если хранилище транзакционное

ЕСТЬ ПАРОЧКА НО

точка банк

Или когда наступает то самое «пу-пу-пуууууу»

№ 152-ФЗ «О ПЕРСОНАЛЬНЫХ ДАННЫХ»

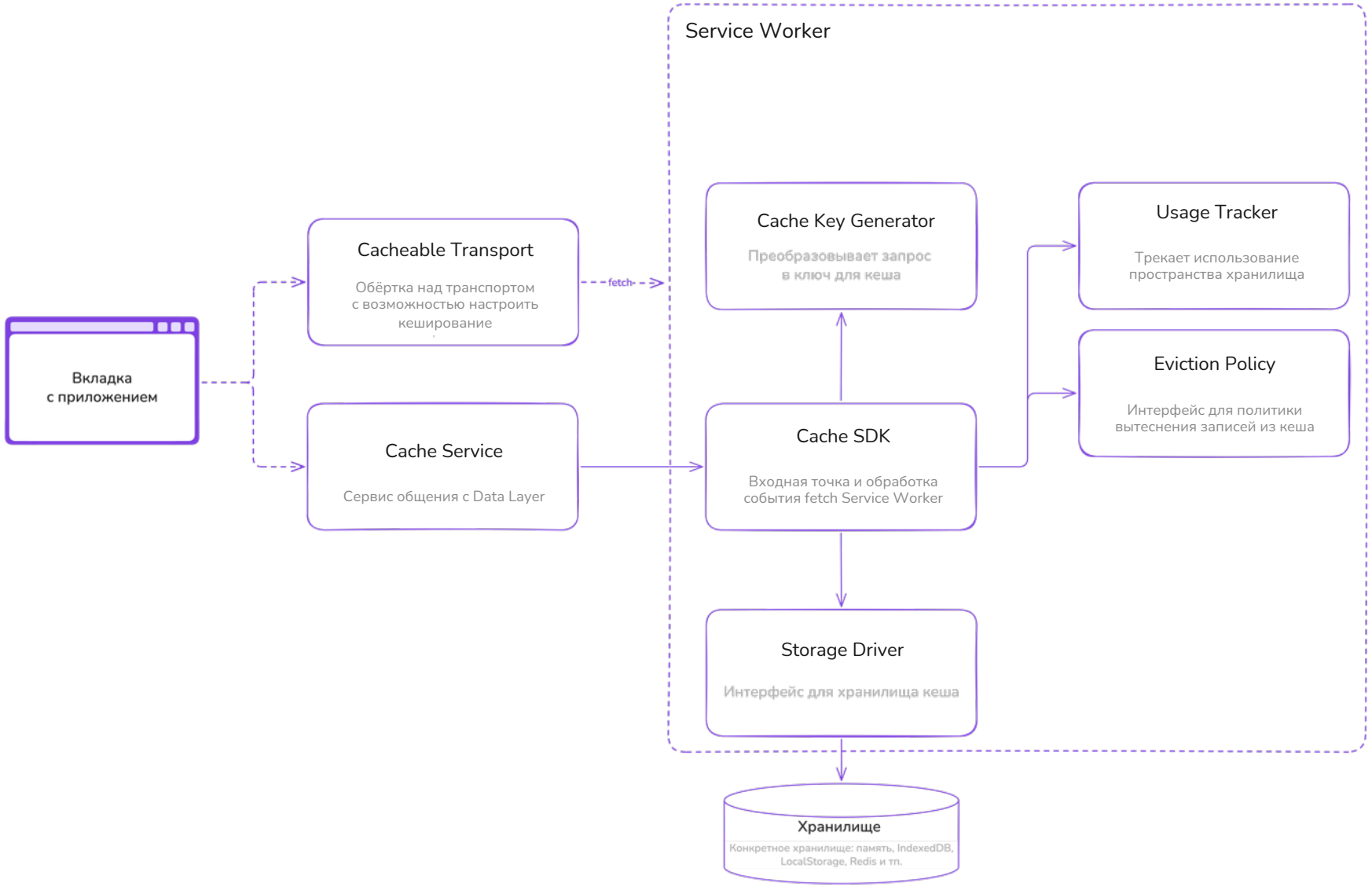
точка банк

Федеральный закон № 152-ФЗ «О персональных данных» от 27.07.2006 г. регулирует обработку, защиту и использование сведений, позволяющих идентифицировать граждан: ФИО, контакты, биометрия, здоровье.

Закон обязывает всех операторов — юридических лиц, ИП и госорганы — получать согласие на обработку данных, обеспечивать их конфиденциальность и безопасность, а также уведомлять Роскомнадзор о своей деятельности.

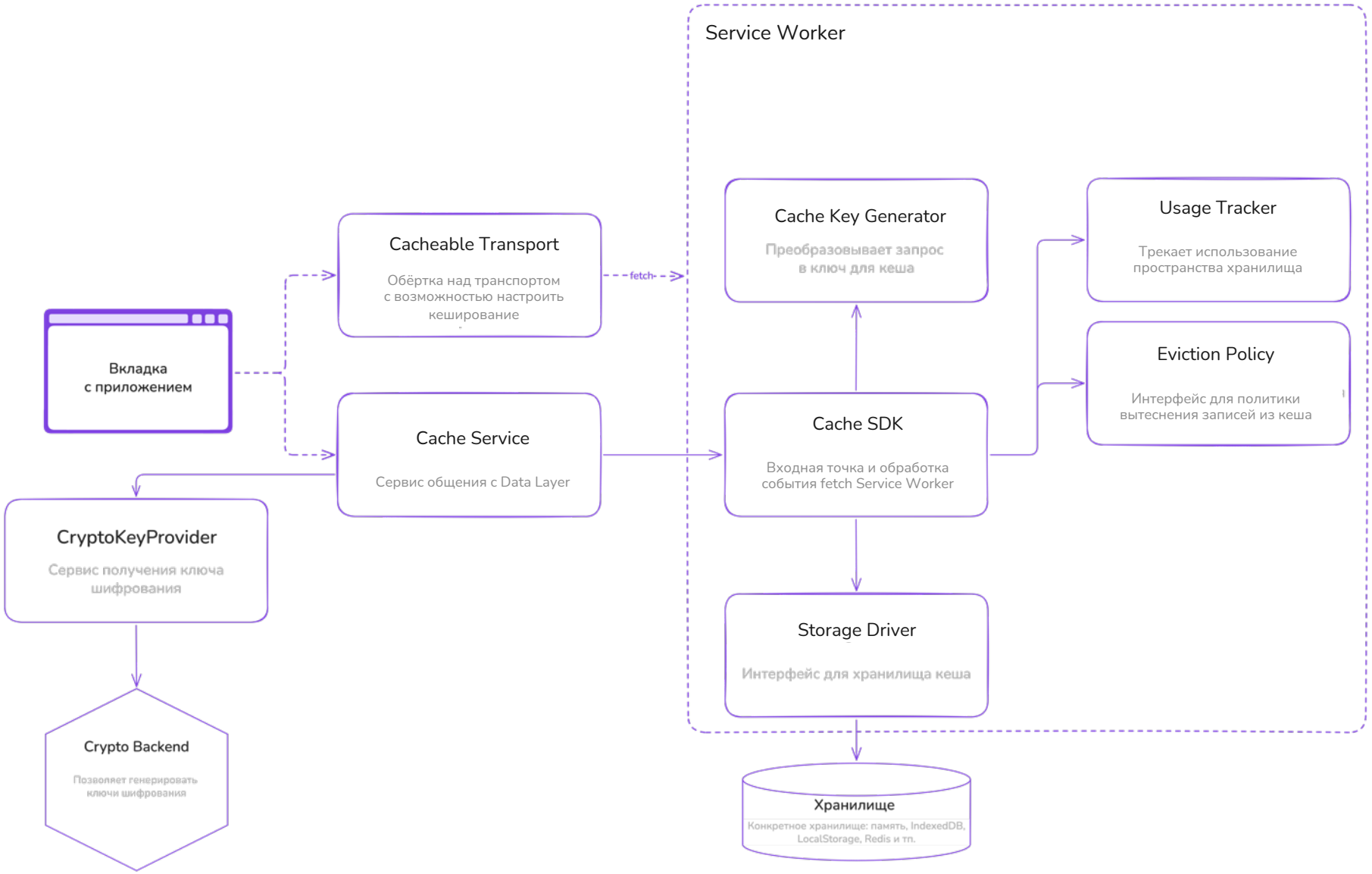
ШИФРУЕМСЯ

точка банк



ШИФРУЕМСЯ

точка банк



ШИФРУЕМСЯ

точка банк

```
1  const encrypt = async (data, key) => {
2    const encoded = encode(data)
3    const iv = generateIv()
4
5    const cipher = await window.crypto.subtle.encrypt({
6      name: 'AES-GCM',
7      iv
8    }, key, encoded)
9
10   return { cipher, iv };
11 }
```

```
1  // https://developer.mozilla.org/en-US/docs/Web/API/Crypto/getRandomValues
2  const generateIv = () =>
3    // https://developer.mozilla.org/en-US/docs/Web/API/AesGcmParams
4    window.crypto.getRandomValues(new Uint8Array(12))
5
```

ШИФРУЕМСЯ

точка банк

```
1  const encrypt = async (data, key) => {
2    const encoded = encode(data)
3    const iv = generateIv()
4
5    const cipher = await window.crypto.subtle.encrypt({
6      name: 'AES-GCM',
7      iv
8    }, key, encoded)
9
10   return { cipher, iv };
11 }
```

```
1  // https://developer.mozilla.org/en-US/docs/Web/API/Crypto/getRandomValues
2  const generateIv = () =>
3    // https://developer.mozilla.org/en-US/docs/Web/API/AesGcmParams
4    window.crypto.getRandomValues(new Uint8Array(12))
5
```

ШИФРУЕМСЯ

точка банк

```
1  const encrypt = async (data, key) => {
2    const encoded = encode(data)
3    const iv = generateIv()
4
5    const cipher = await window.crypto.subtle.encrypt({
6      name: 'AES-GCM',
7      iv
8    }, key, encoded)
9
10   return { cipher, iv };
11 }
```

```
1  // https://developer.mozilla.org/en-US/docs/Web/API/Crypto/getRandomValues
2  const generateIv = () =>
3    // https://developer.mozilla.org/en-US/docs/Web/API/AesGcmParams
4    window.crypto.getRandomValues(new Uint8Array(12))
5
```

ШИФРУЕМСЯ

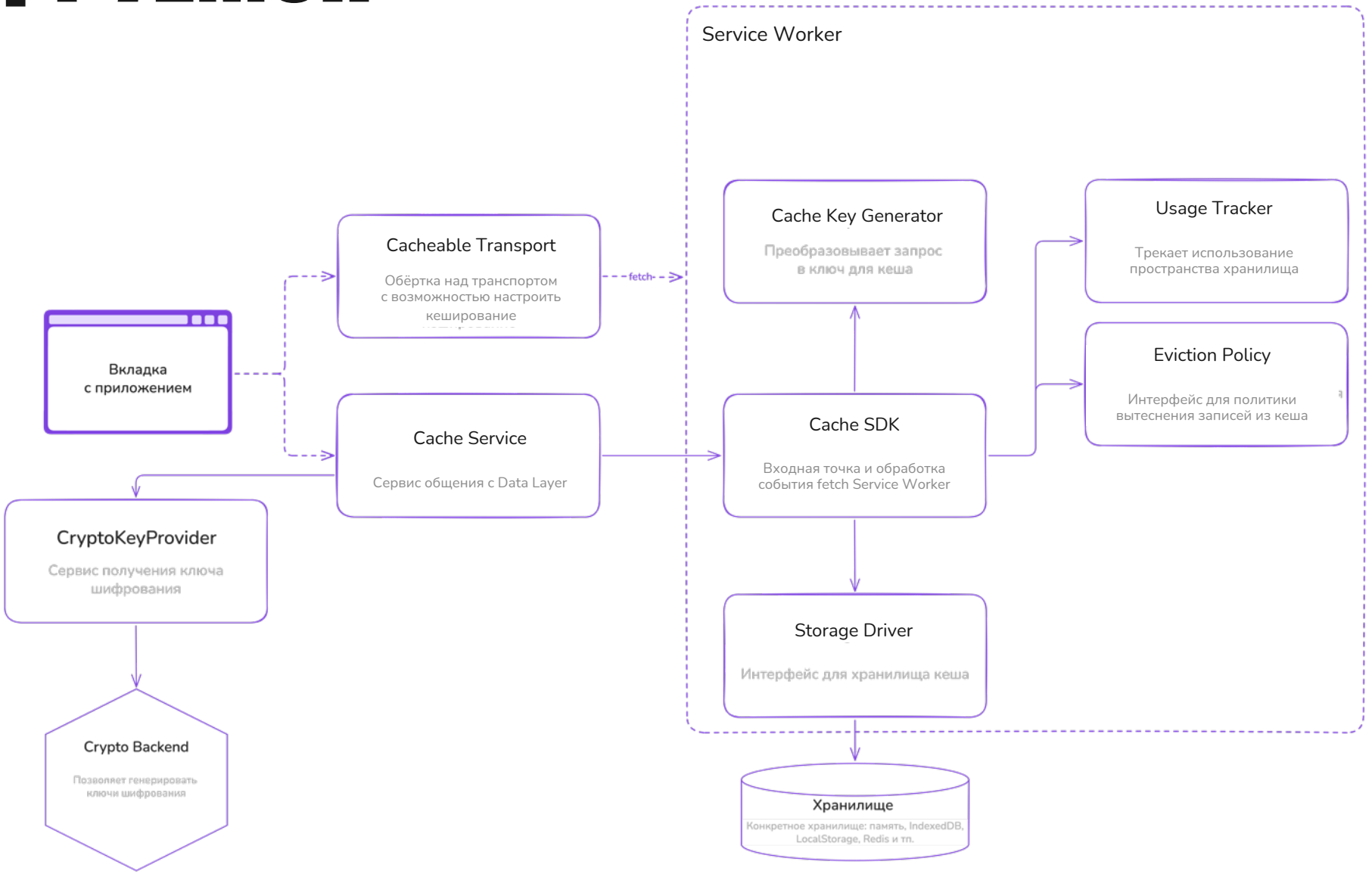
точка банк

```
1  const encrypt = async (data, key) => {
2    const encoded = encode(data)
3    const iv = generateIv()
4
5    const cipher = await window.crypto.subtle.encrypt({
6      name: 'AES-GCM',
7      iv
8    }, key, encoded)
9
10   return { cipher, iv };
11 }
```

```
1  // https://developer.mozilla.org/en-US/docs/Web/API/Crypto/getRandomValues
2  const generateIv = () =>
3    // https://developer.mozilla.org/en-US/docs/Web/API/AesGcmParams
4    window.crypto.getRandomValues(new Uint8Array(12))
5
```

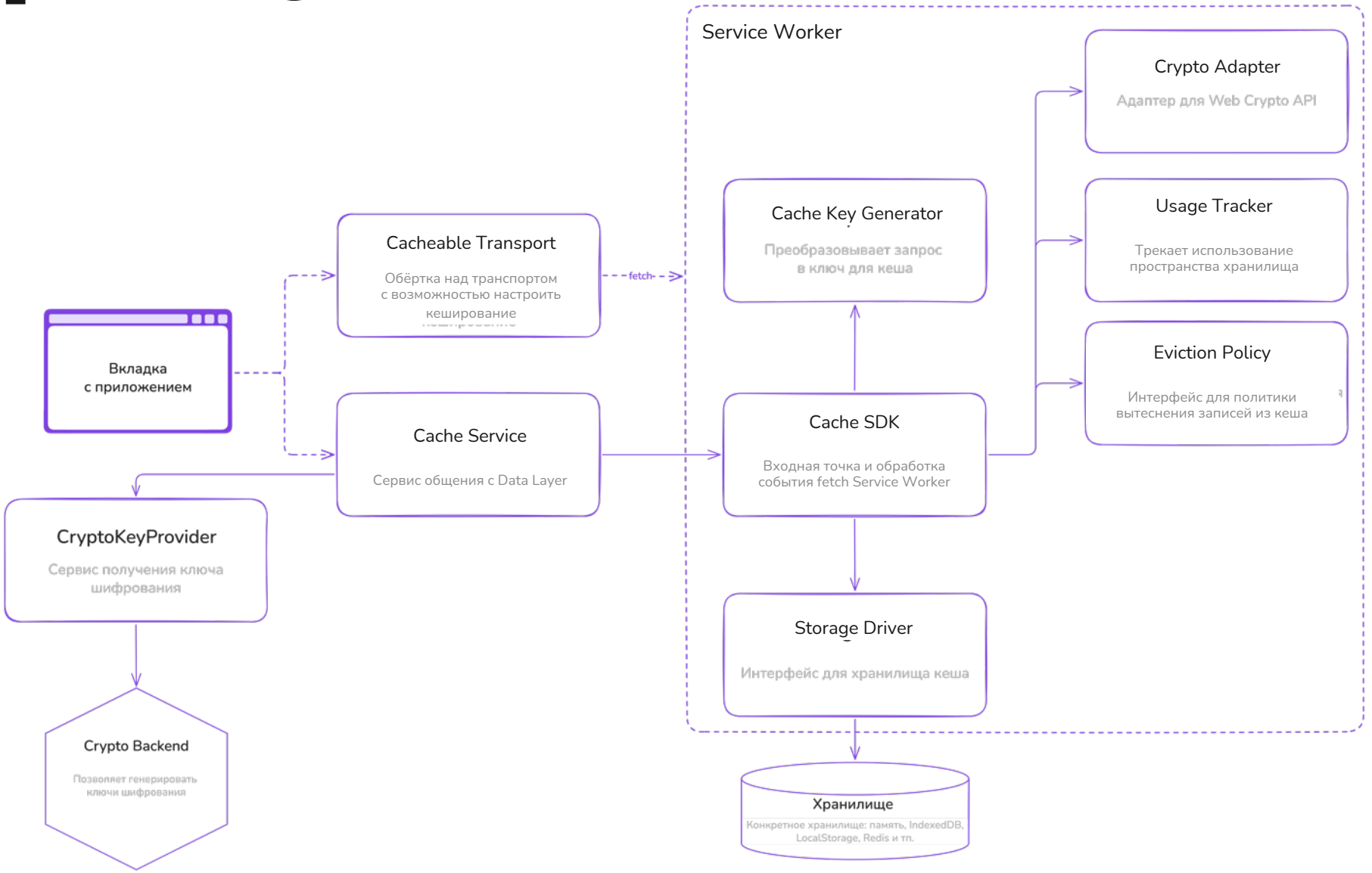
ШИФРУЕМСЯ

точка банк



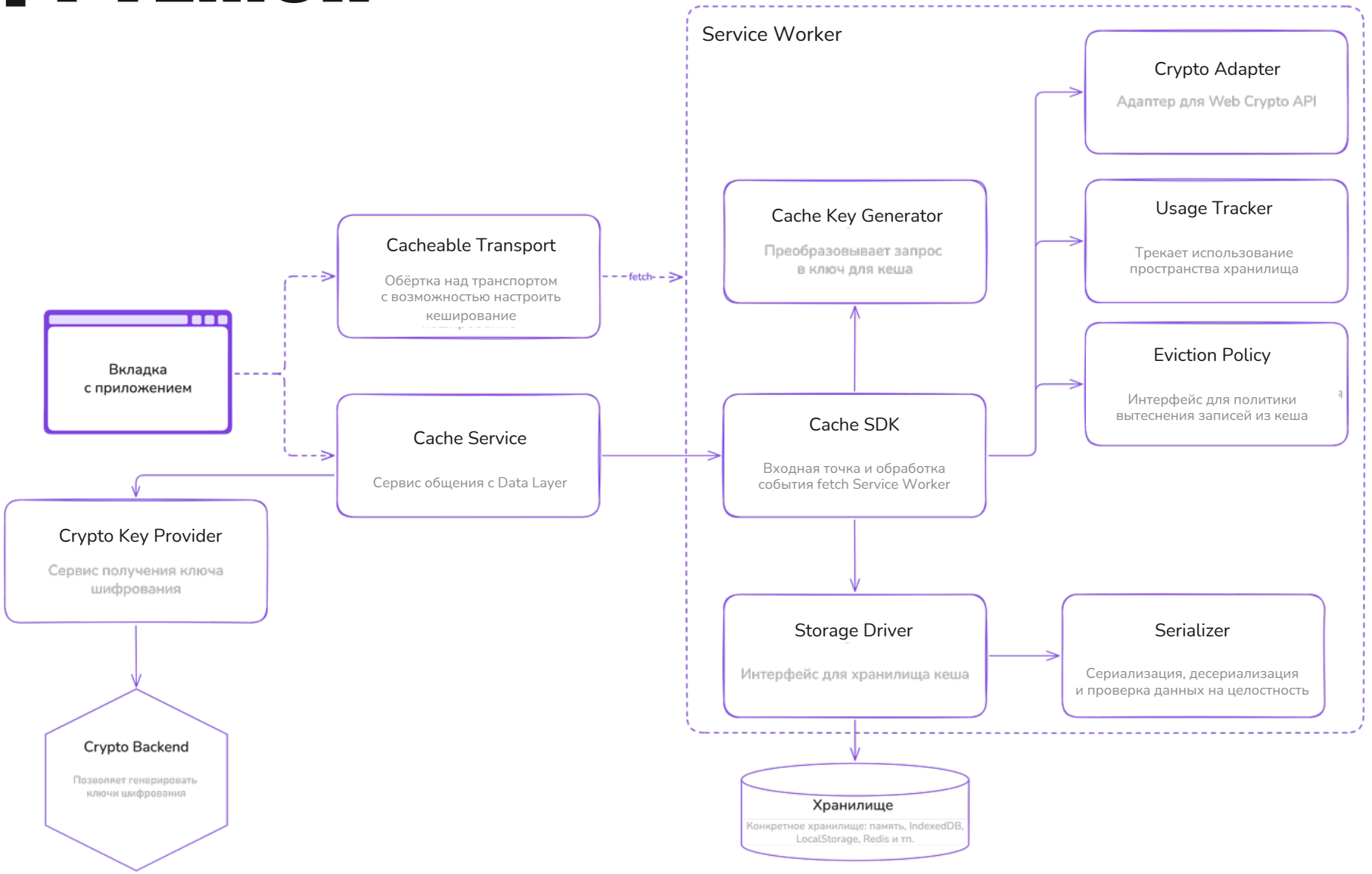
ШИФРУЕМСЯ

точка банк



ШИФРУЕМСЯ

точка банк



ШИФРУЕМСЯ

точка банк

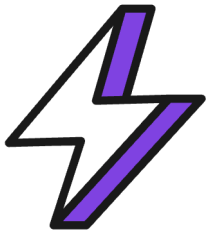
```
1  // Оборачиваем данные в объект для согласованной сериализации
2  const wrappedData = { data: record };
3  const jsonString = JSON.stringify(wrappedData);
4  const encoder = new TextEncoder();
5
6  return Promise.resolve(encoder.encode(jsonString));
```

А МОЖНО БЕЗ ЭТОГО?

точка банк

А МОЖНО БЕЗ ЭТОГО?

точка банк

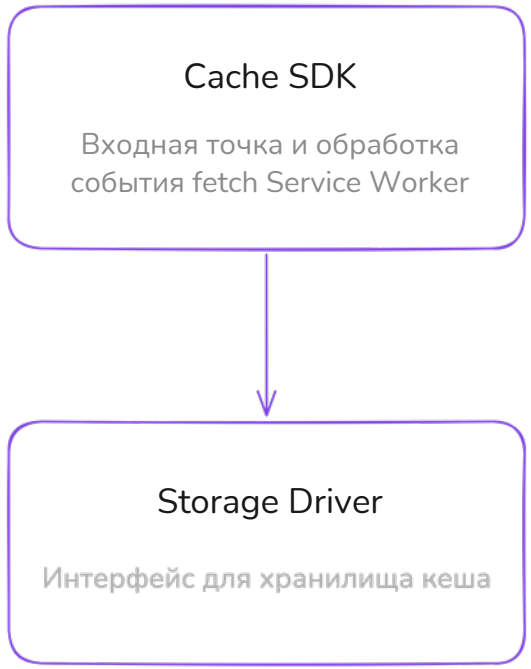


Подключаем горячий кеш!

Данные в памяти можно
не шифровать

ГРЕЕМ КЕШ

точка банк

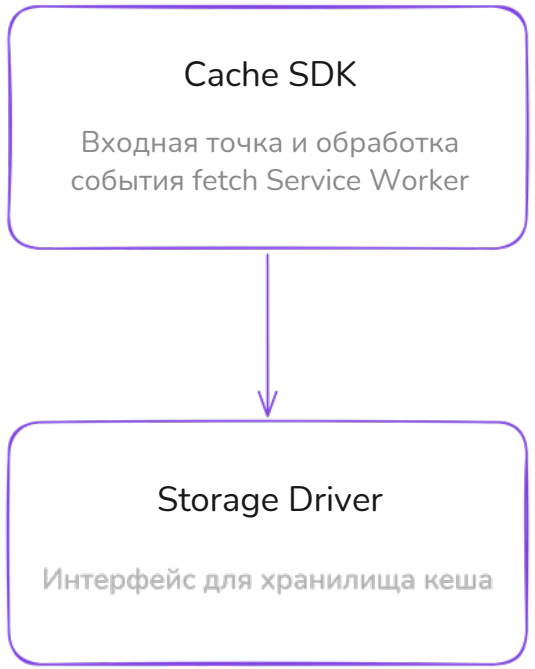


ГРЕЕМ КЕШ

точка банк

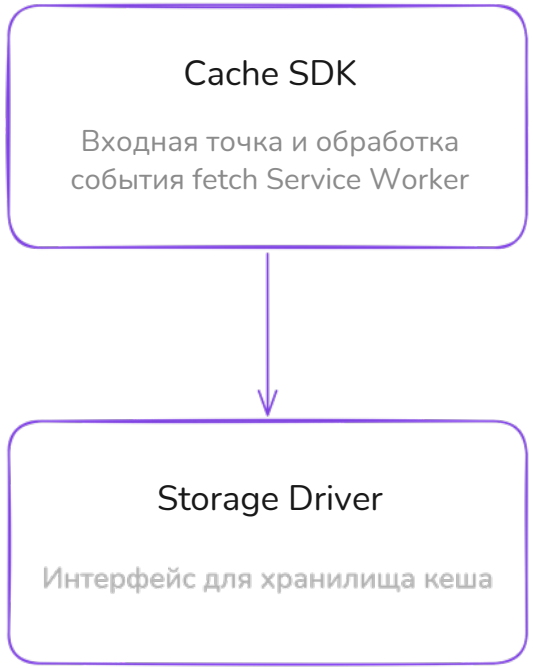
Most Firstly Used

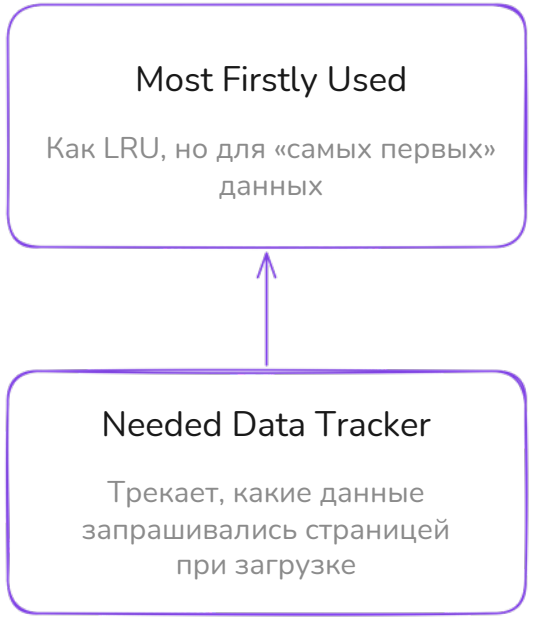
Как LRU, но для «самых первых» данных



ГРЕЕМ КЕШ

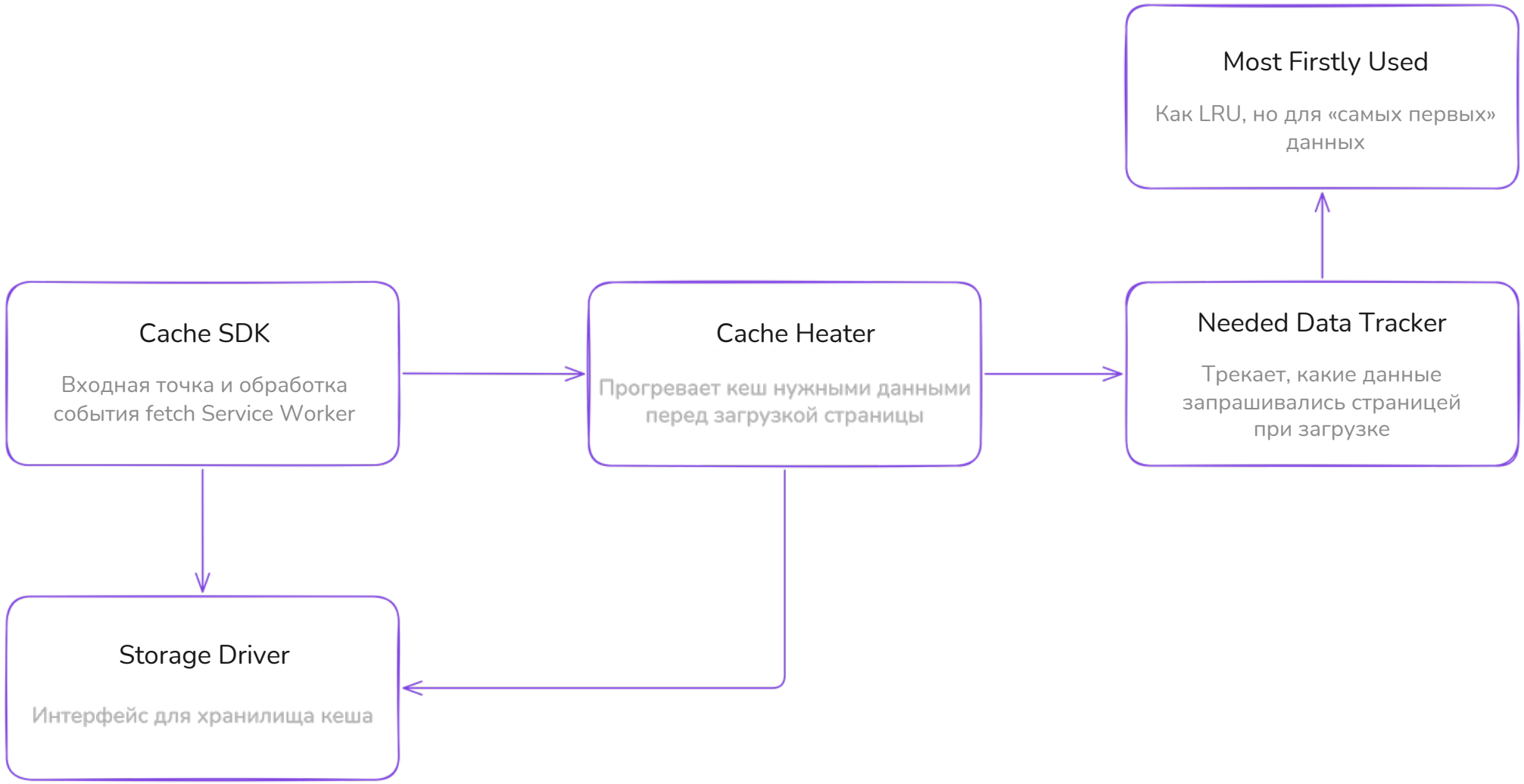
точка банк





ГРЕЕМ КЕШ

точка банк



ИНВАЛИДАЦИЯ

точка банк

Меняем актуальность данных на производительность

СТРАТЕГИИ ИНВАЛИДАЦИИ

точка банк

СТРАТЕГИИ ИНВАЛИДАЦИИ

точка банк

Time To Live (TTL)

Пока не прошло время, считаем данные актуальными

СТРАТЕГИИ ИНВАЛИДАЦИИ

точка банк

Time To Live (TTL)

Пока не прошло время,
считаем данные
актуальными

Stale While Revalidate (SWR)

Показываем из кеша,
идём за новыми данными
и в следующий раз
покажем новые

А МОЖНО ИНАЧЕ?

точка банк

А МОЖНО ИНАЧЕ?

точка банк

Завязаться на RESTfull

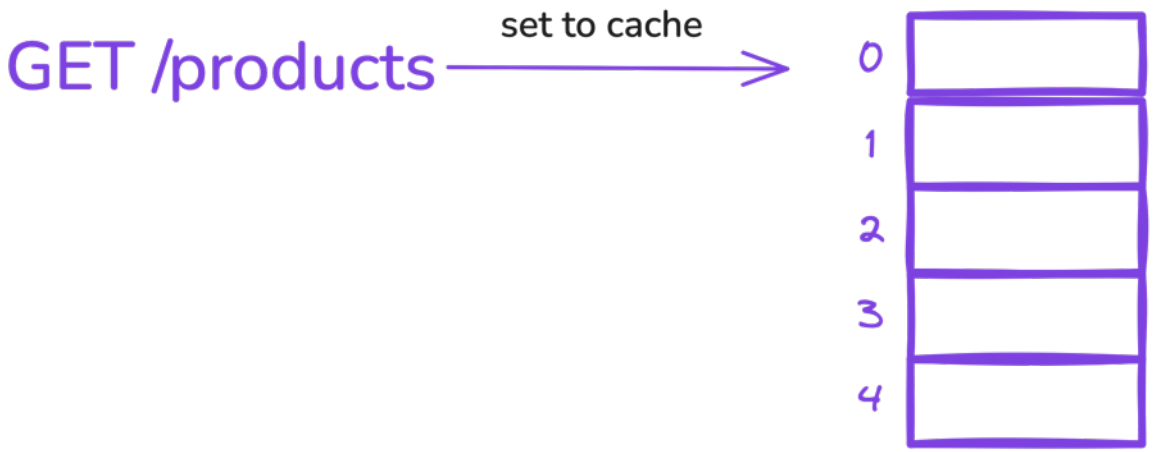
GET запросы пишут в кеш
POST/PUT/DELETE — этот кеш инвалидируют

А МОЖНО ИНАЧЕ?

точка банк

Завязаться на RESTfull

GET запросы пишут в кеш
POST/PUT/DELETE — этот кеш инвалидируют



Мы

А МОЖНО ИНАЧЕ?

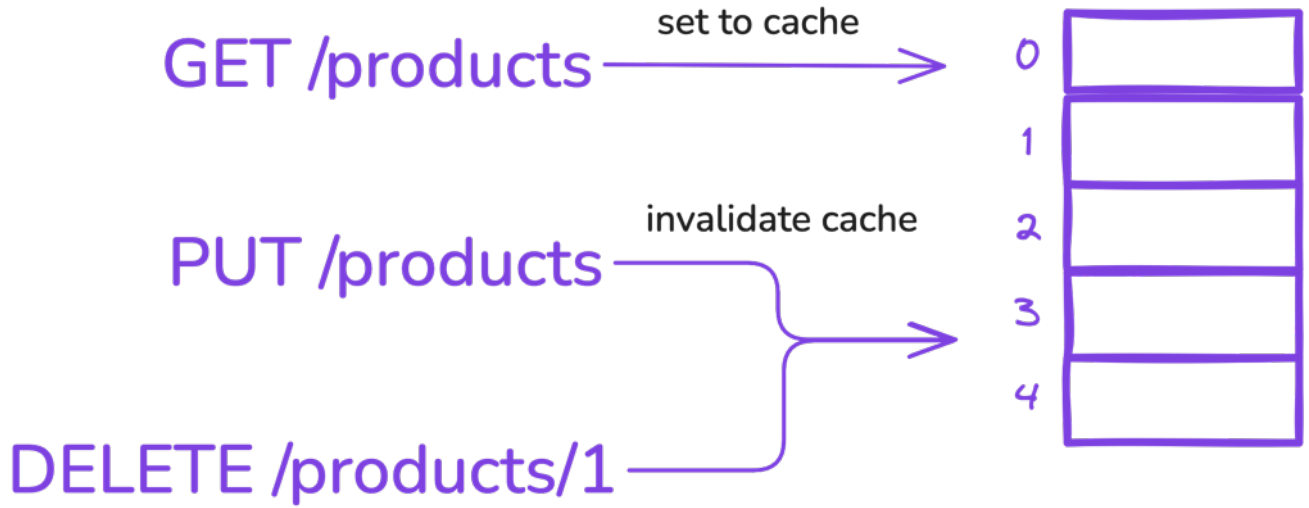
точка банк

Завязаться на RESTfull

GET запросы пишут в кеш

POST/PUT/DELETE — этот кеш инвалидируют

Мы



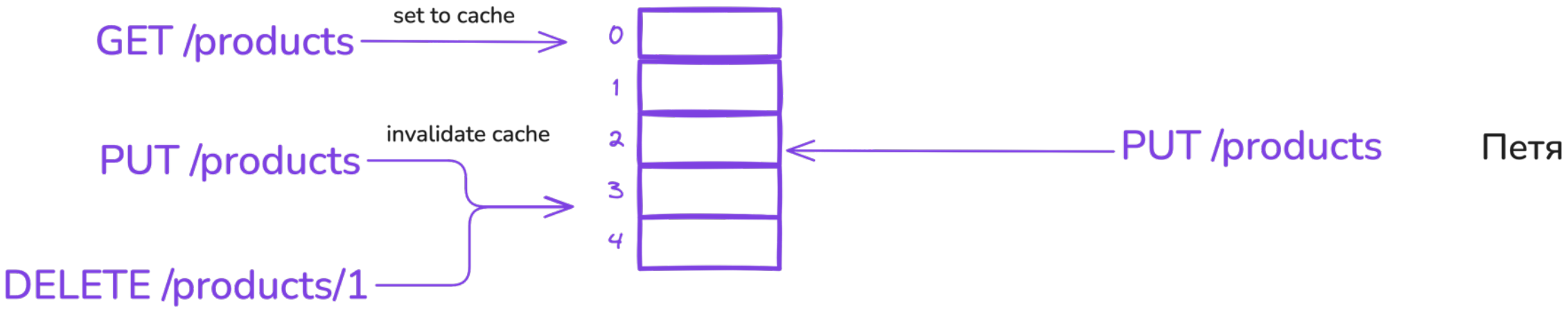
А МОЖНО ИНАЧЕ?

точка банк

Завязаться на RESTfull

GET запросы пишут в кеш

POST/PUT/DELETE — этот кеш инвалидируют



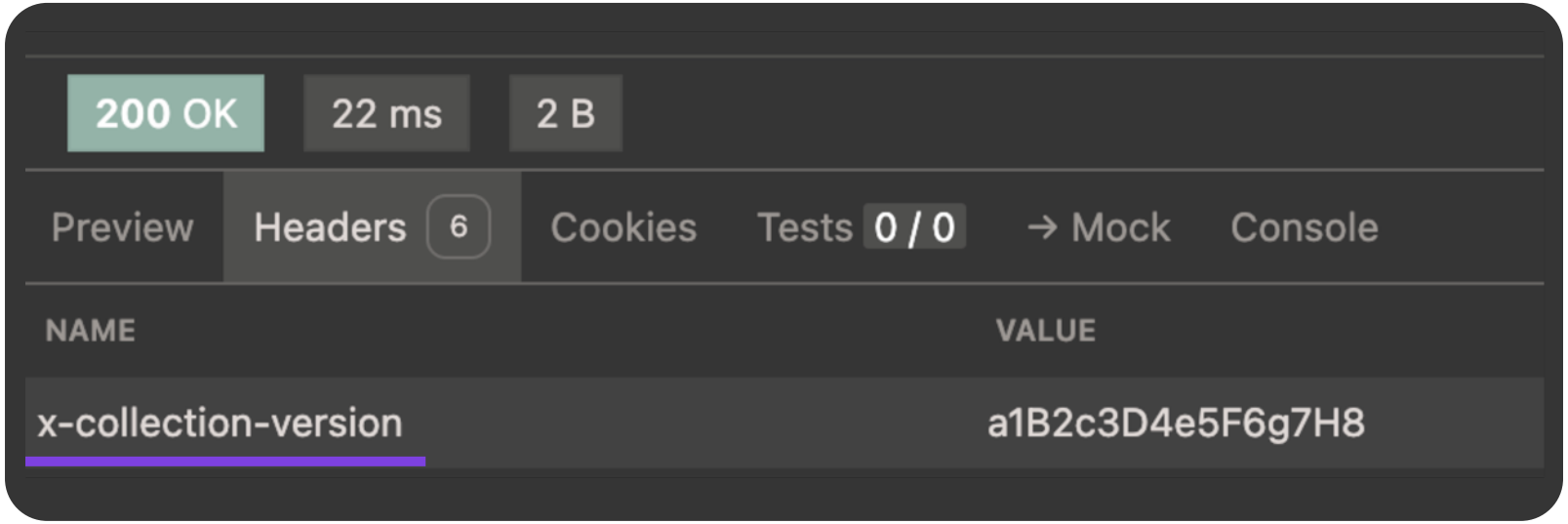
А МОЖНО ИНАЧЕ?

точка банк

Завязаться на RESTfull

GET запросы пишут в кеш
POST/PUT/DELETE — этот кеш инвалидируют

GET /products →



А МОЖНО ИНАЧЕ?

точка банк

Завязаться на RESTfull

GET запросы пишут в кеш
POST/PUT/DELETE — этот кеш инвалидируют

```
// POST /api
{
    "jsonrpc": "2.0",
    "method": "rpc.request",
    "params": {
        ...
    }
}
```

А МОЖНО ИНАЧЕ?

точка банк

Завязаться на RESTfull

GET запросы пишут в кеш
POST/PUT/DELETE — этот кеш инвалидируют

Граф запросов

Тупо хардкод в стиле «на запрос GET /products влияет PUT/products, POST /products/1» и так далее

А МОЖНО ИНАЧЕ?

точка банк

Завязаться на RESTfull

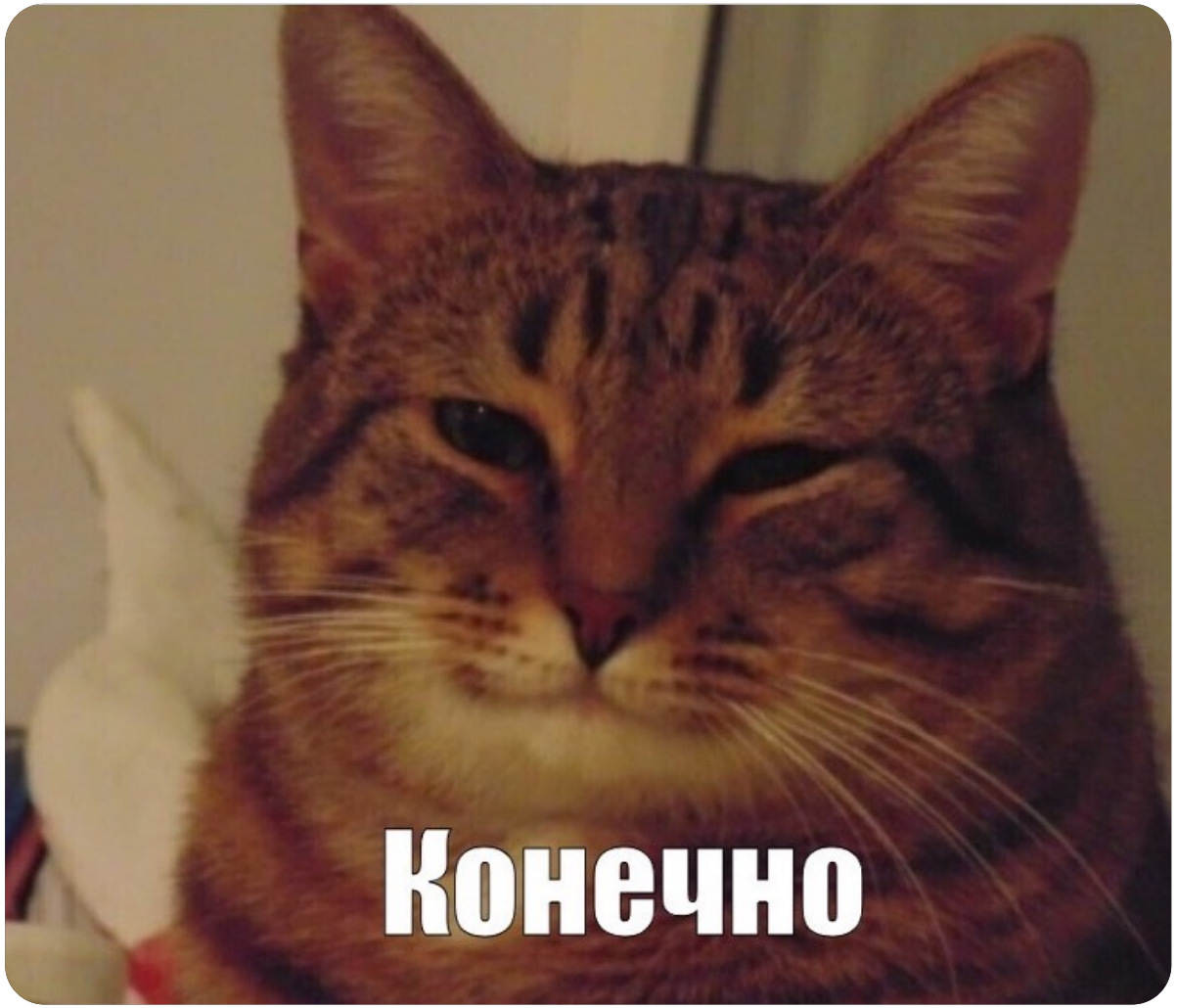
GET запросы пишут в кеш
POST/PUT/DELETE — этот кеш инвалидируют

Граф запросов

Тупо хардкод в стиле «на запрос GET /products влияет PUT/products, POST /products/1» и так далее

```
{
    "GET /products": [
        "POST /products/:id",
        "DELETE /products/:id",
        "PUT /products"
    ]
}
```

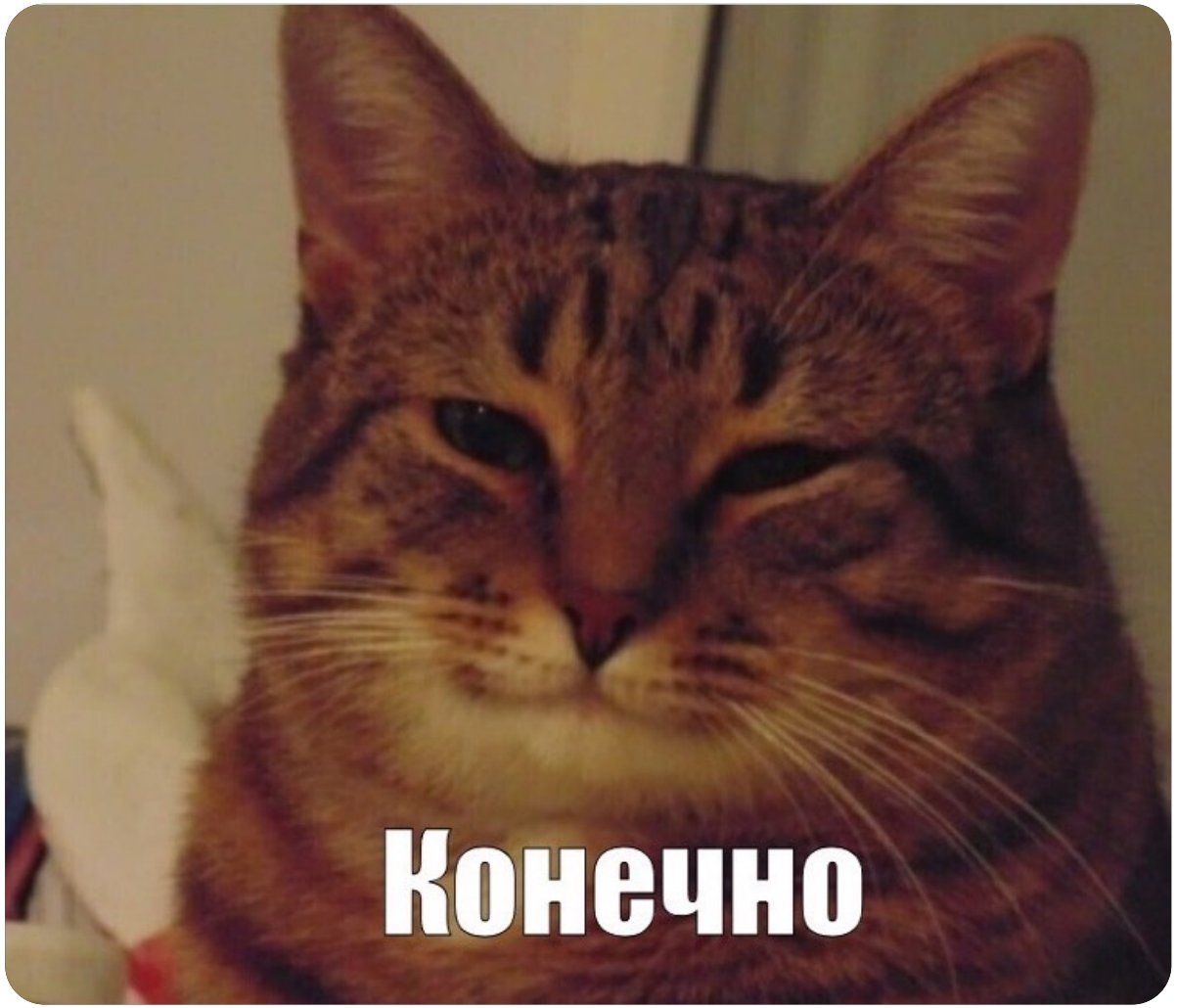
А МОЖНО ИНАЧЕ?



А МОЖНО ИНАЧЕ?

В стиле TanStack Query

Иметь систему тегов, чтобы помечать зависимые запросы



А МОЖНО ИНАЧЕ?

точка банк

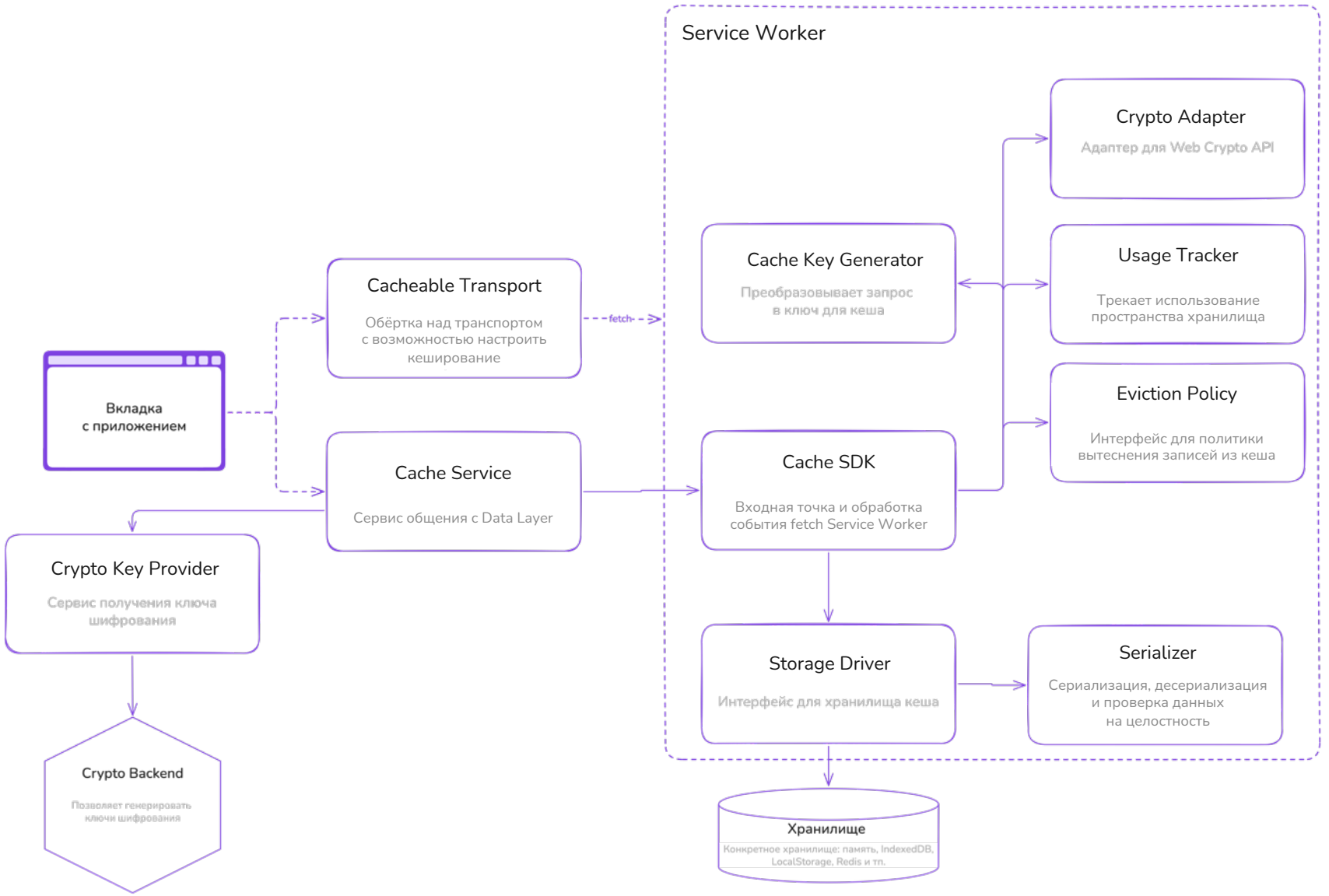
В стиле TanStack Query

Иметь систему тегов, чтобы помечать зависимые запросы

```
1
2  function Todos({ todoId }) {
3      const result = useQuery({
4        queryKey: ['todos', todoId],
5        queryFn: () => fetchTodoById(todoId),
6      })
7    }
```

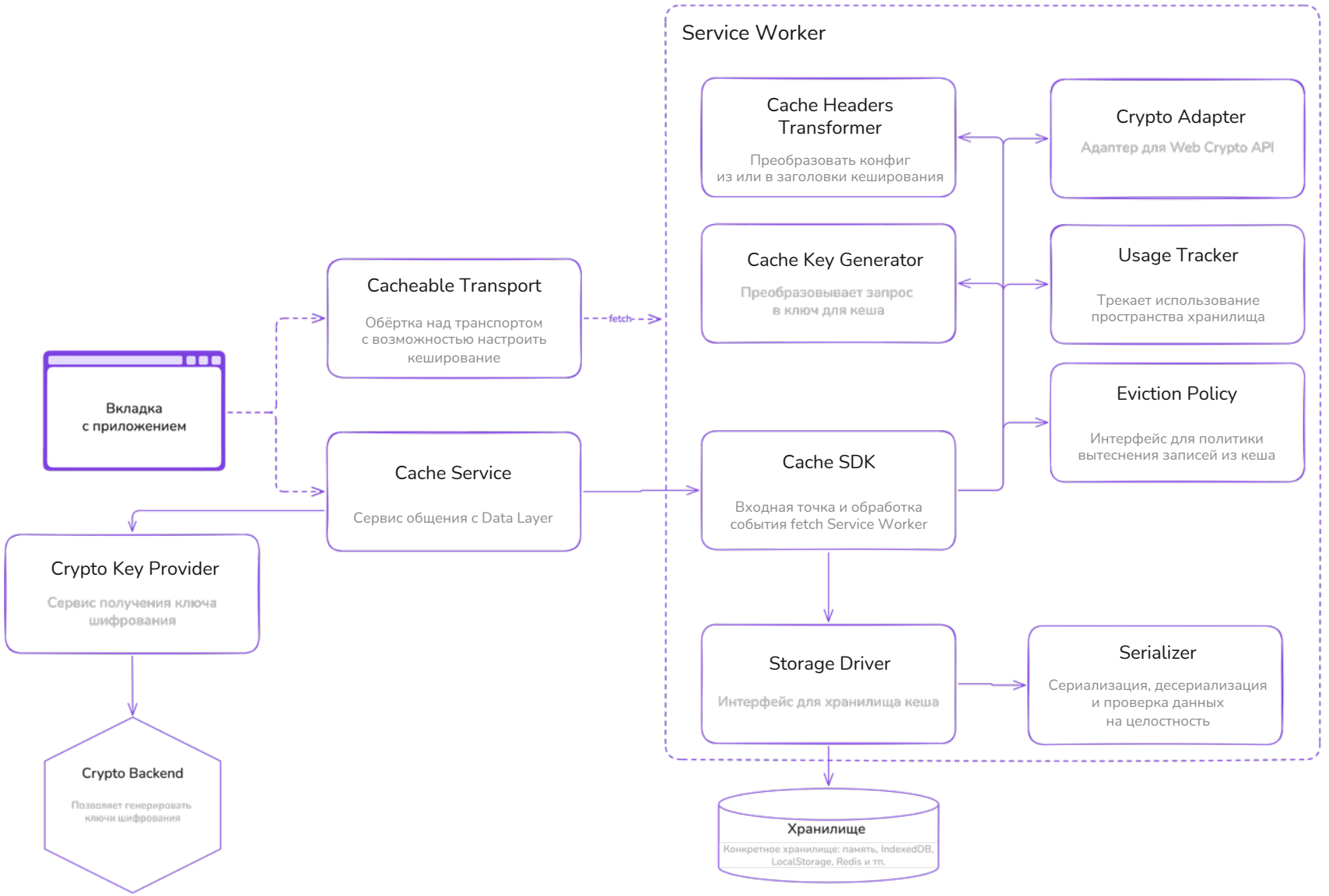
КОНФИГАЕМ ЧЕРЕЗ ЗАГОЛОВКИ

точка банк



КОНФИГАЕМ ЧЕРЕЗ ЗАГОЛОВКИ

точка банк



ДВОЙНОЕ ОБНОВЛЕНИЕ

точка банк

точка банк

ЭТО ЧТО, МНЕ ТЕПЕРЬ НАЧИНАТЬ РАЗБИРАТЬСЯ В БИЗНЕС-ЛОГИКЕ СВОЕГО ПРОЕКТА?

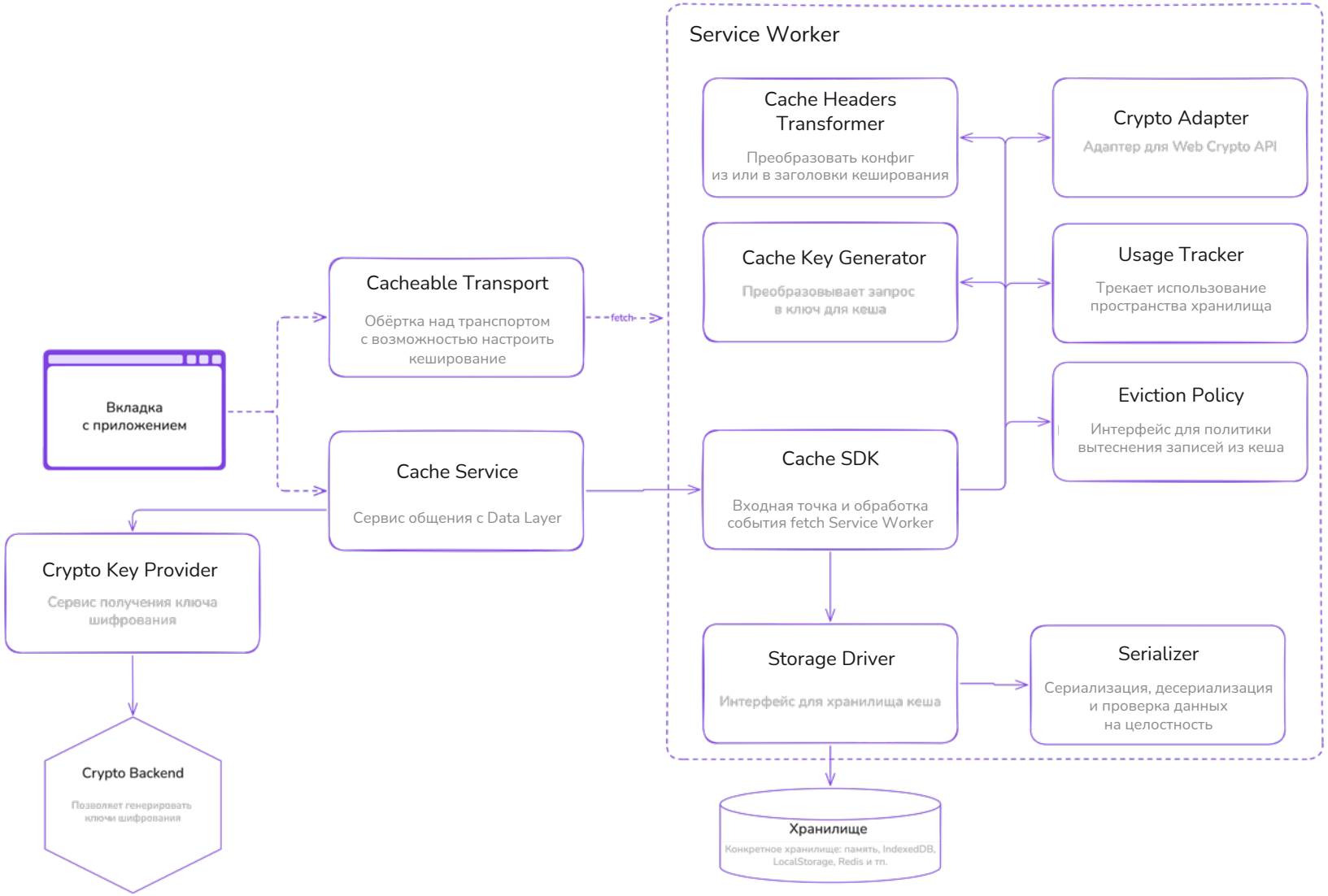
Кто-то из нас

ДВОЙНОЕ ОБНОВЛЕНИЕ

точка банк

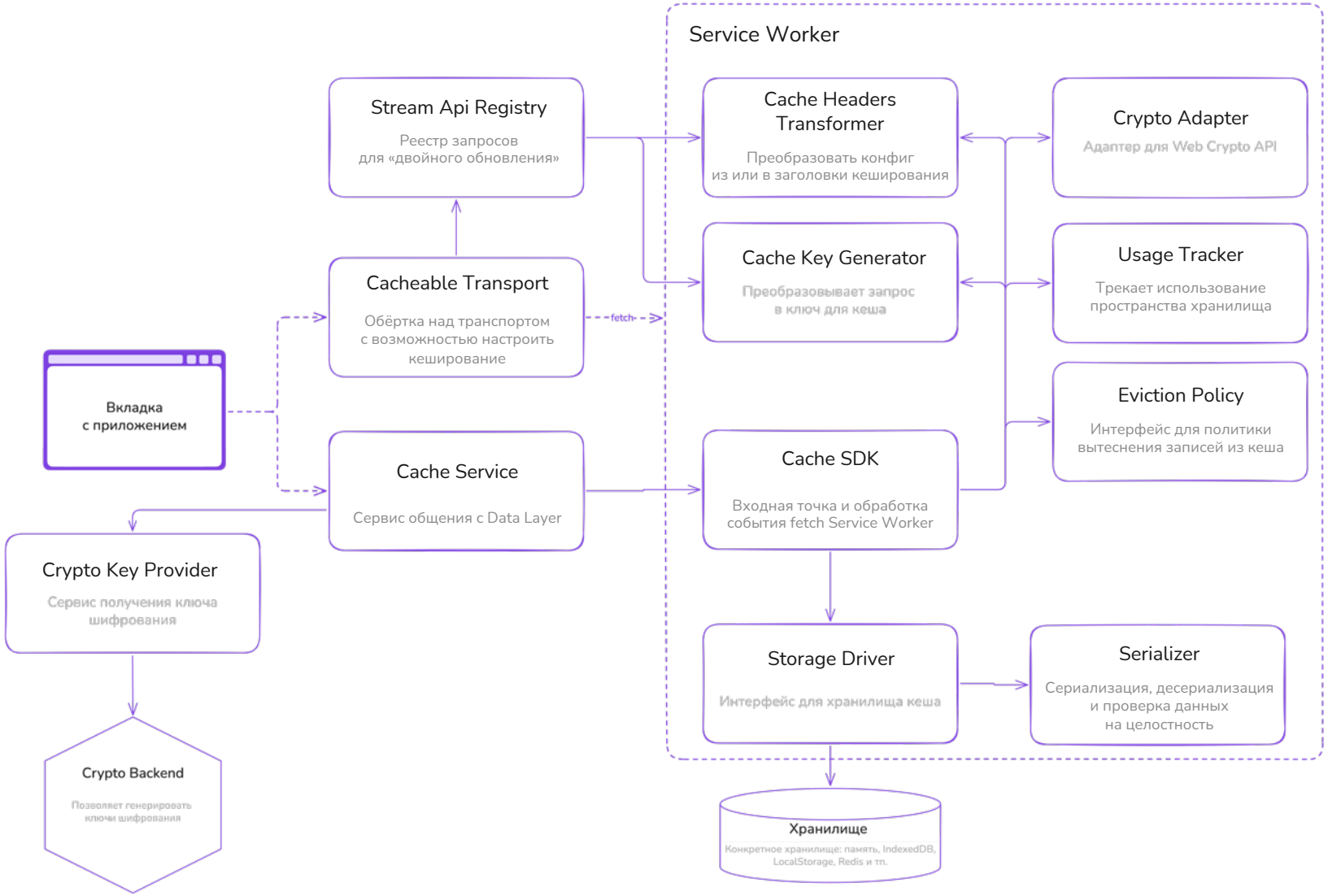
ДВОЙНОЕ ОБНОВЛЕНИЕ КАК ОПЦИЯ

точка банк



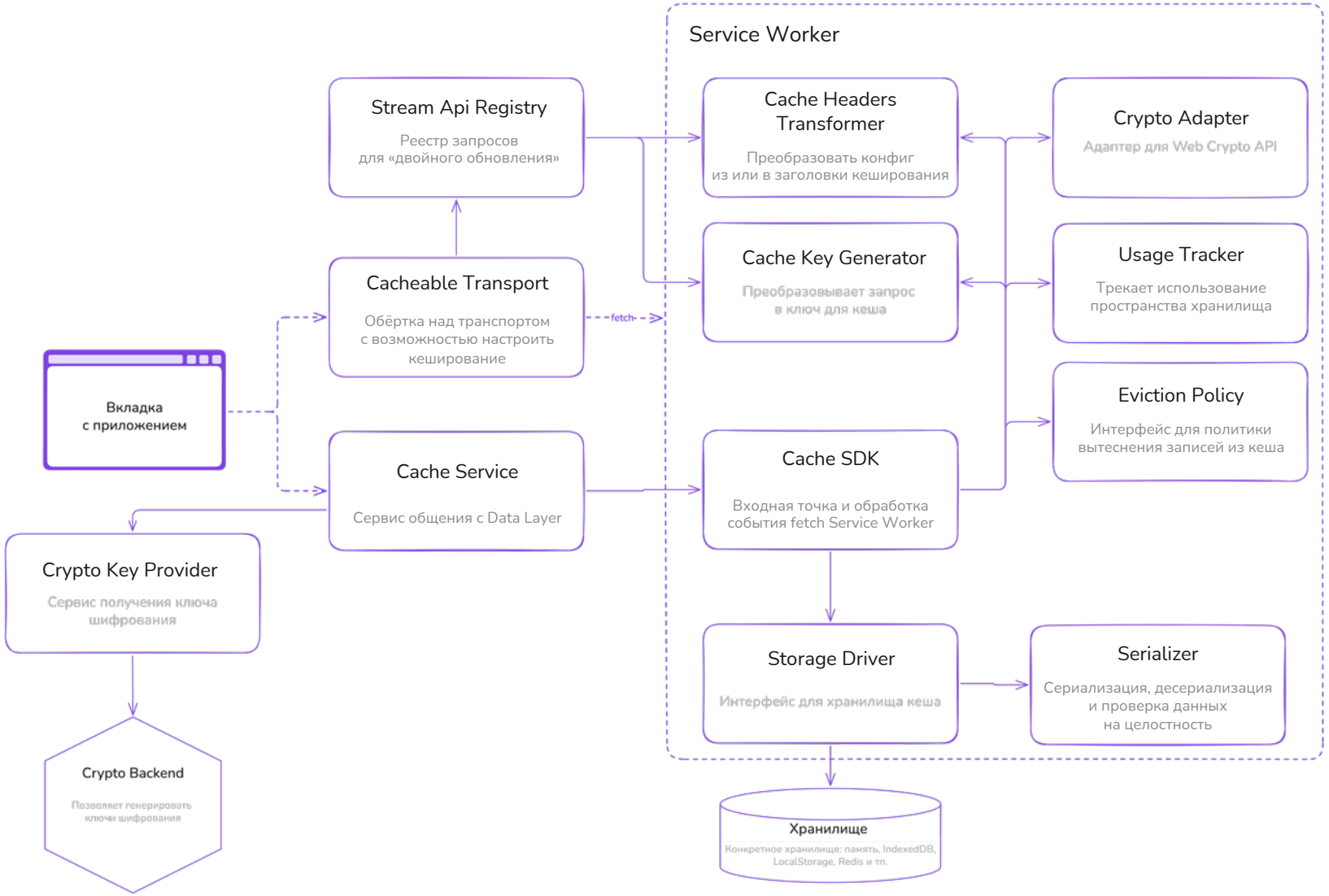
ДВОЙНОЕ ОБНОВЛЕНИЕ КАК ОПЦИЯ

точка банк



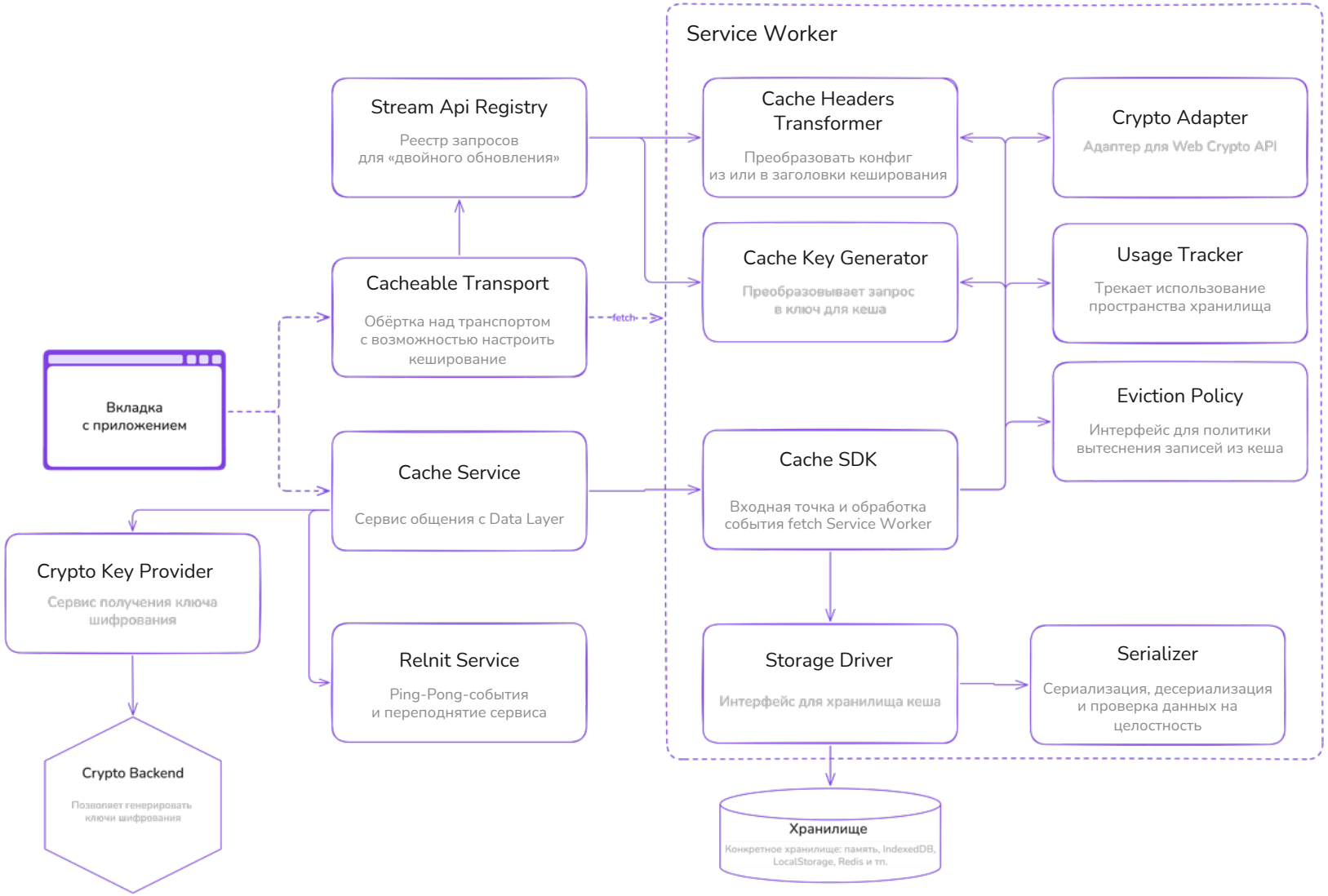
HEALTH-CHECK И РЕИНИЦИАЛИЗАЦИЯ

точка банк



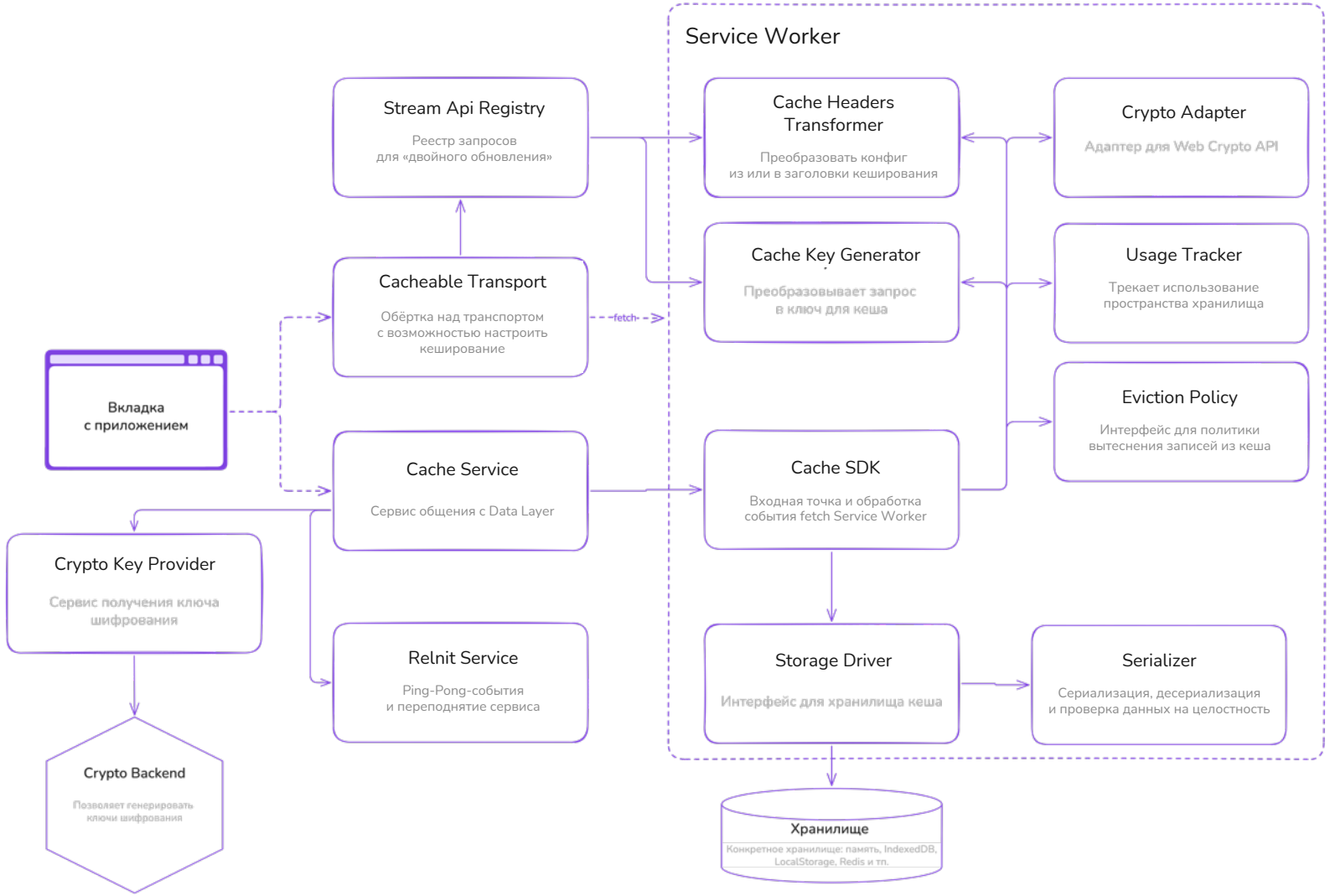
HEALTH-CHECK И РЕИНИЦИАЛИЗАЦИЯ

точка банк



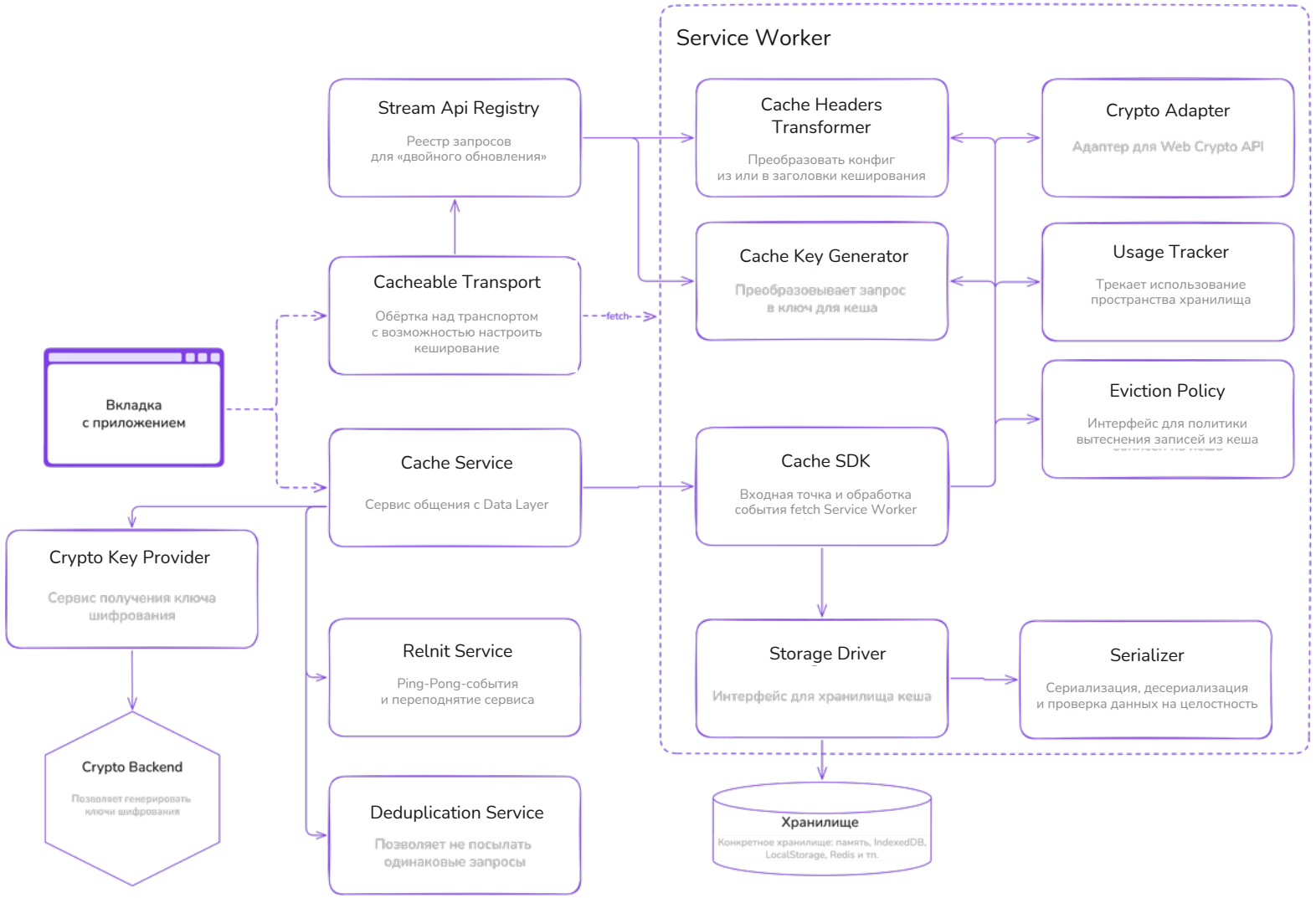
ДЕДУПЛИКАЦИЯ ЗАПРОСОВ

точка банк



ДЕДУПЛИКАЦИЯ ЗАПРОСОВ

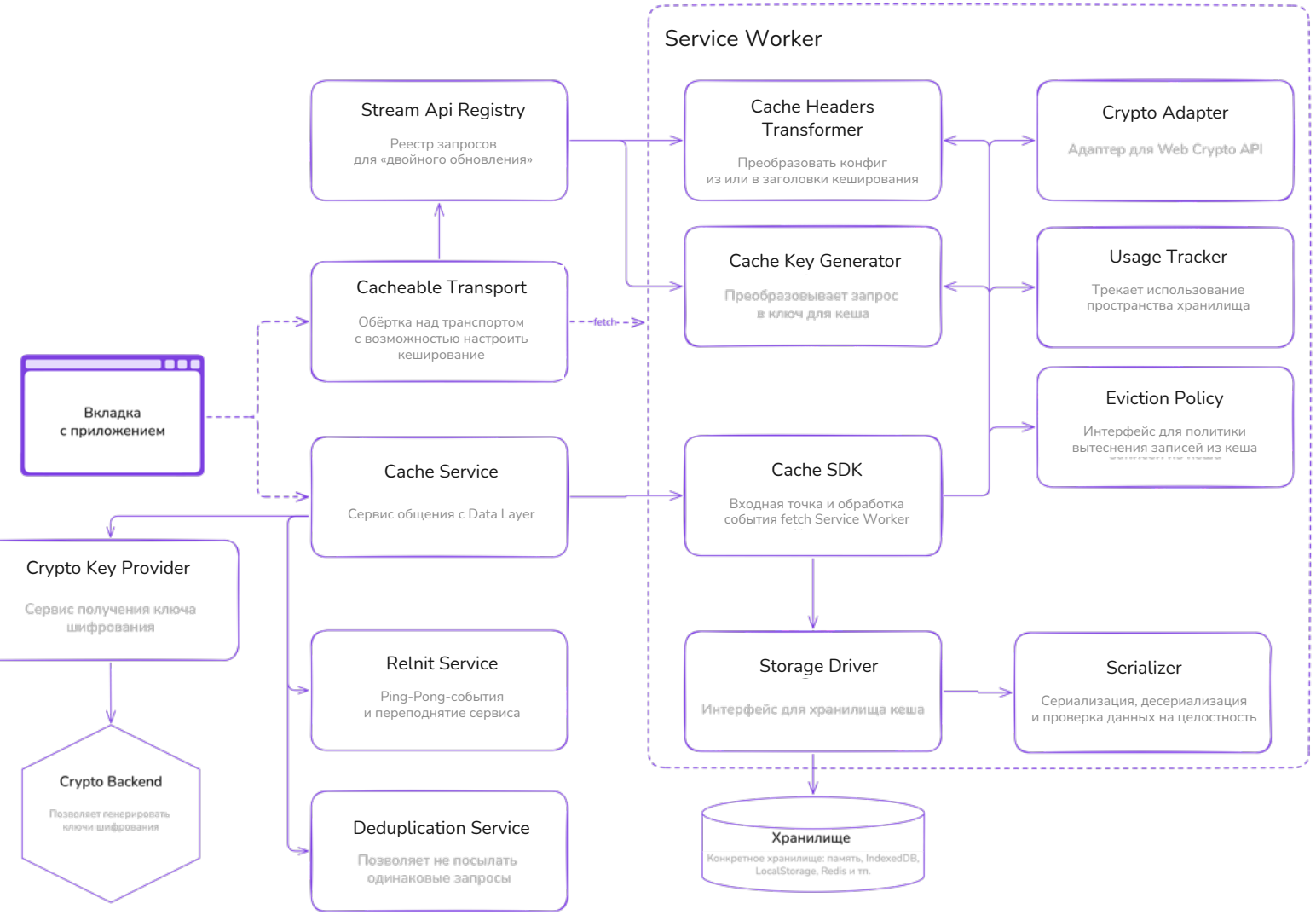
точка банк



ДЕДУПЛИКАЦИЯ ЗАПРОСОВ

точка банк

200	GET	/static/v1/ui-kit/pictograms/Mono	fetch
200	POST	/api/v1/cs-api	xhr
200	POST	/api/v1/task-manager/jsonrpc/priv	xhr
200	POST	/api/v1/stepler/v3/jsonrpc	xhr
200	POST	/api/v1/stepler/v3/jsonrpc	xhr
200	POST	/api/v1/stepler/v3/jsonrpc	xhr
200	POST	/api/v1/quasi-overdraft/jsonrpc	xhr
200	POST	/api/v1/cs-business-risk/v2/jsonr	xhr



ВСЁ?

точка банк

НЕ ЗАБЫВАЕМ ПРО CI

НУ И НАДО ОНО БЫЛО?

точка банк

ПРОДАМ ~~ГАРАЖ~~ КЕШИРОВАНИЕ

точка банк

ПРОДАМ ~~ГАРАЖ~~ КЕШИРОВАНИЕ

точка банк

1. Удобство интерфеса

ПРОДАМ ~~ГАРАЖ~~ КЕШИРОВАНИЕ

точка банк

1. Удобство интерфеса
2. Частичный офлайн

ПРОДАМ ~~ГАРАЖ~~ КЕШИРОВАНИЕ

точка банк

1. Удобство интерфеса
2. Частичный офлайн
3. Скорость интерфейса

ПРОДАМ ~~ГАРАЖ~~ КЕШИРОВАНИЕ

точка банк

1. Удобство интерфеса
2. Частичный офлайн
3. Скорость интерфейса

4. **Меньше пострадавших**
   При сбое меньше народу пойдёт на упавший сервис

ПРОДАМ ~~ГАРАЖ~~ КЕШИРОВАНИЕ

точка банк

1. Удобство интерфеса
2. Частичный офлайн
3. Скорость интерфейса

4. **Меньше пострадавших**
   При сбое меньше народу пойдёт на упавший сервис

5. **Снижение RPS**
   Меньше запросов — меньше нагрузка на бэкенды

ЦИФЕРКИ, ЦИФЕРКИ, ЦИФЕРКИ!

точка банк

Запрос query_customer_tariffs

	По сети (мс)	Из кеша (мс)	Разница (мс)
p50	168	3	165
p90	350	27	323
p99	1244	416	828

ЦИФЕРКИ, ЦИФЕРКИ, ЦИФЕРКИ!

точка банк

Запрос timeline_get_count

	По сети (мс)	Из кеша (мс)	Разница (мс)
p50	156	5	151
p90	353	56	297
p99	1646,5	477,7	1168,8

ЦИФЕРКИ, ЦИФЕРКИ, ЦИФЕРКИ!

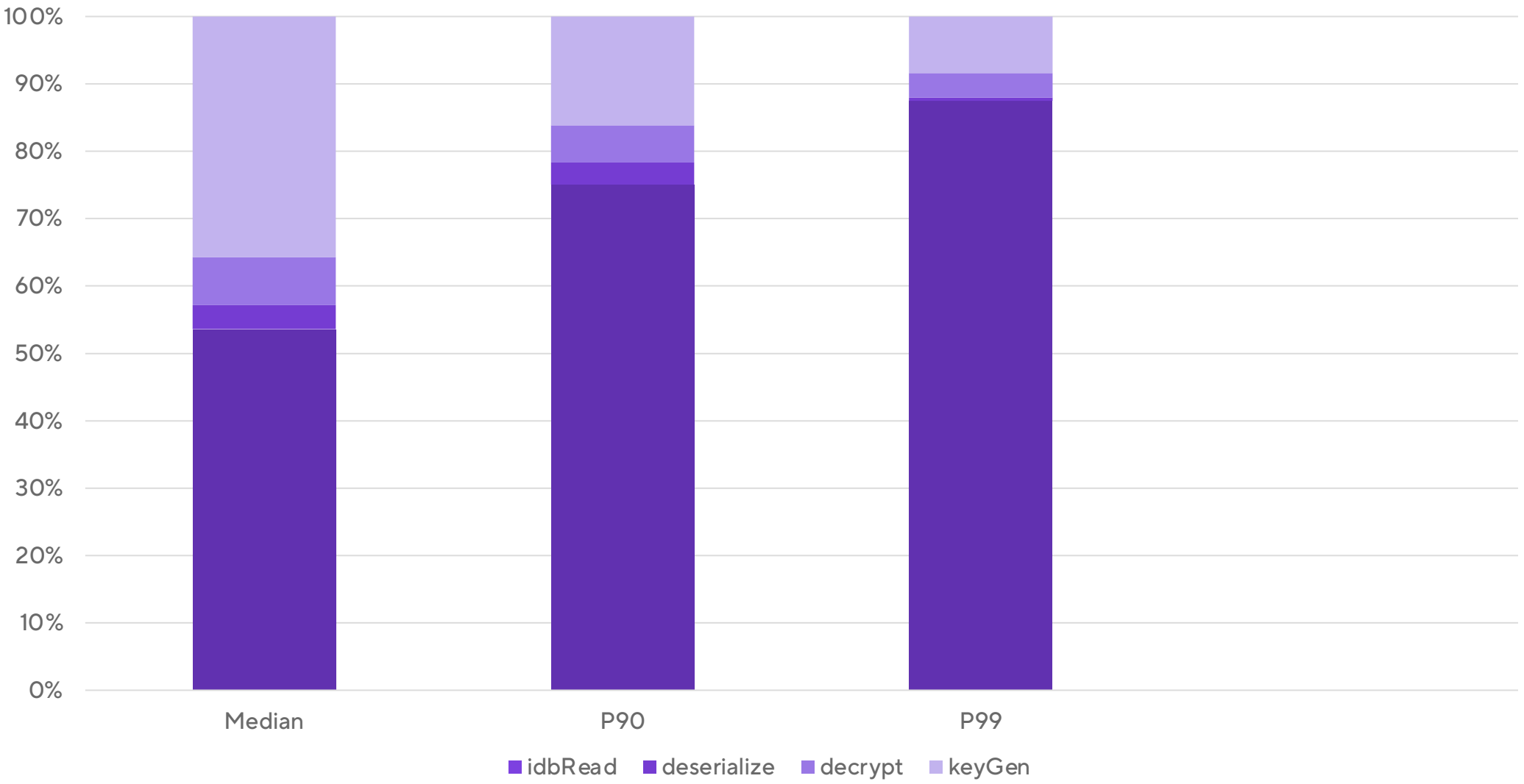
точка банк

Запрос timeline_get_list

	По сети (мс)	Из кеша (мс)	Разница (мс)
p50	183,5	6,5	177
p90	418	69	349
p99	1710,9	484,9	1226

РАСПРЕДЕЛЕНИЕ

точка банк



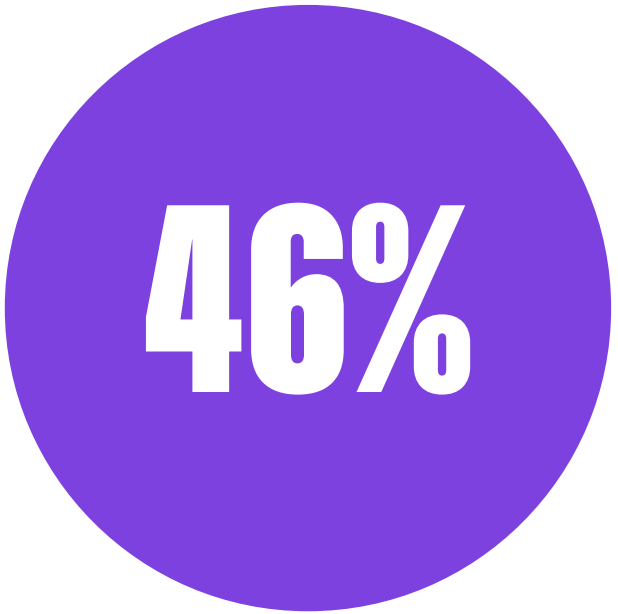
HIT-RATE

точка банк

Query_customer_tariffs

Timeline_get_count

Timeline_get_list



ЧТО МОНИТОРИТЬ?

точка банк

Hit-Rate (попадание в кеш)

С разбивкой по URL’ам

Время чтения из кеша

Тоже с разбивкой

Влияние на RPS и скорость

Это уже на бэке

ЕСТЬ ЖЕ HTTP-КЕШ

точка банк

ЕСТЬ ЖЕ HTTP-КЕШ

точка банк

- Нужна поддержка заголовков кеширования на сервере

ЕСТЬ ЖЕ HTTP-КЕШ

- Нужна поддержка заголовков кеширования на сервере
- Мы им не управляем

ЕСТЬ ЖЕ HTTP-КЕШ

точка банк

- Нужна поддержка заголовков кеширования на сервере
- Мы им не управляем
- Не понимаем, насколько он эффективен

ЕСТЬ ЖЕ HTTP-КЕШ

- Нужна поддержка заголовков кеширования на сервере
- Мы им не управляем
- Не понимаем, насколько он эффективен
- Ограниченные механизмы инвалидации — только Etag и TTL

ЕСТЬ ЖЕ HTTP-КЕШ

- Нужна поддержка заголовков кеширования на сервере
- Мы им не управляем
- Не понимаем, насколько он эффективен
- Ограниченные механизмы инвалидации — только Etag и TTL
- Неудобно, если у нас не RESTfull

А ЧЕГО НЕ TANSTACK?

точка банк

А ЧЕГО НЕ TANSTACK?

точка банк

- Всех на него пересадишь

А ЧЕГО НЕ TANSTACK?

точка банк

- Всех на него пересадишь
- Всё равно нужен будет пакет, который ещё и обновлять надо везде

А ЧЕГО НЕ TANSTACK?

точка банк

- Всех на него пересадишь
- Всё равно нужен будет пакет, который ещё и обновлять надо везде
- Живёт в основном потоке

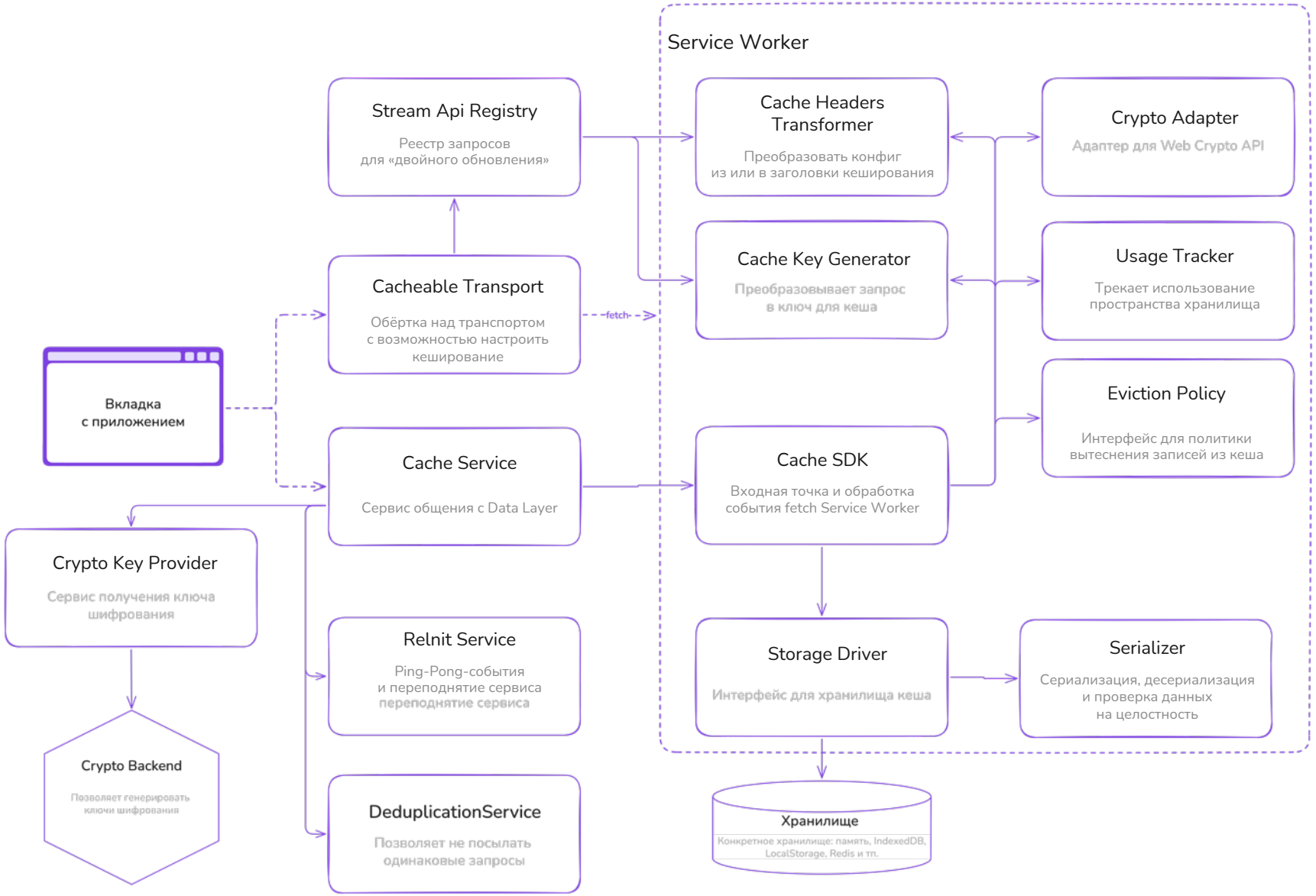
А ЧЕГО НЕ TANSTACK?

точка банк

- Всех на него пересадишь
- Всё равно нужен будет пакет, который ещё и обновлять надо везде
- Живёт в основном потоке
- Не умеет в несколько вкладок

СОВСЕМ БОЛЬШОЙ ВЫМОХАЛ!

точка банк



ПОПРОБУЙТЕ САМИ

точка банк

ПОПРОБУЙТЕ САМИ

точка банк

```
import { CacheService } from '@ft-packages/team_wtf-cache';

serviceWorker.register('./service.worker.js').then(() => {
  const dataLayer = new CacheService();

  dataLayer.init(cryptoKey, { userId })
    .catch((error: Error) => {
      console.error('DataLayer error', error);
    });
})
```

ПОПРОБУЙТЕ САМИ

точка банк

```
import { CacheService } from '@ft-packages/team_wtf-cache';

serviceWorker.register('./service.worker.js').then(() => {
  const dataLayer = new CacheService();

  dataLayer.init(cryptoKey, { userId })
    .catch((error: Error) => {
      console.error('DataLayer error', error);
    });
})
```

```
import { CacheHeadersTransformer } from '@ft-packages/team_wtf-cache';

const doRequest = () => {
  return fetch('https://api.example.com', {
    headers: CacheHeadersTransformer.getHeadersFromConfig({
      isCacheEnabled: true,
      ttl: '1d'
    })
  });
}
```

точка банк

Даниил Рублёв
TG: @spunky_dr | @coaled_frontender

СПАСИБО ЗА ВНИМАНИЕ!

Голосовалка тут

Все ссылки тут