Яндекс Такси

Микросервисы и балансеры

Кеши и C++

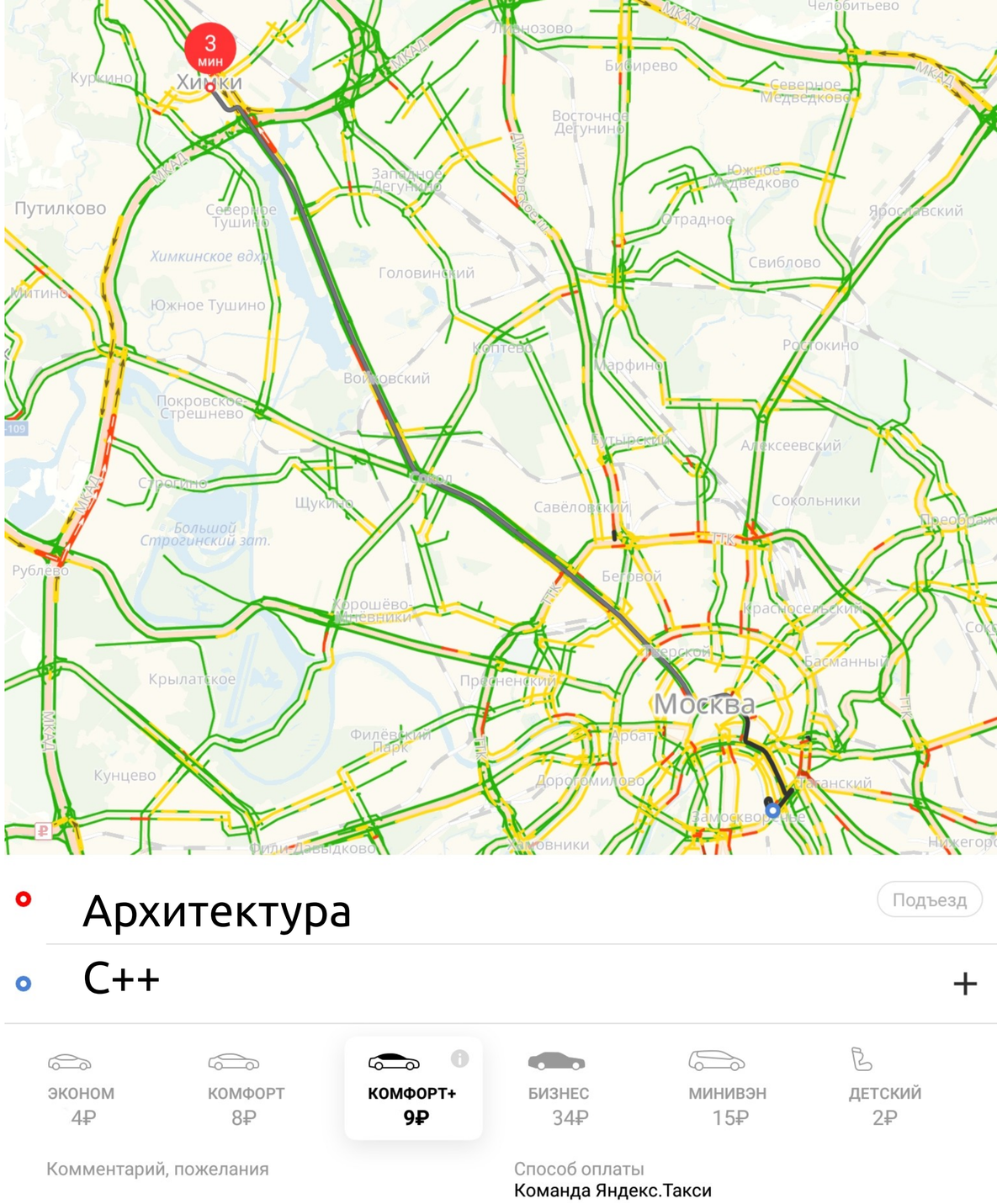
Полухин Антон

Antony Polukhin

Яндекс Такси

Содержание

- Архитектуры
  - Лучшая архитектура!
  - Монолит
  - Неправильный микросервис
  - Правильный микросервис
- Балансеры
  - Классика
  - Service Mesh
- Кеши и C++

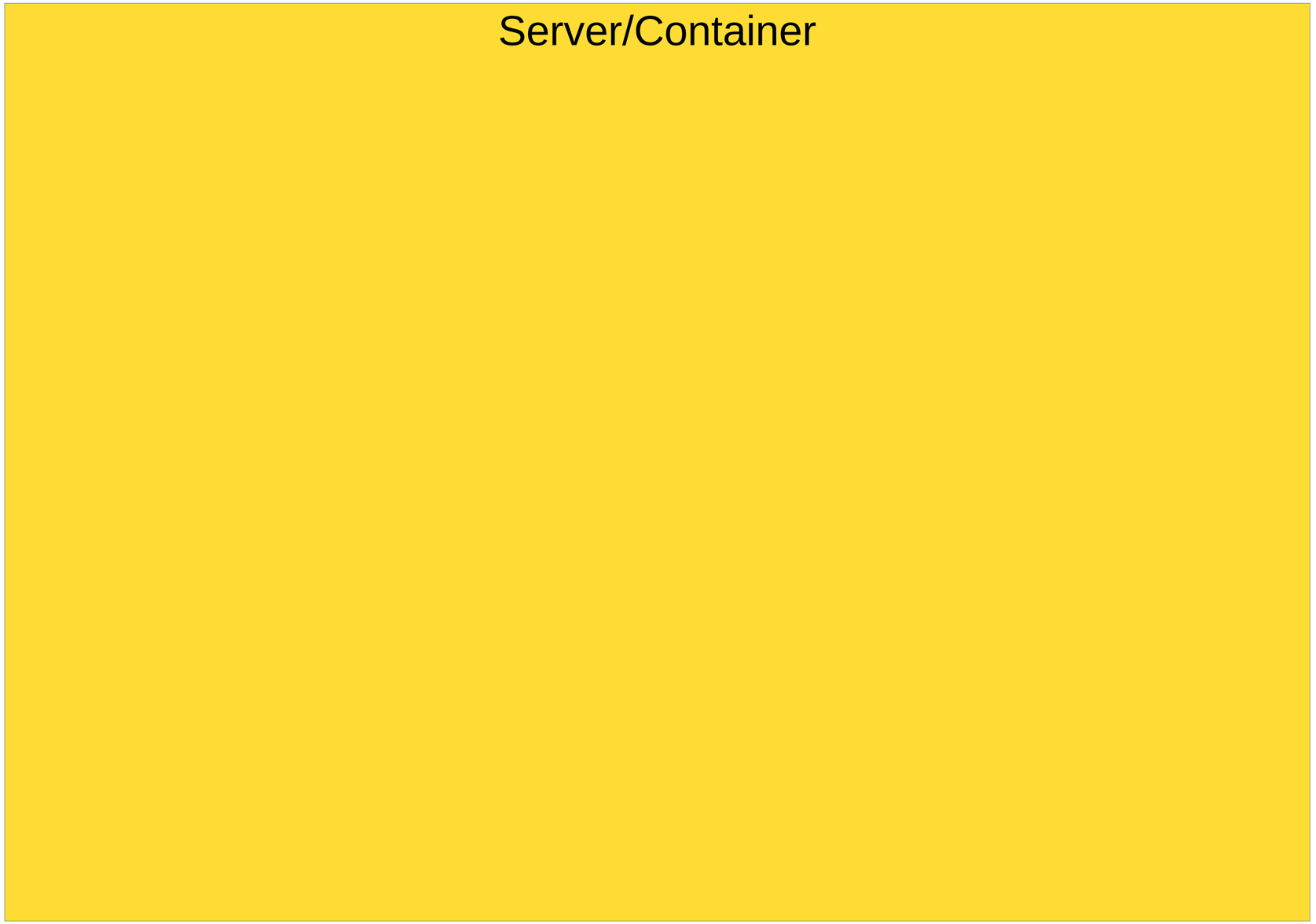


Самая лучшая архитектура это...

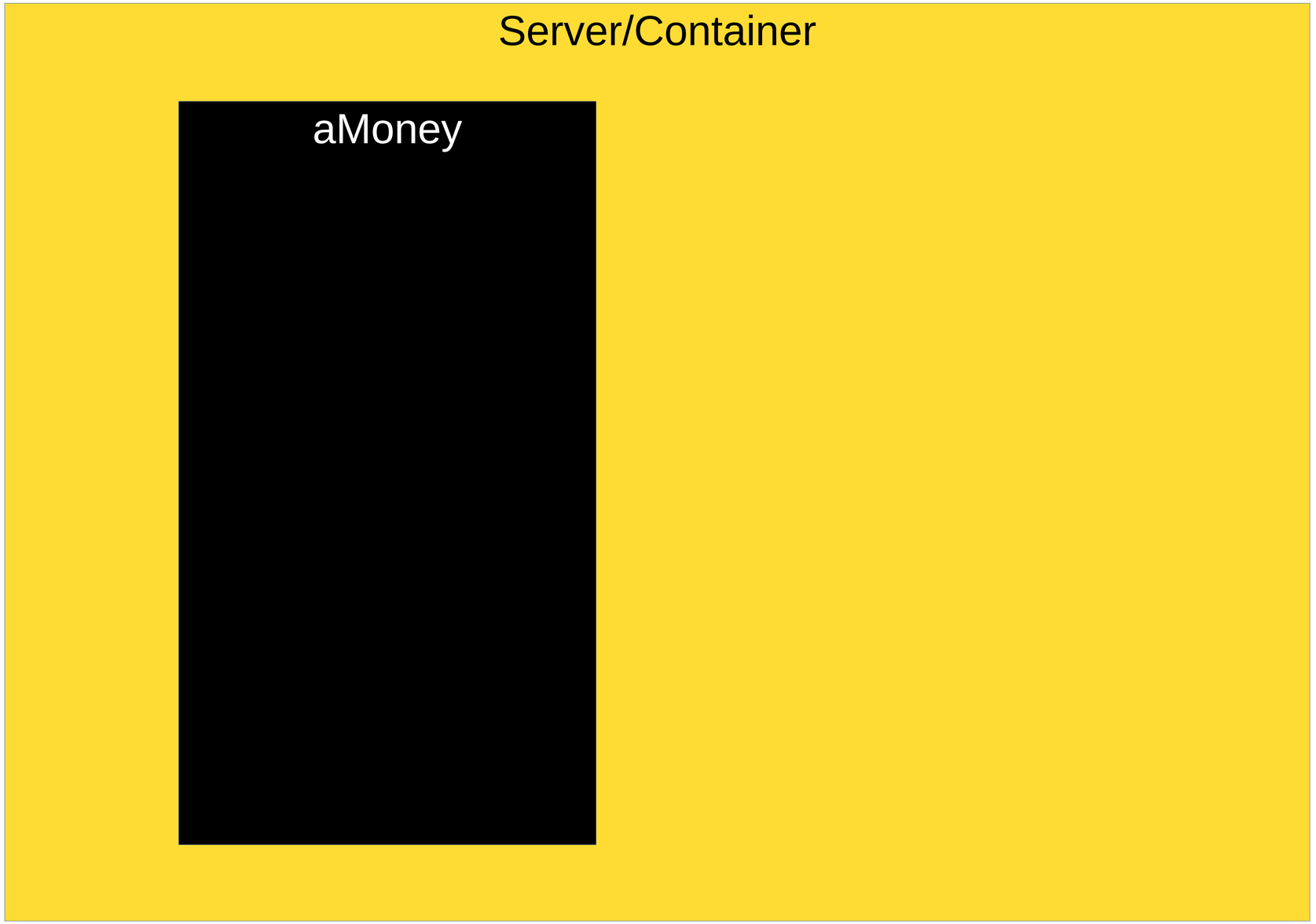
...та, которая ВАМ удобна!

Монолит

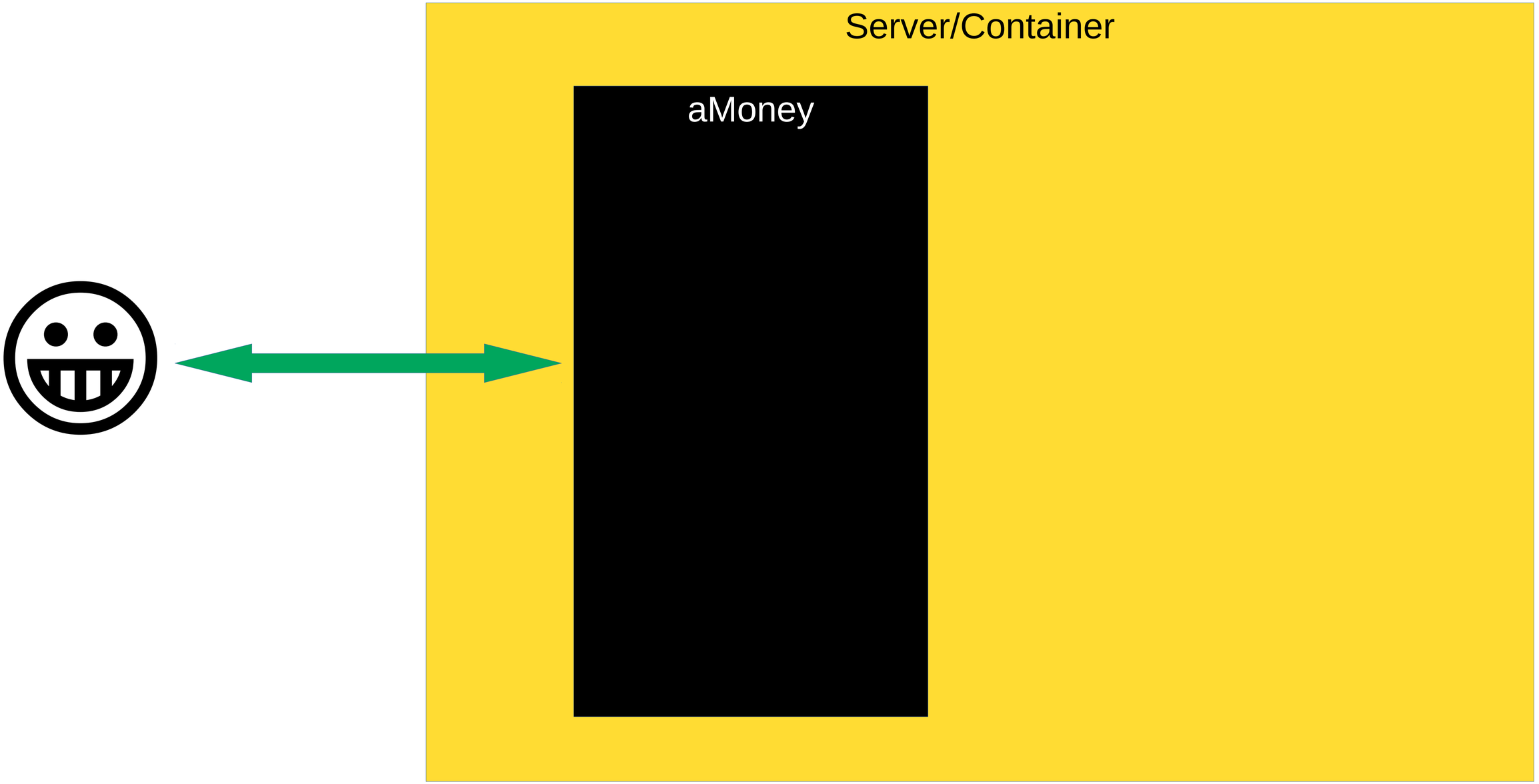
Монолит



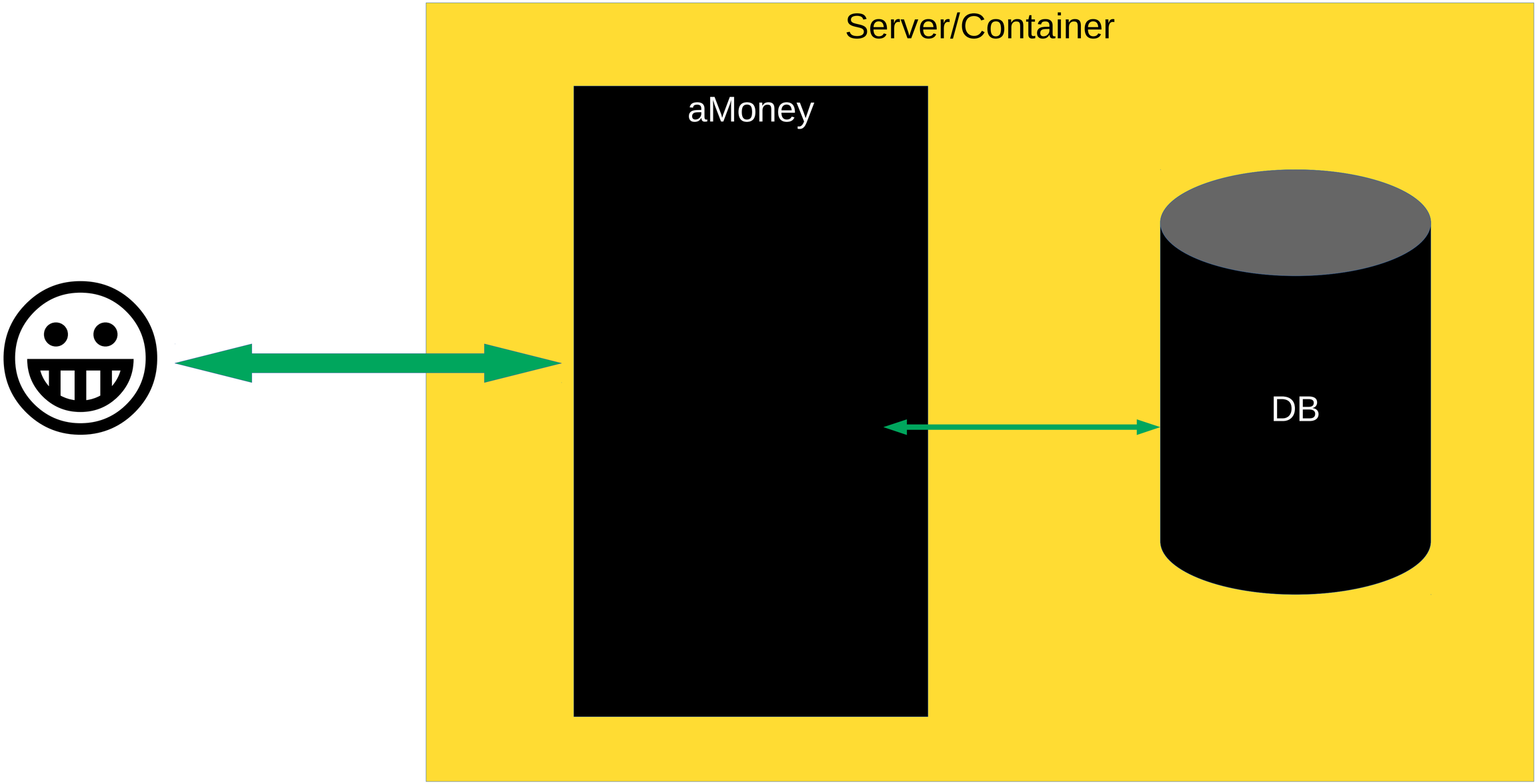
Монолит



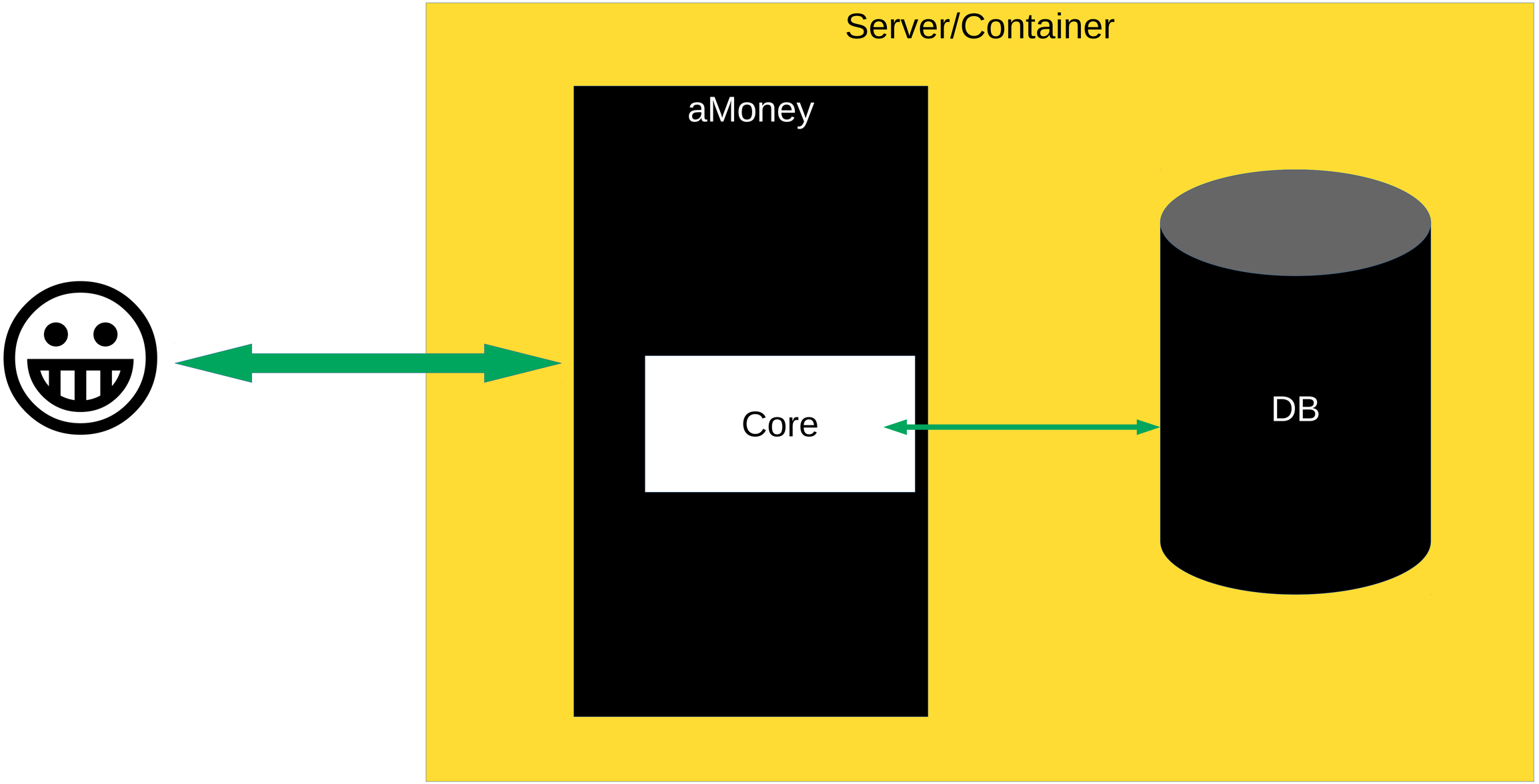
Монолит



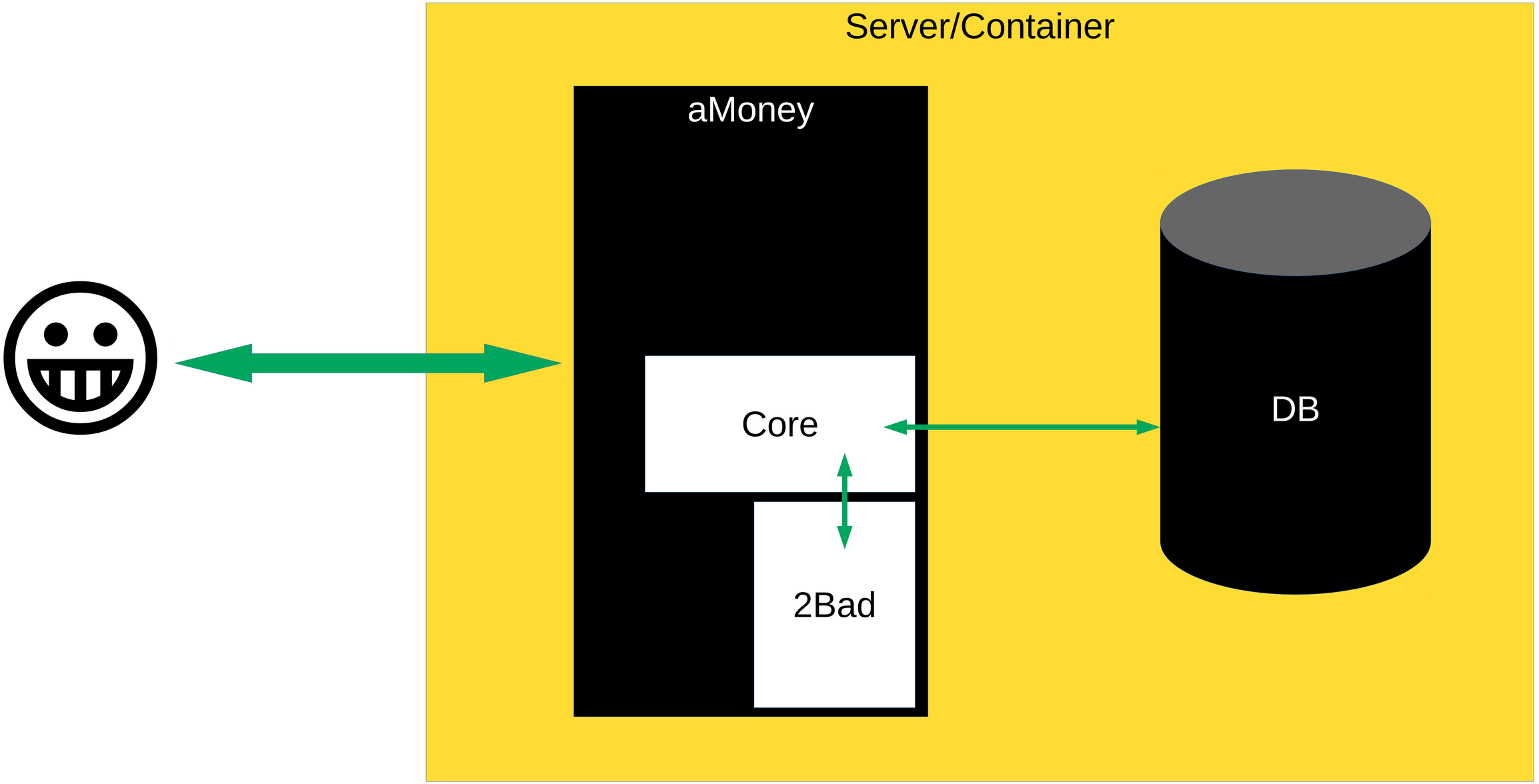
Монолит



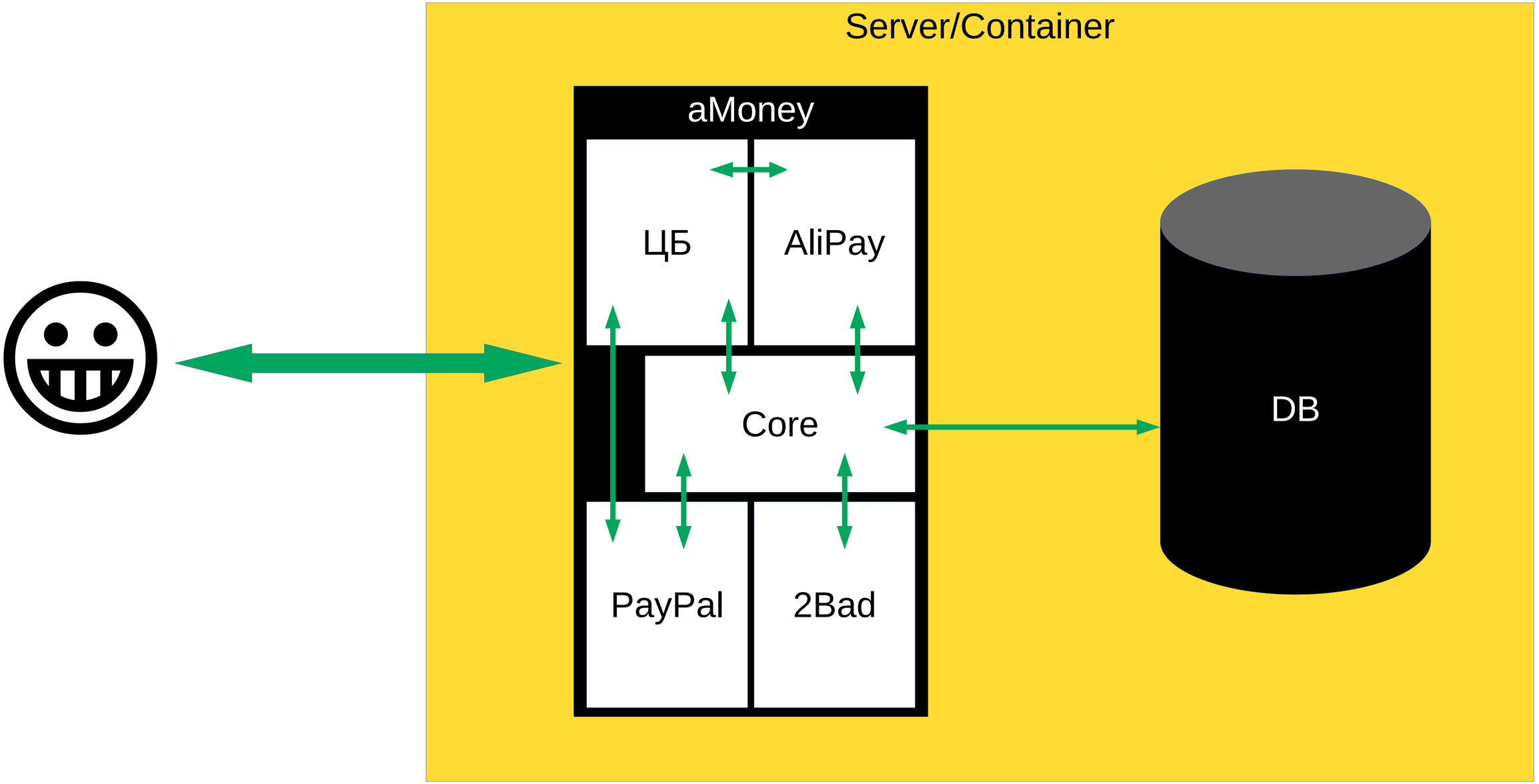
Монолит



Монолит



Монолит



Плюсы/минусы для небольшой команды

Плюсы/минусы для небольшой команды

Плюсы:

Плюсы/минусы для небольшой команды

Плюсы:

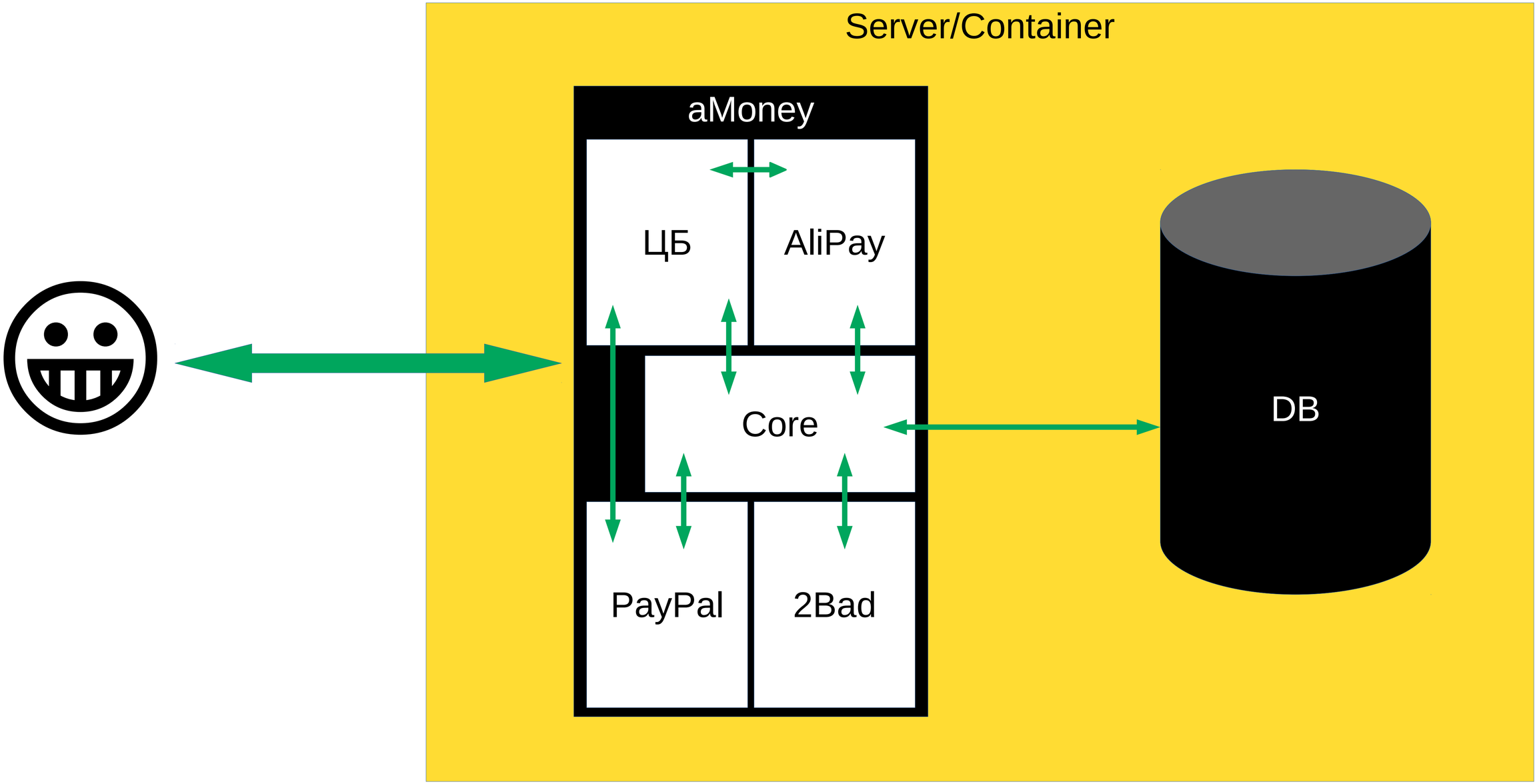
- Простой деплой

Плюсы/минусы для небольшой команды

Плюсы:

- Простой деплой
- Дешёвая передача данных между модулями

Монолит



Плюсы/минусы для небольшой команды

Плюсы:

- Простой деплой
- Дешёвая передача данных между модулями

Плюсы/минусы для небольшой команды

Плюсы:

- Простой деплой
- Дешёвая передача данных между модулями
- Тесное общение между разработчиками

Плюсы/минусы для небольшой команды

Плюсы:

- Простой деплой
- Дешёвая передача данных между модулями
- Тесное общение между разработчиками

Минусы:

Плюсы/минусы для небольшой команды

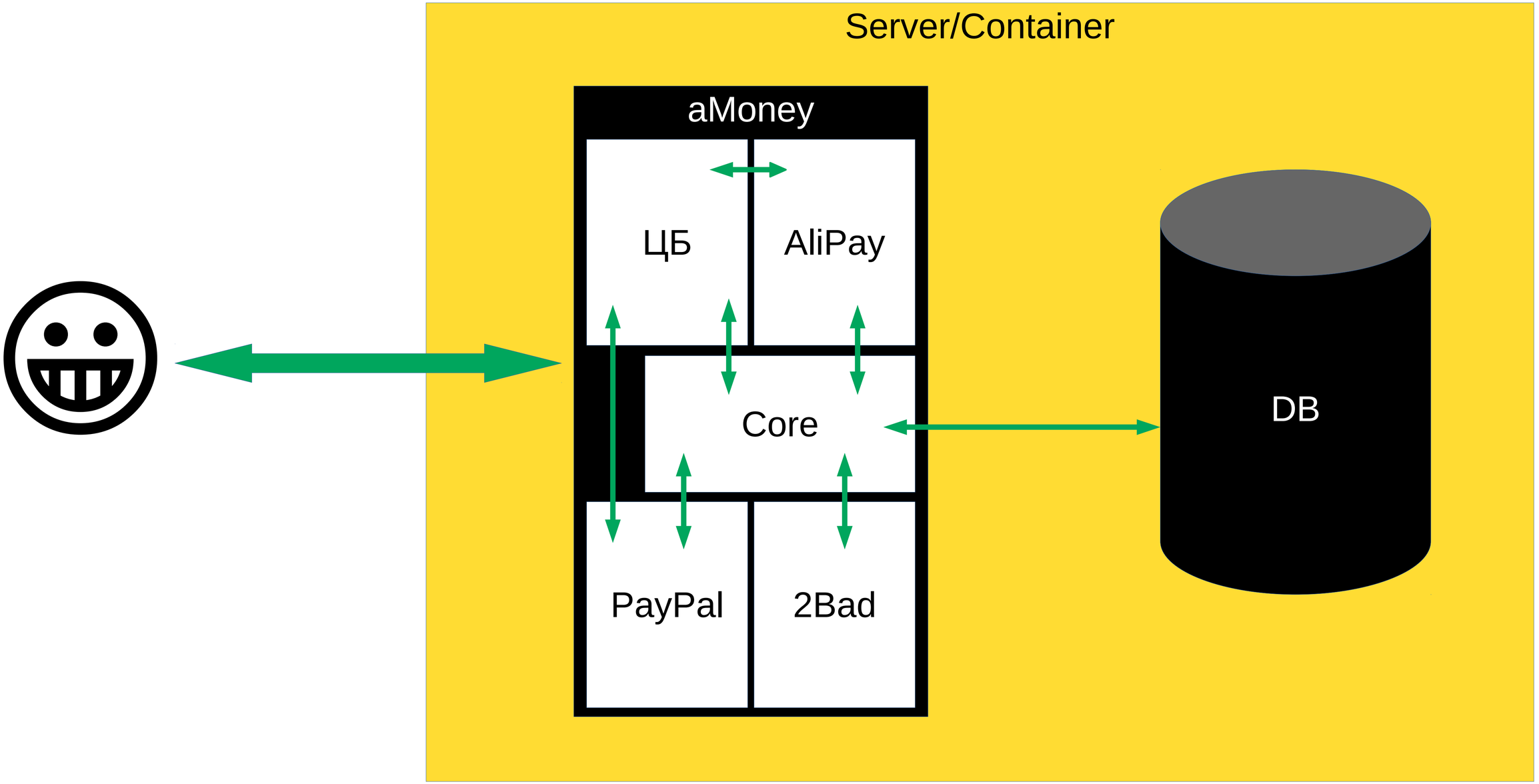
Плюсы:

- Простой деплой
- Дешёвая передача данных между модулями
- Тесное общение между разработчиками

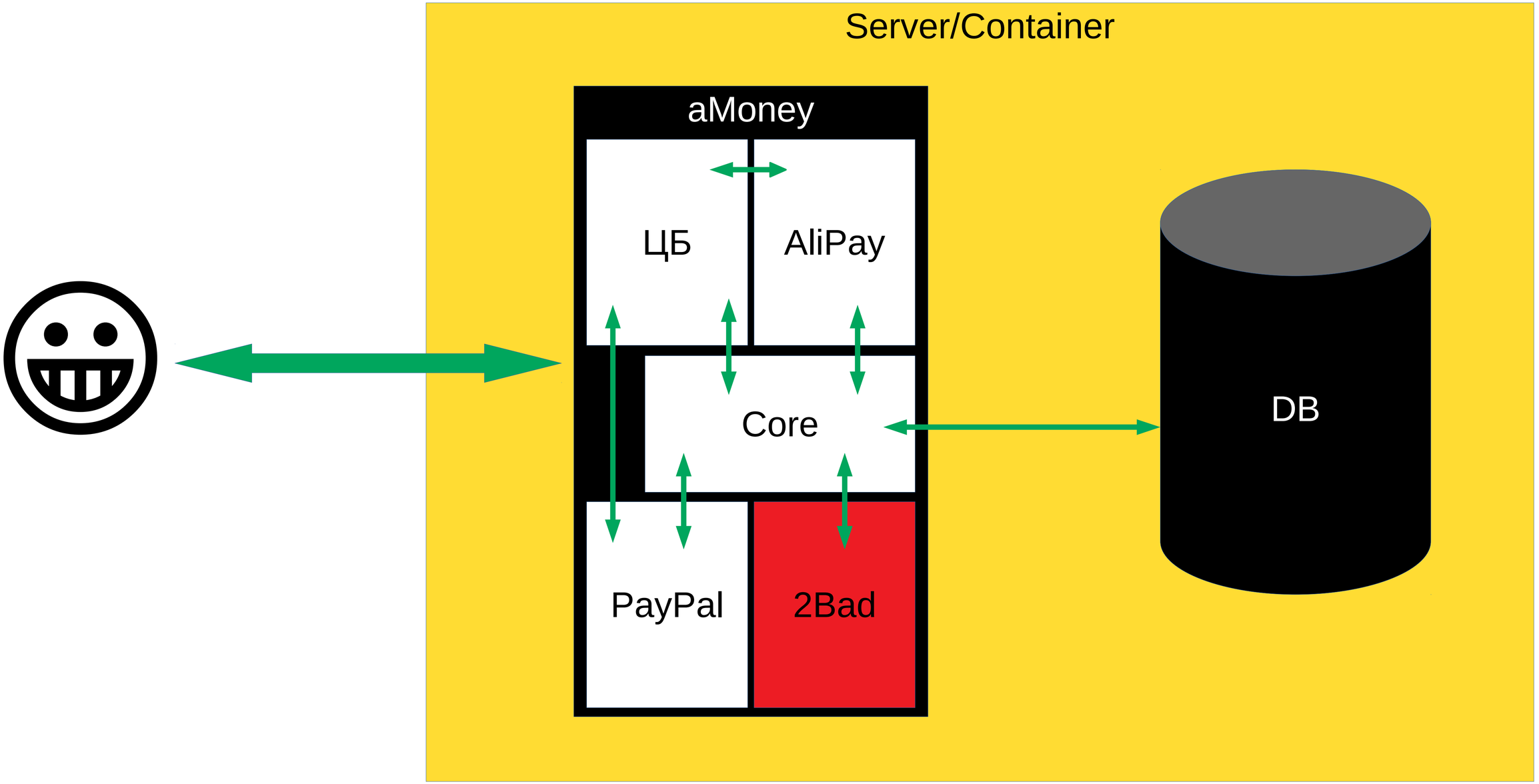
Минусы:

- Не идеальная надёжность

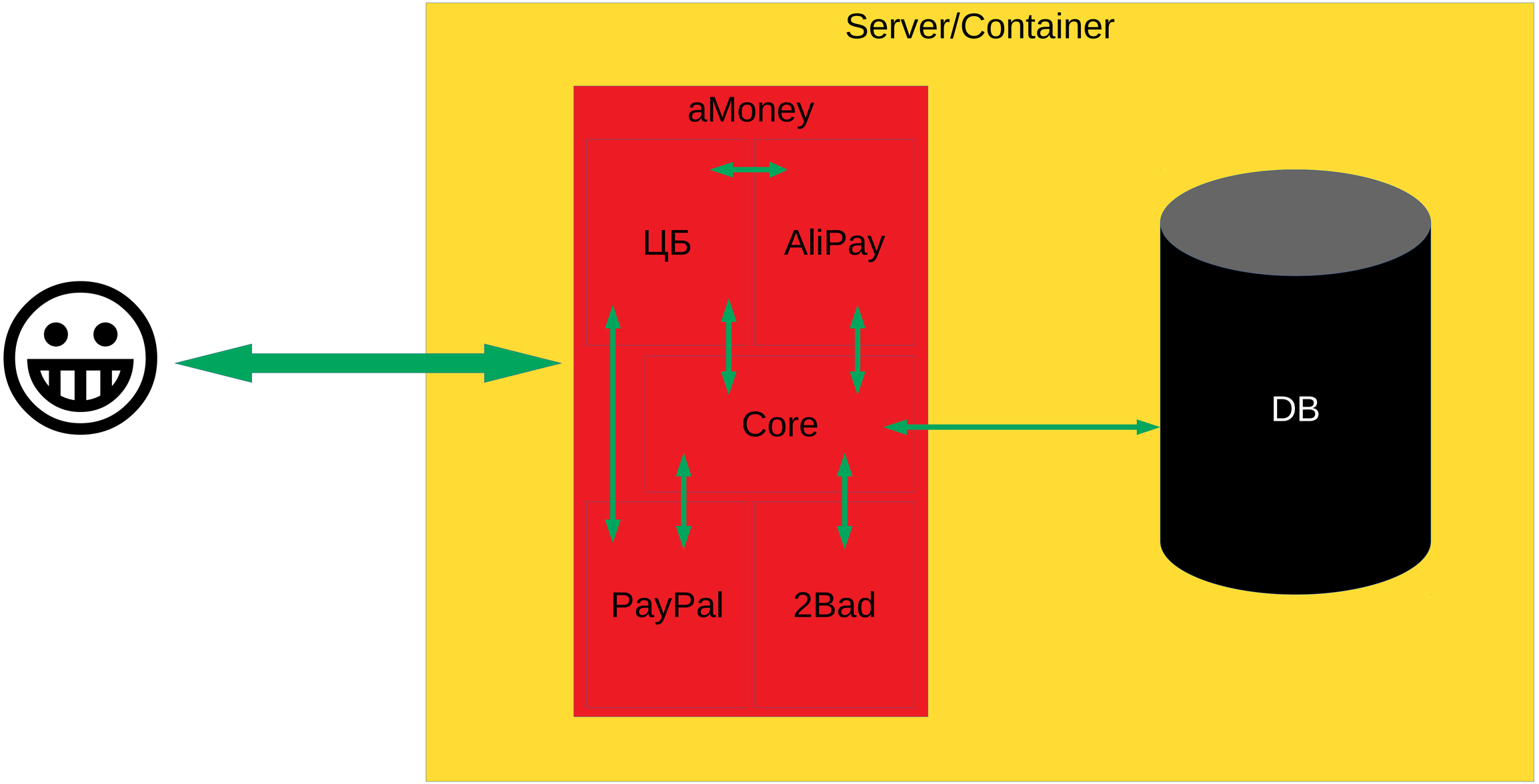
Монолит



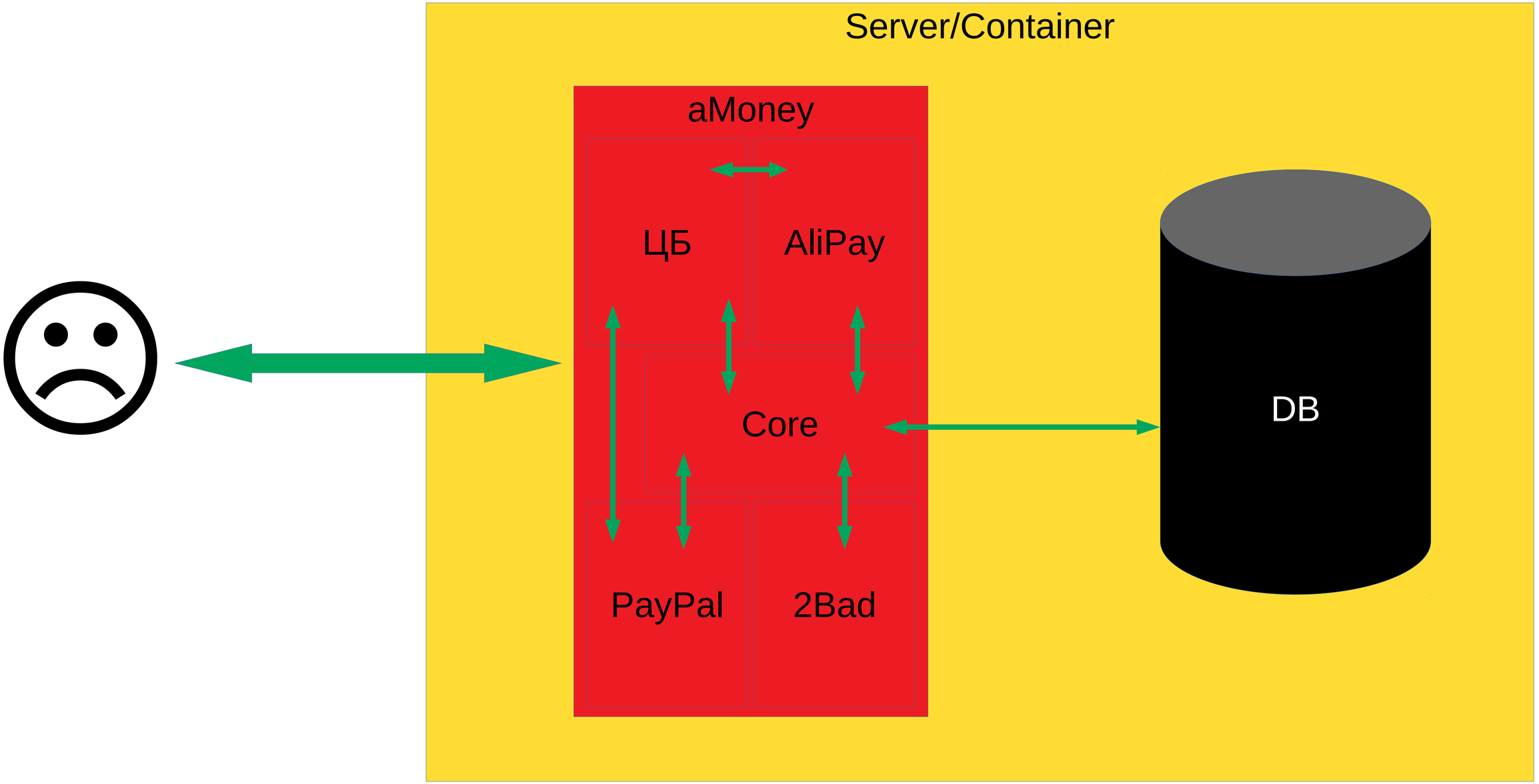
Монолит



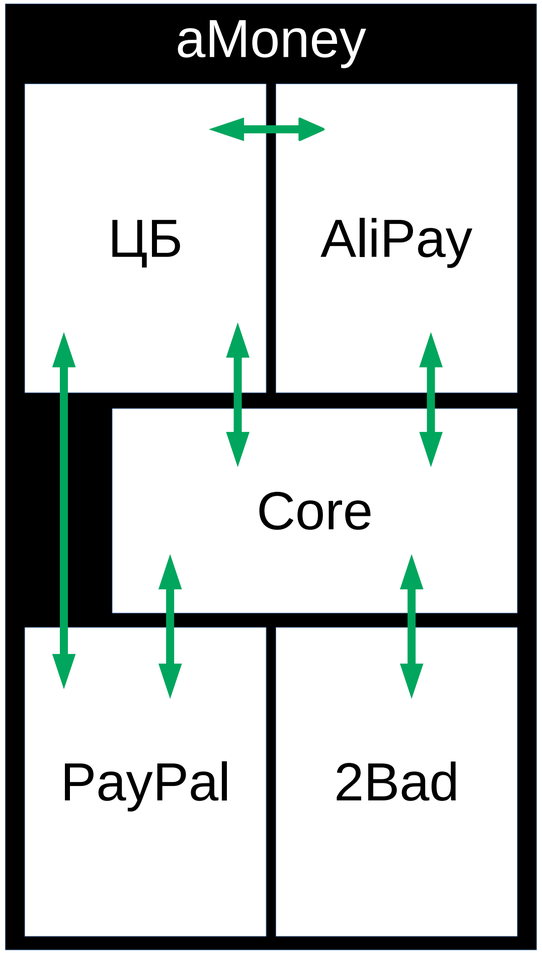
Монолит

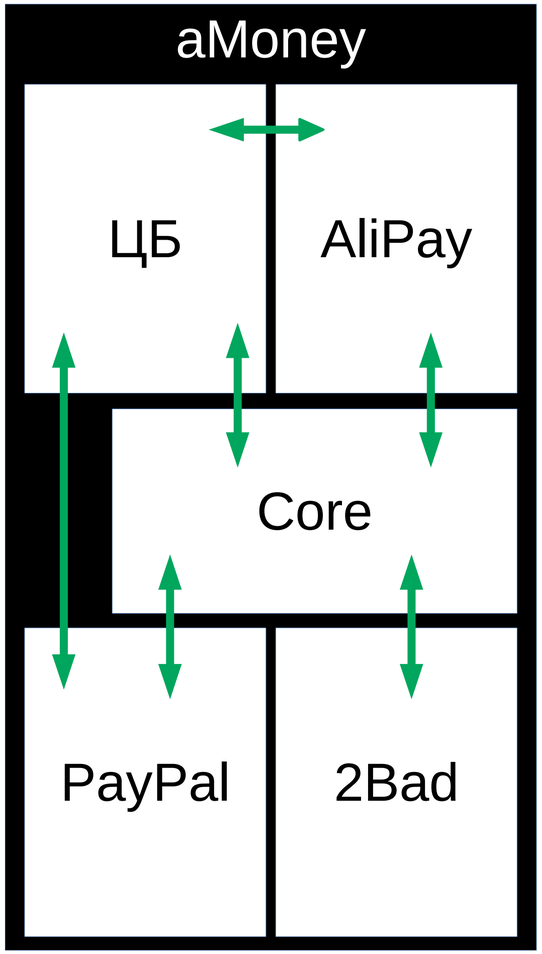


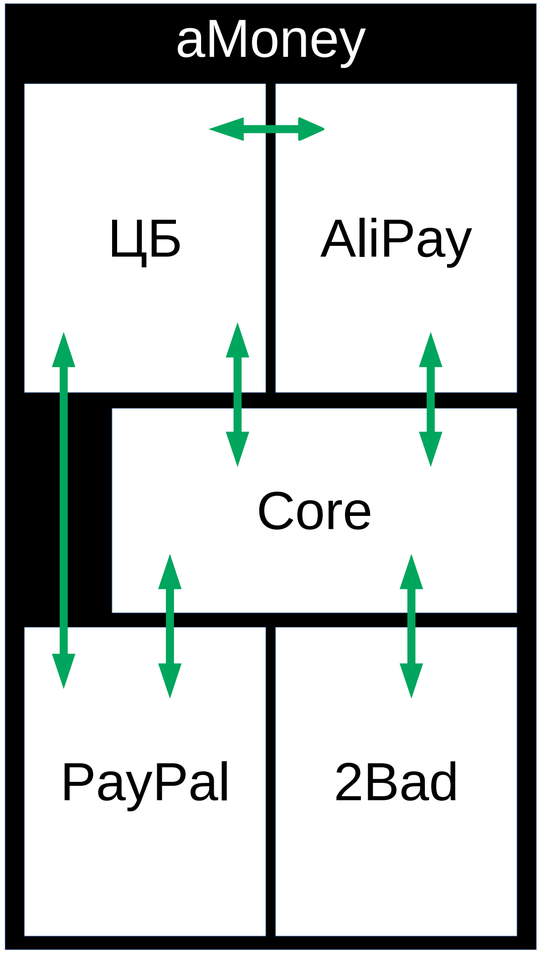
Монолит



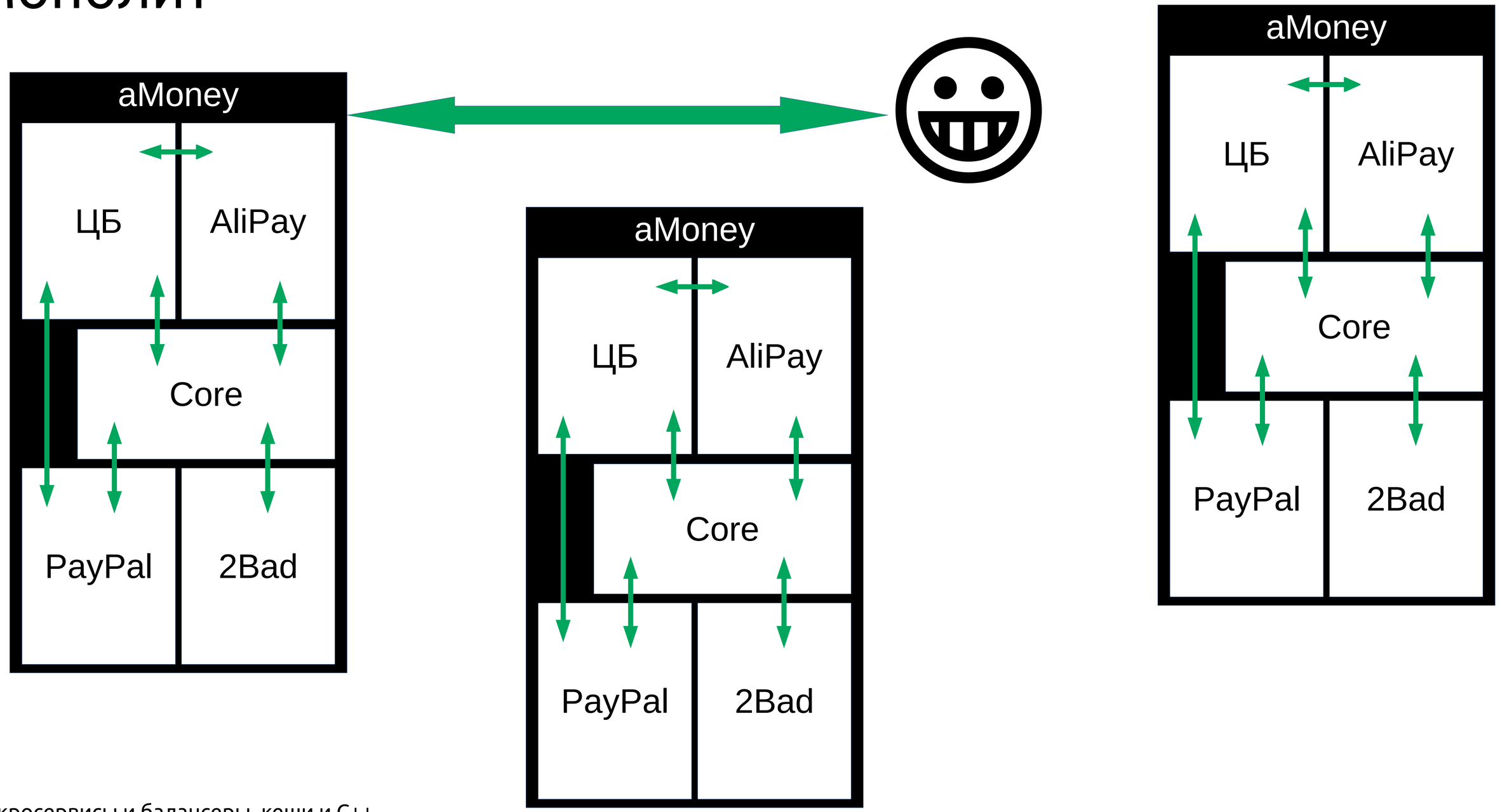
Монолит


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

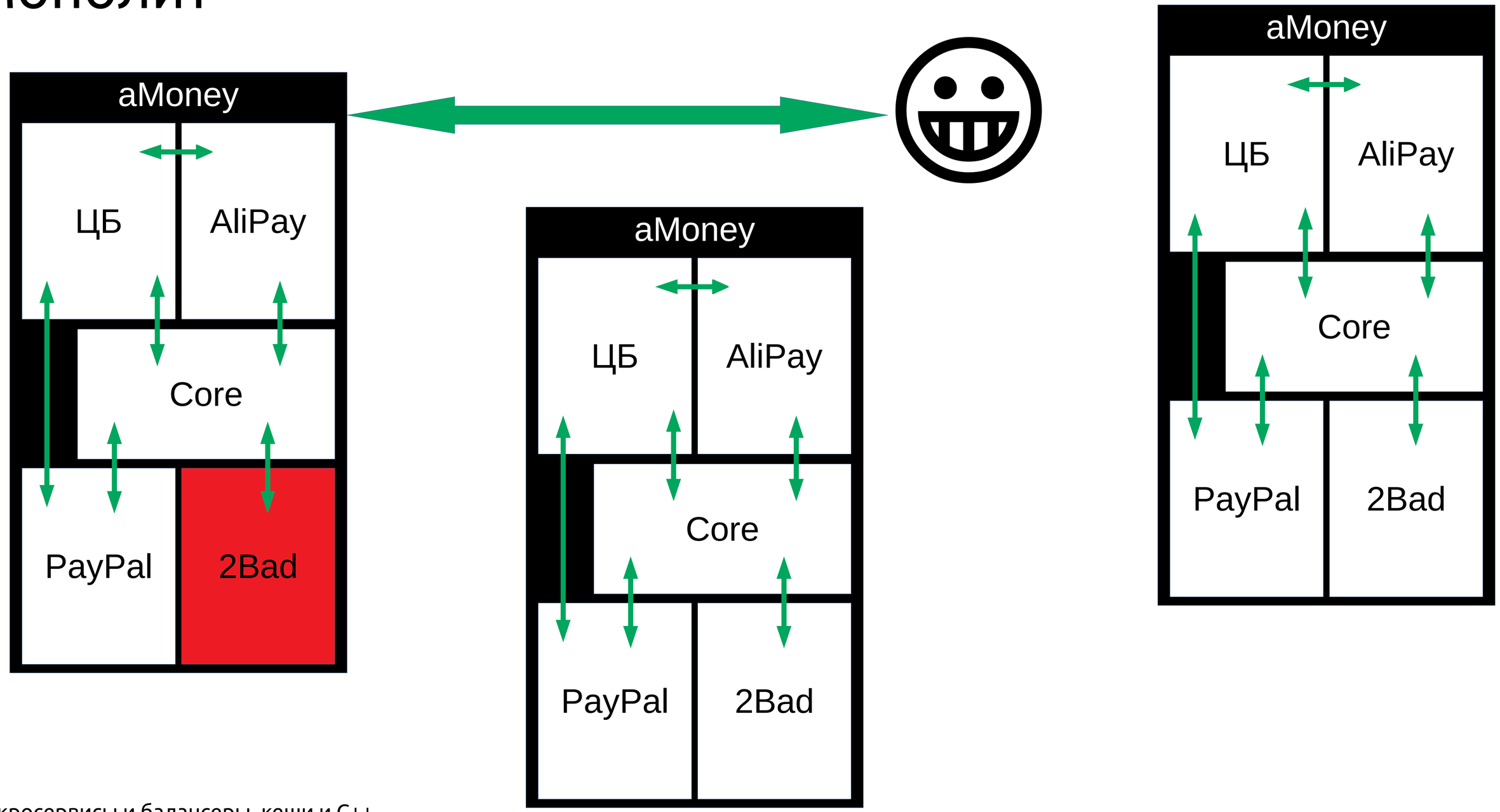

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

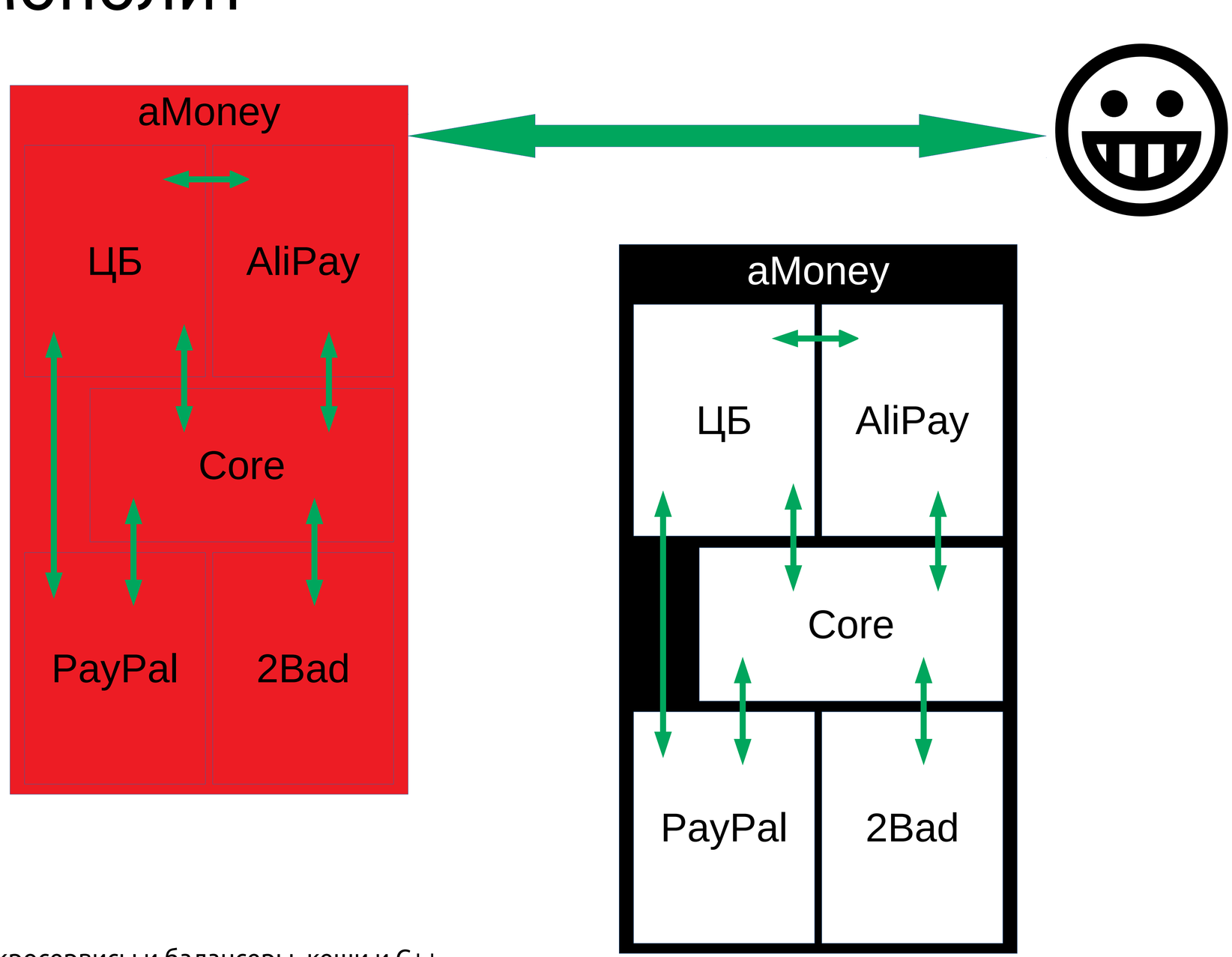
Монолит


aMoney
ЦБ
AliPay
Core
PayPal
2Bad
aMoney
ЦБ
AliPay
Core
PayPal
2Bad
aMoney
ЦБ
AliPay
Core
PayPal
2Bad

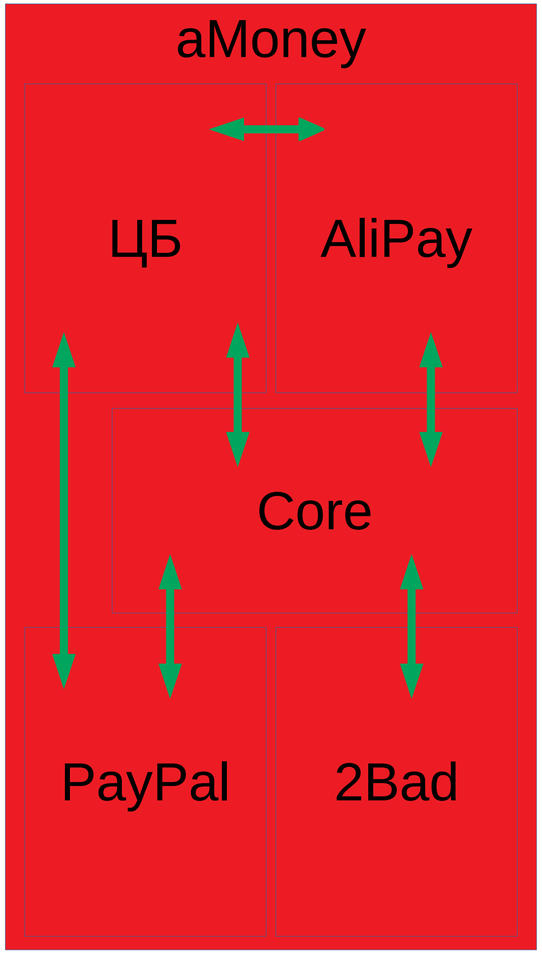
Монолит

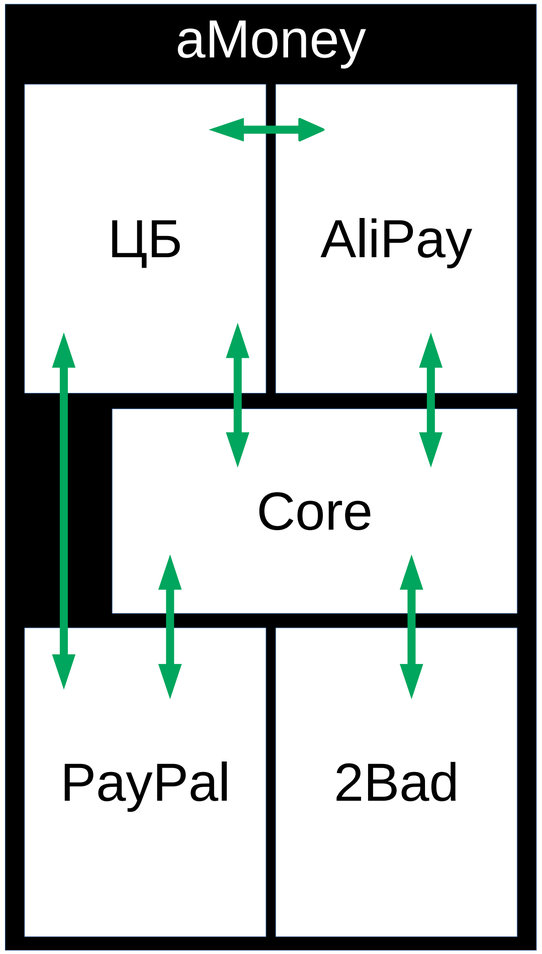

aMoney
ЦБ
AliPay
Core
PayPal
2Bad
aMoney
ЦБ
AliPay
Core
PayPal
2Bad
aMoney
ЦБ
AliPay
Core
PayPal
2Bad

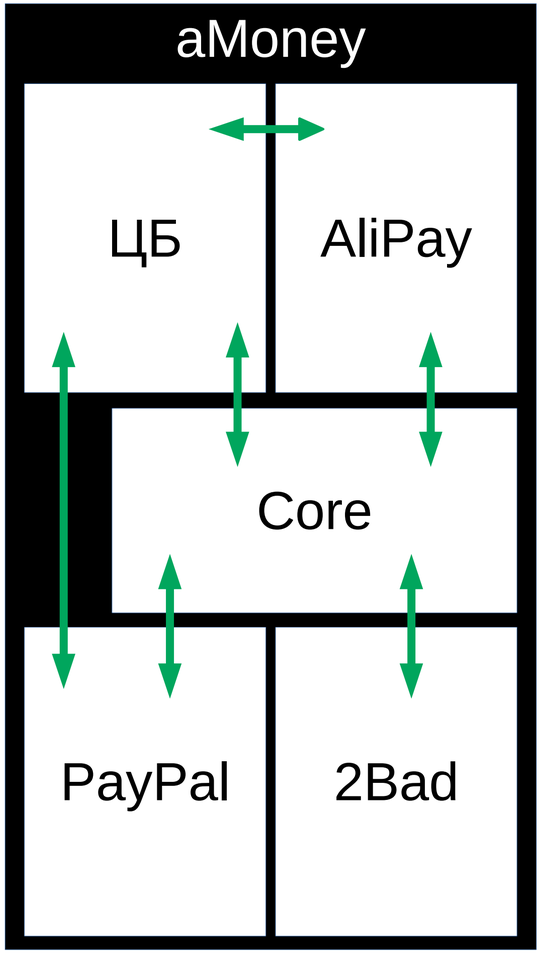
Монолит


aMoney
ЦБ
AliPay
Core
PayPal
2Bad
aMoney
ЦБ
AliPay
Core
PayPal
2Bad
aMoney
ЦБ
AliPay
Core
PayPal
2Bad

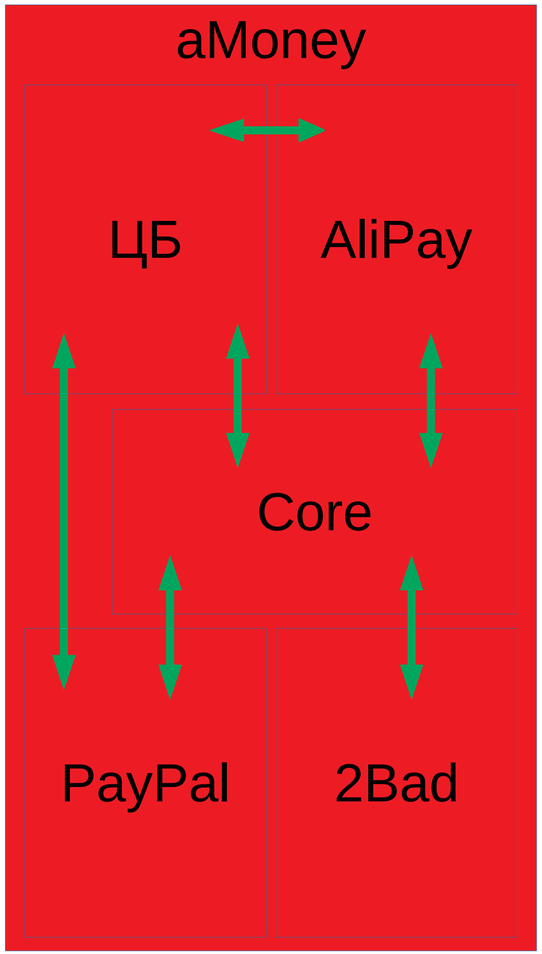
Монолит

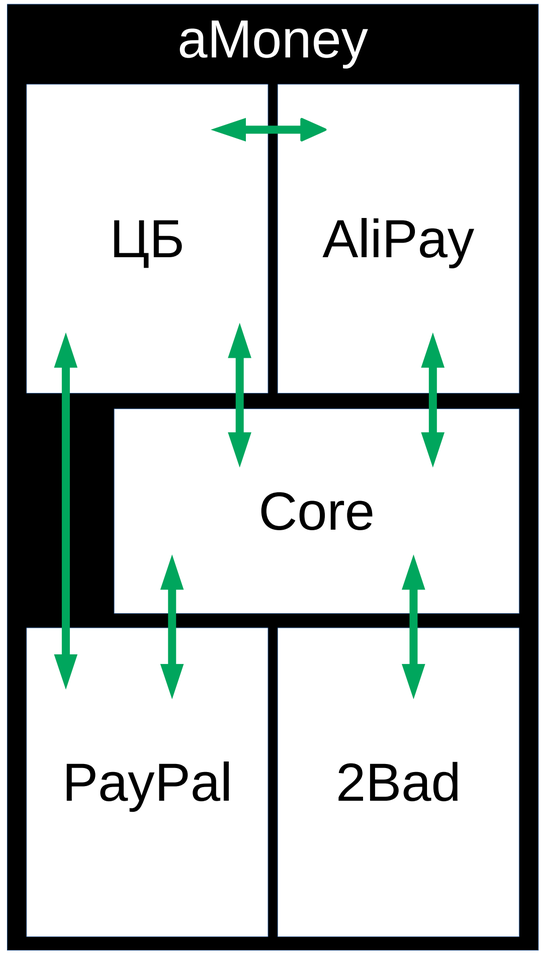

aMoney
ЦБ
AliPay
Core
PayPal
2Bad

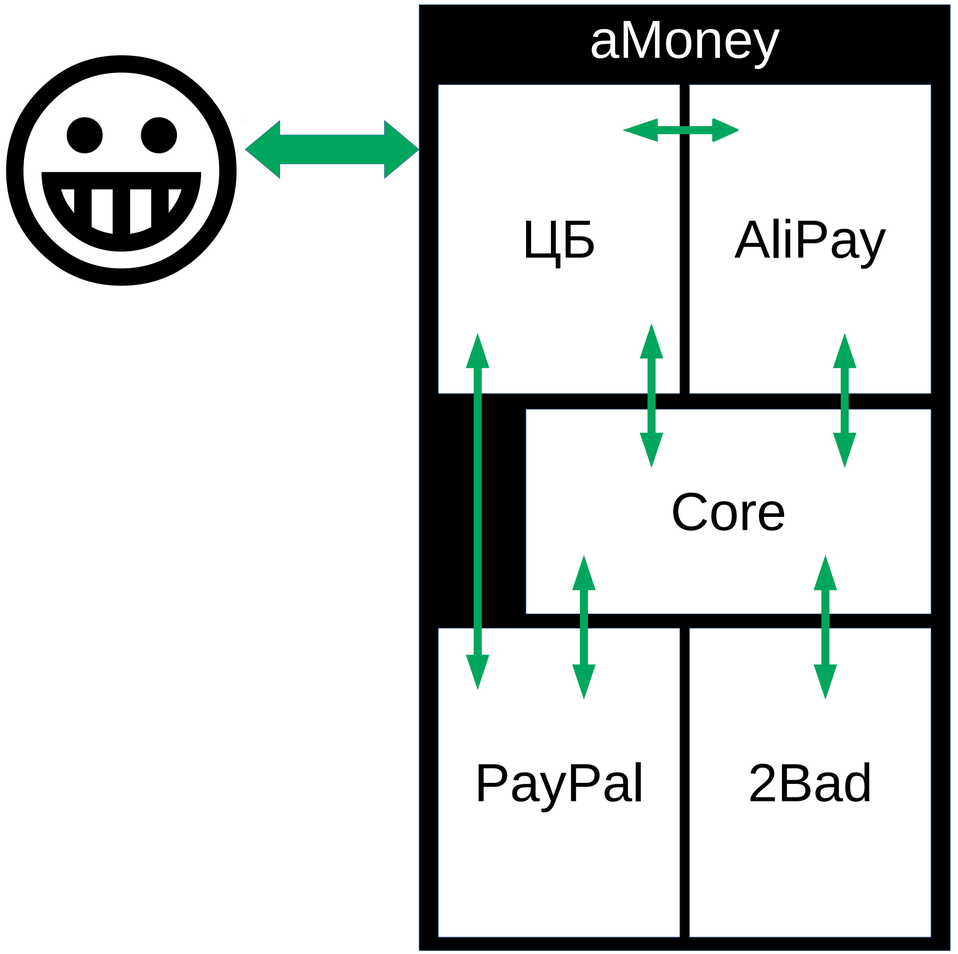

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

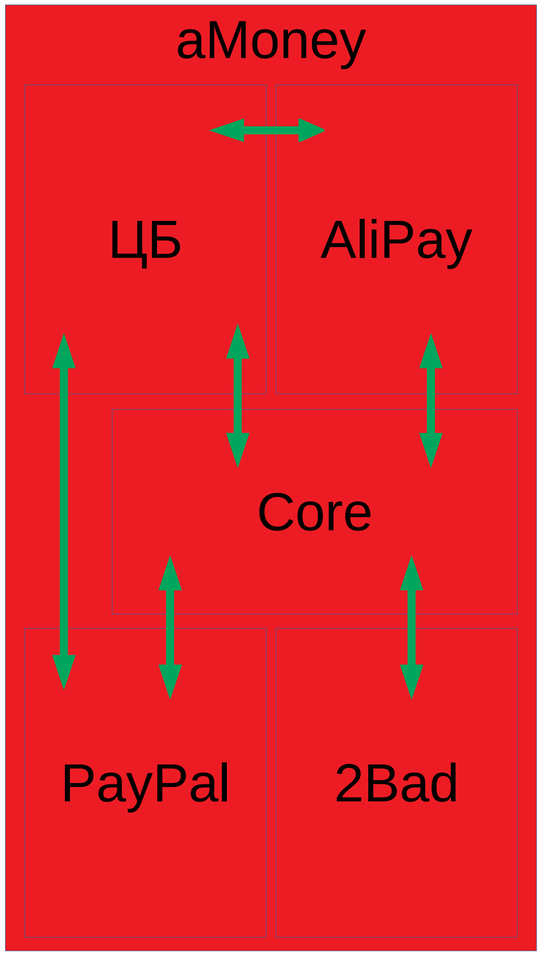
Монолит

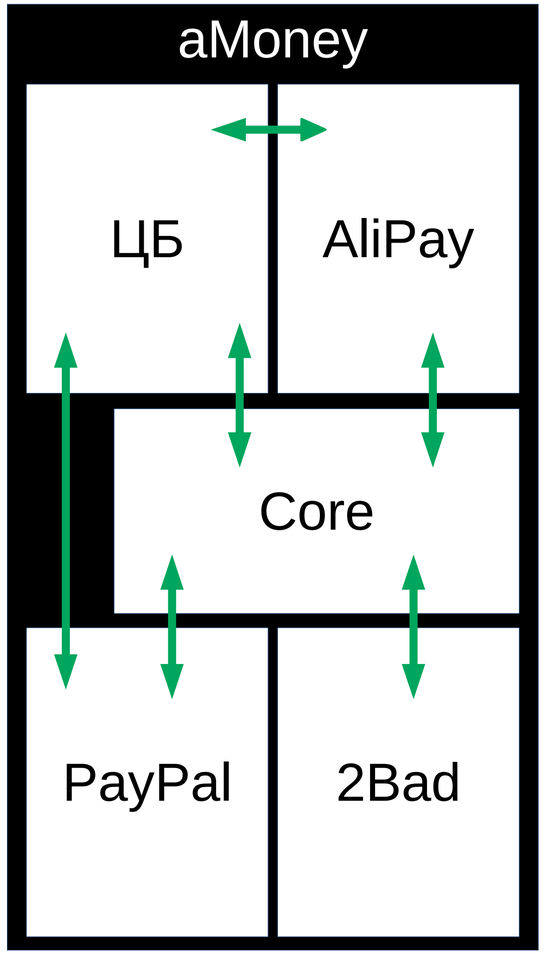

aMoney
ЦБ
AliPay
Core
PayPal
2Bad

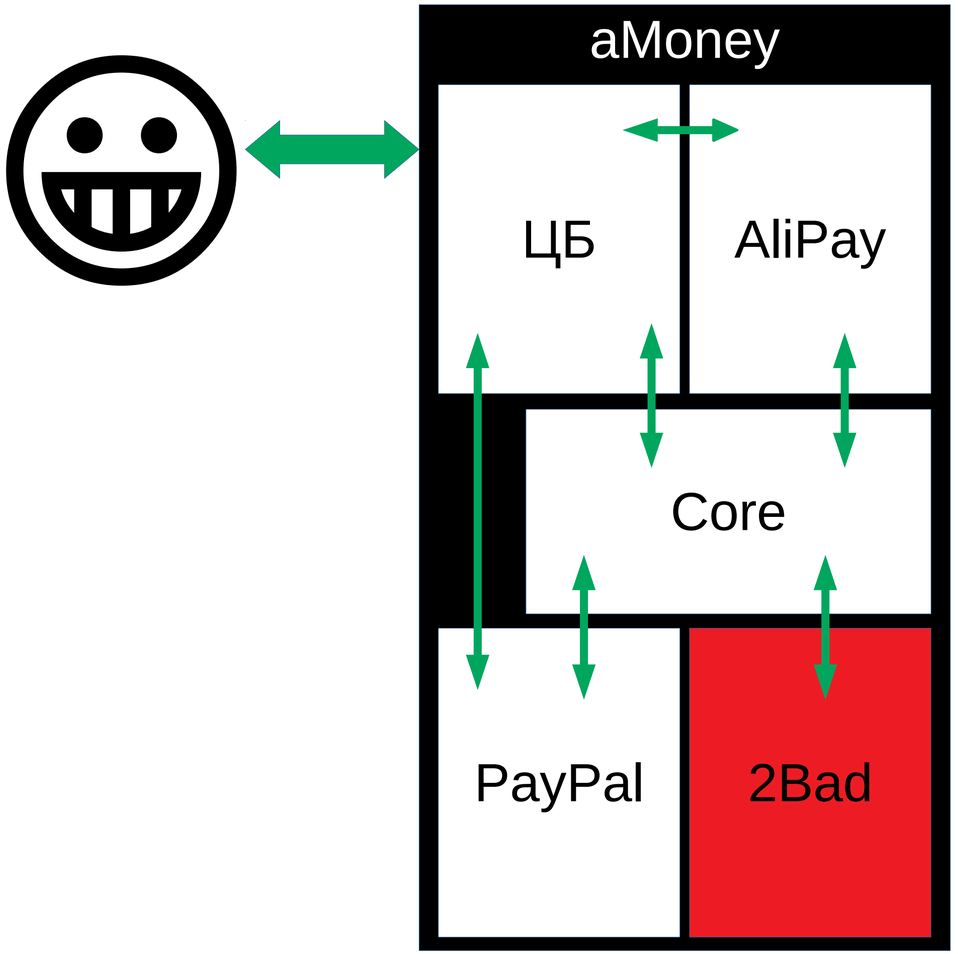

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

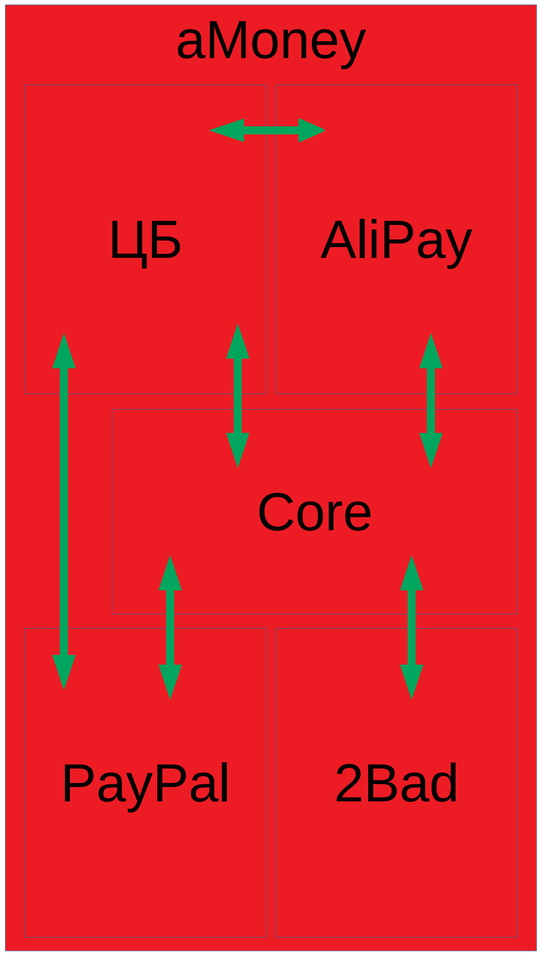
Монолит

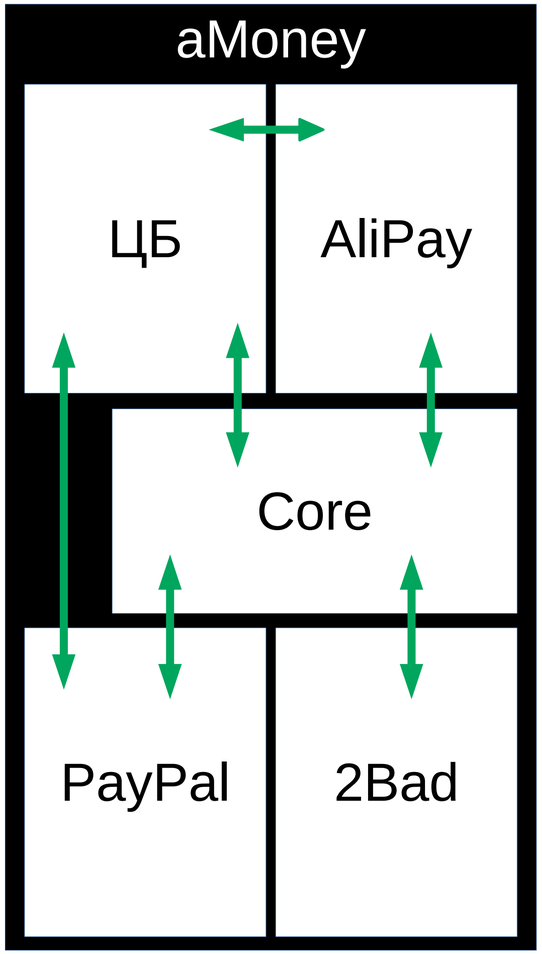

aMoney
ЦБ
AliPay
Core
PayPal
2Bad

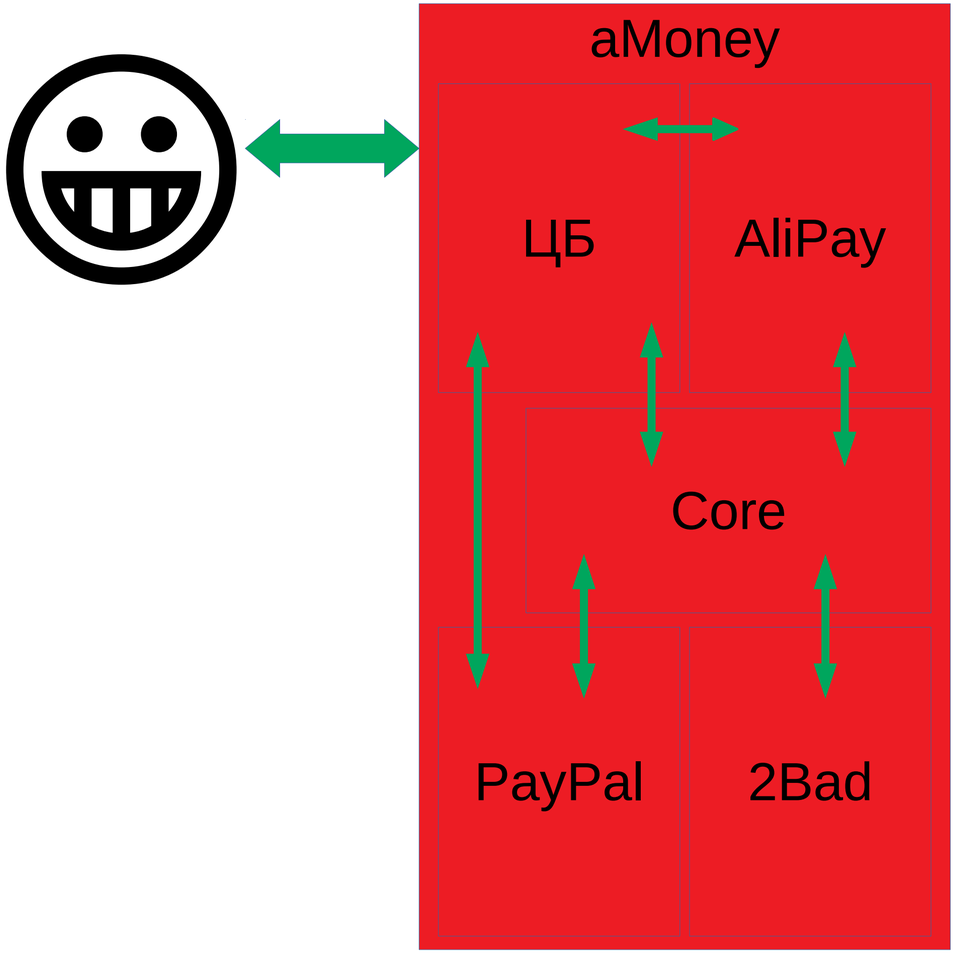

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

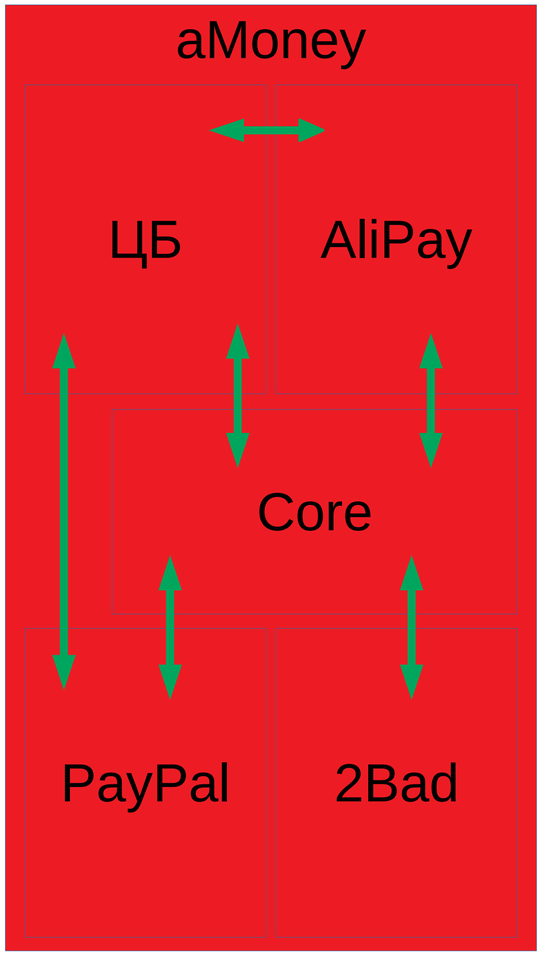
Монолит

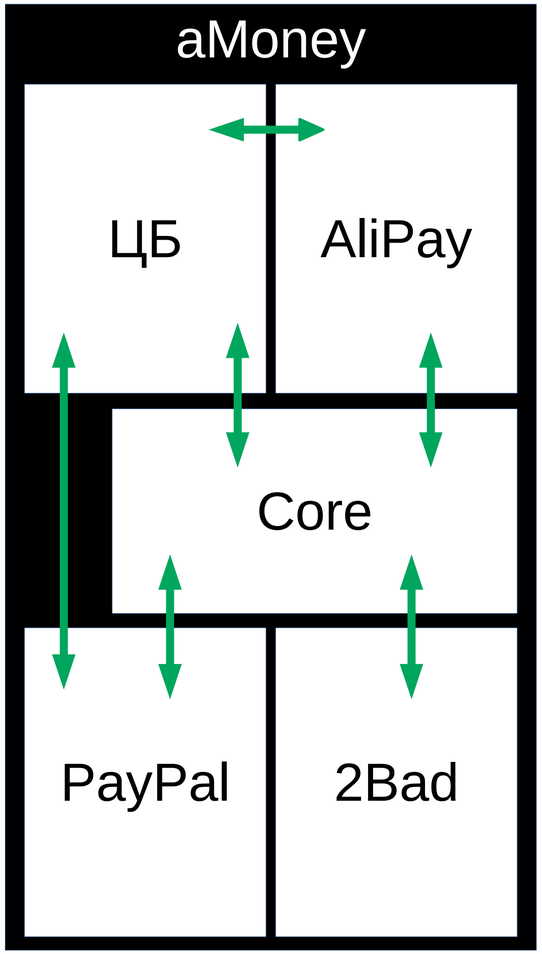

aMoney
ЦБ
AliPay
Core
PayPal
2Bad

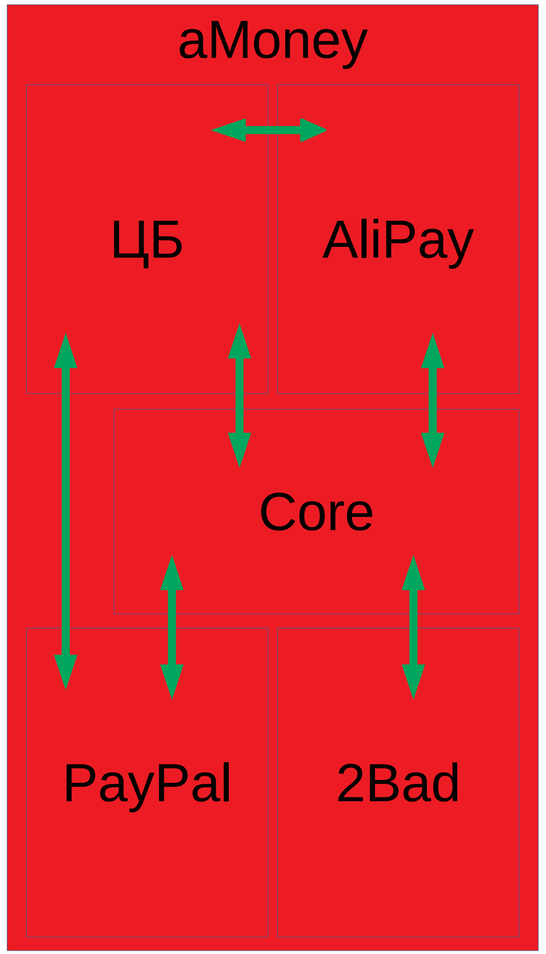

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

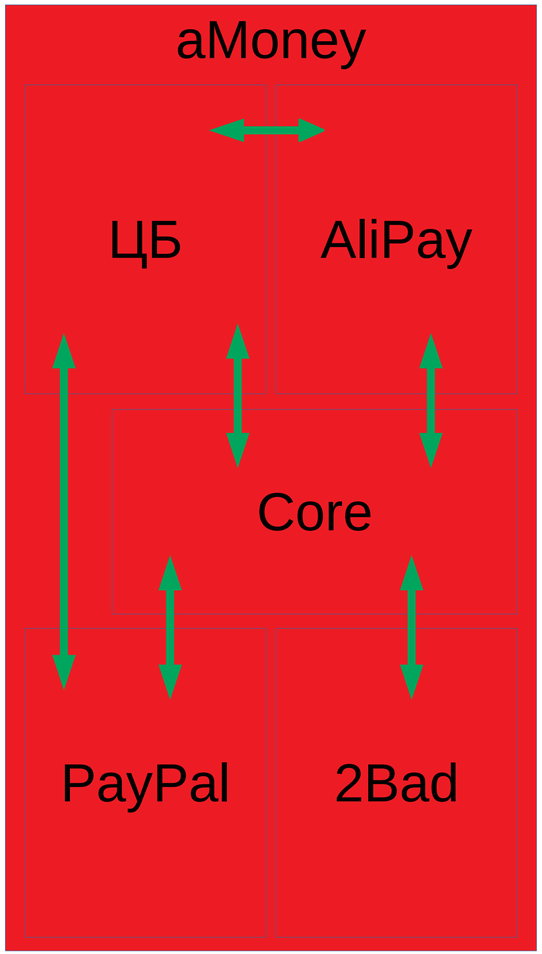
Монолит

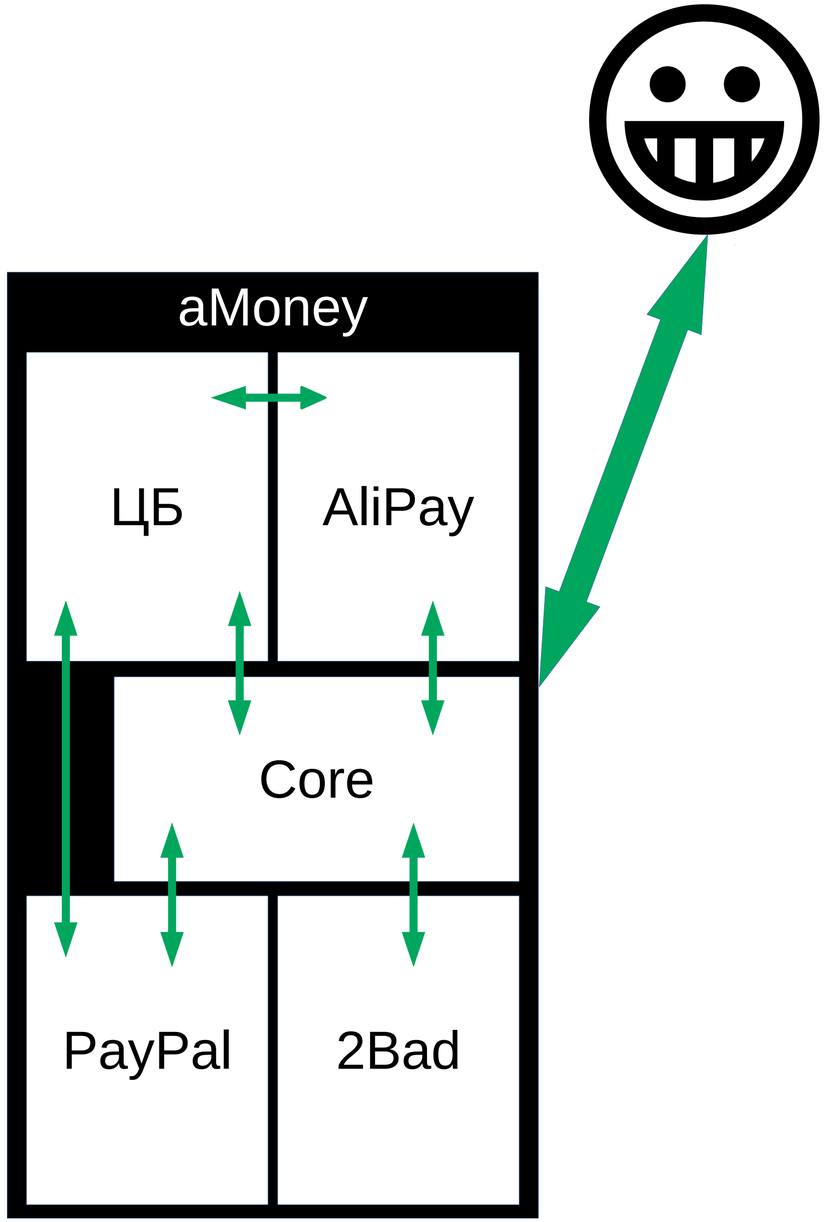

aMoney
ЦБ
AliPay
Core
PayPal
2Bad

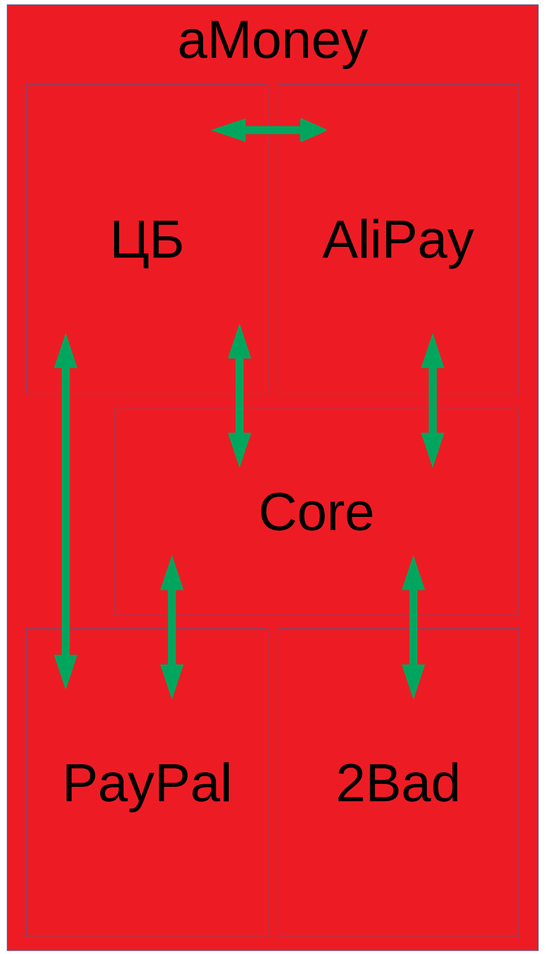

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

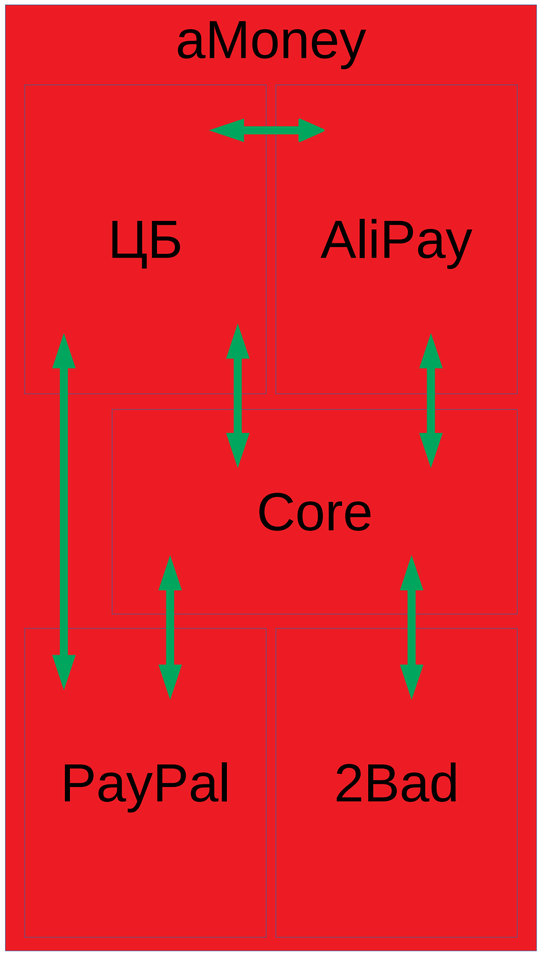
Монолит

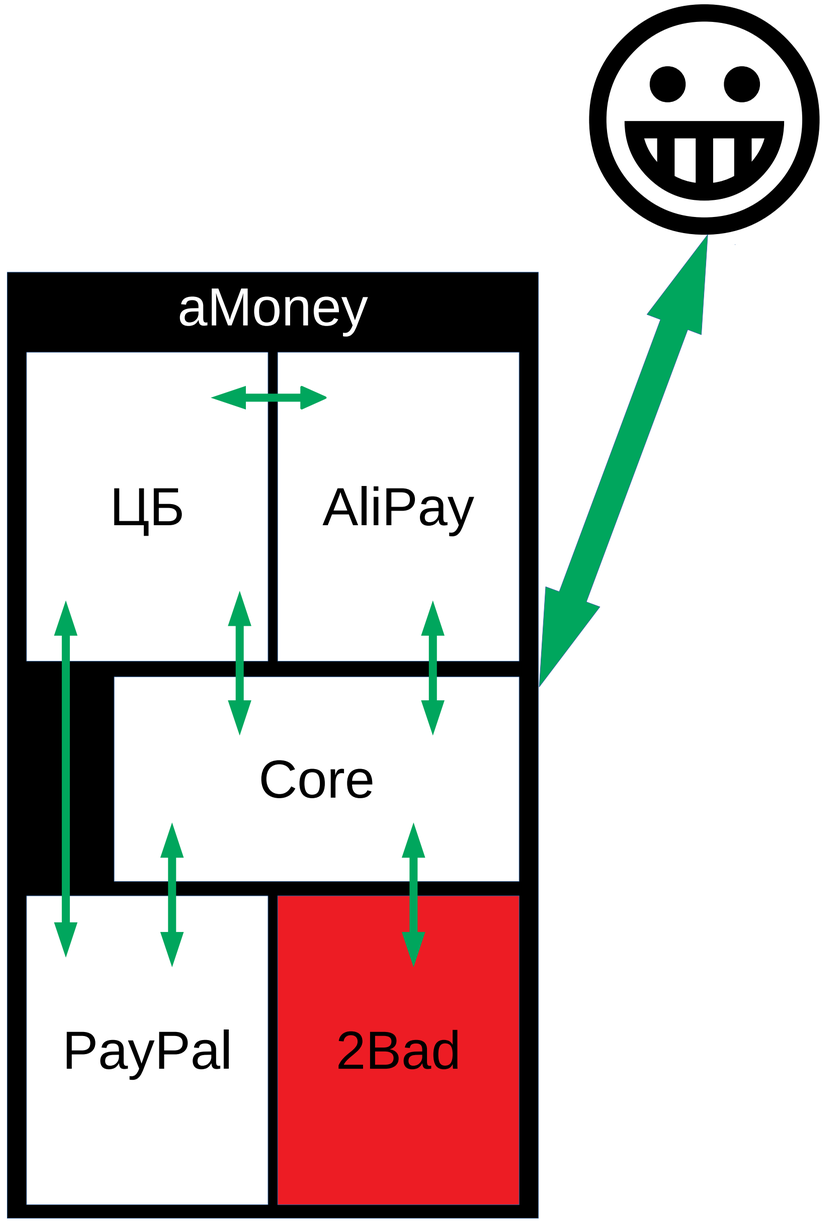

aMoney
ЦБ
AliPay
Core
PayPal
2Bad

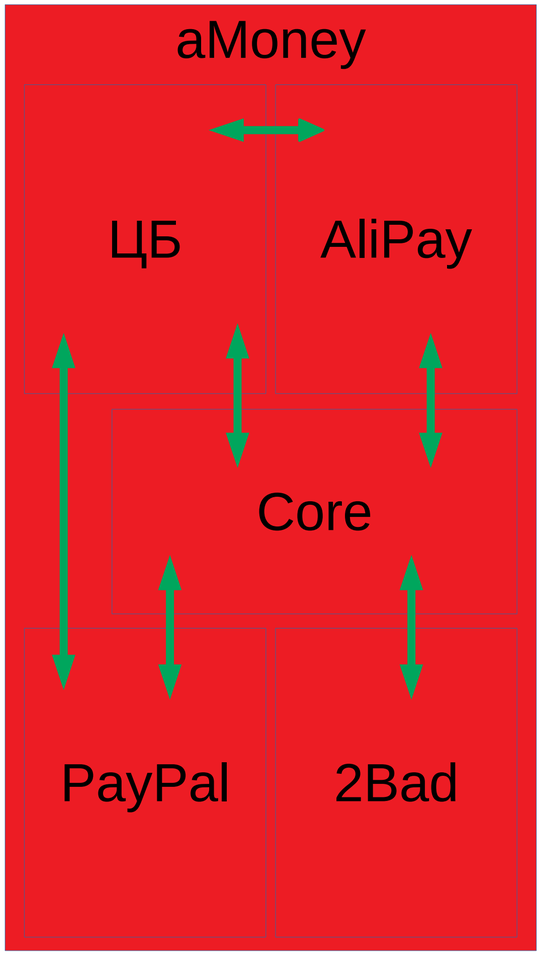

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

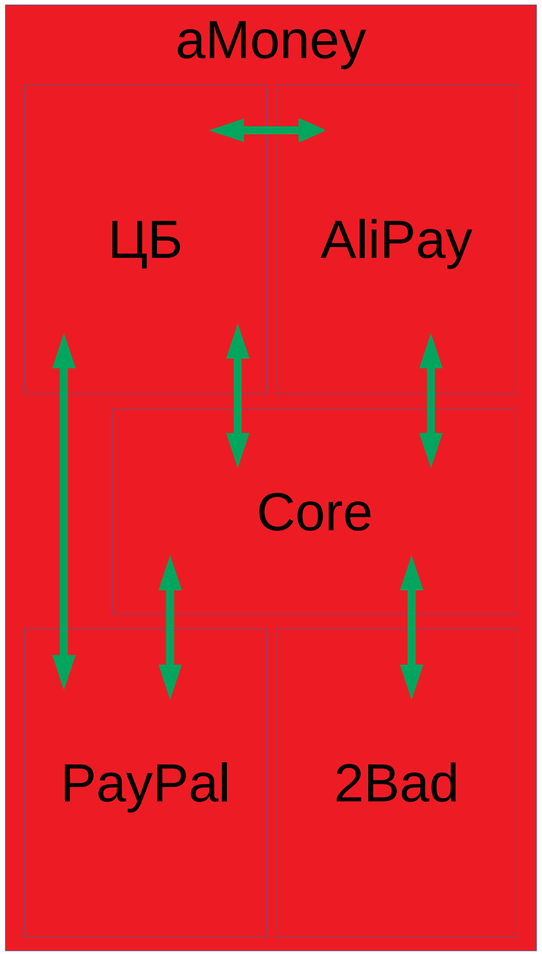
Монолит

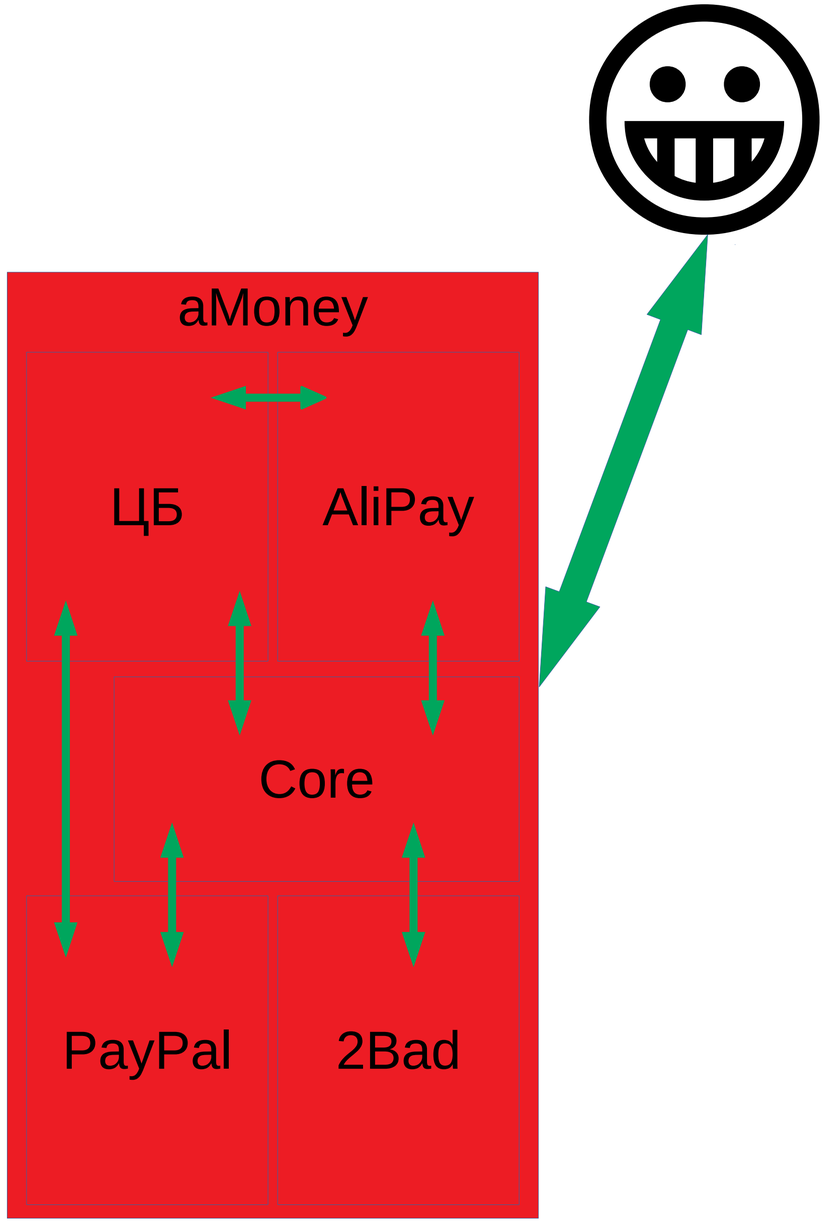

aMoney
ЦБ
AliPay
Core
PayPal
2Bad

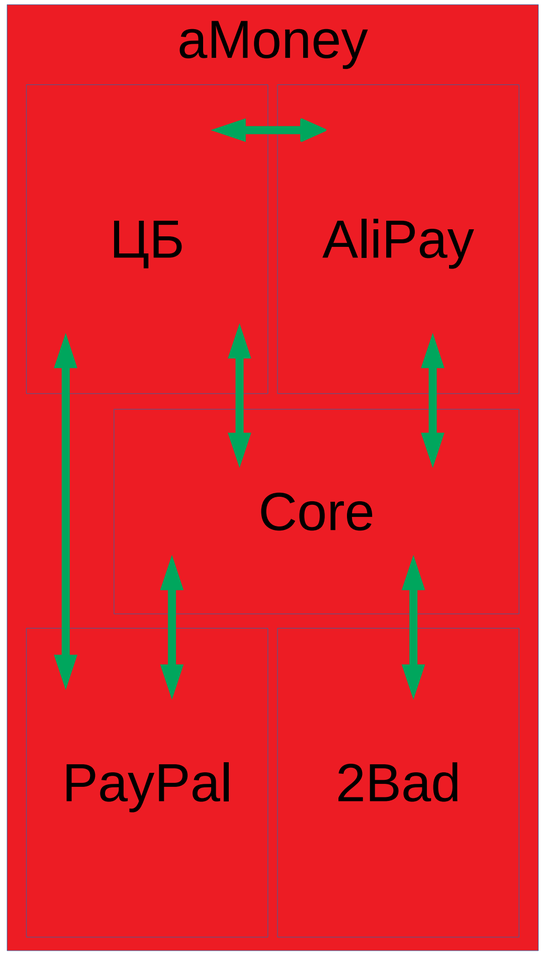

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

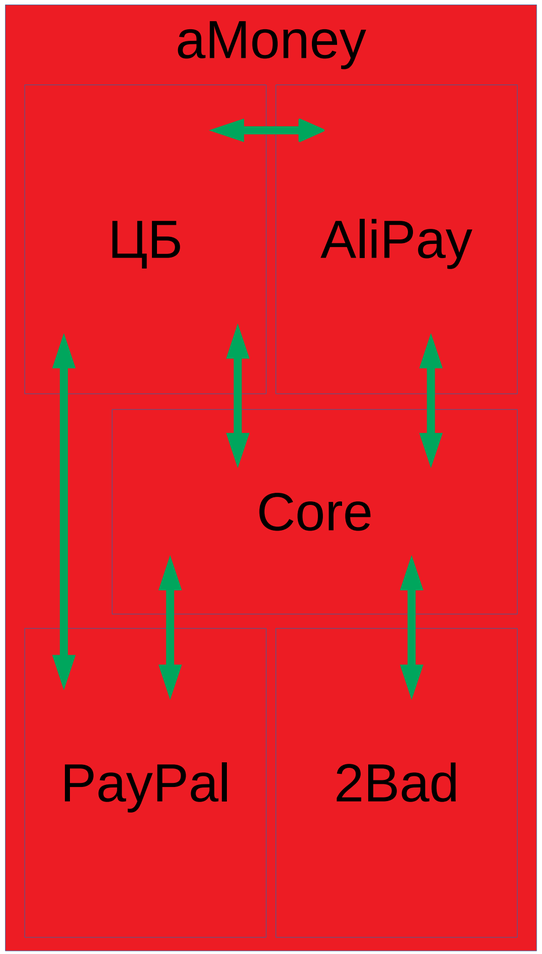
Монолит

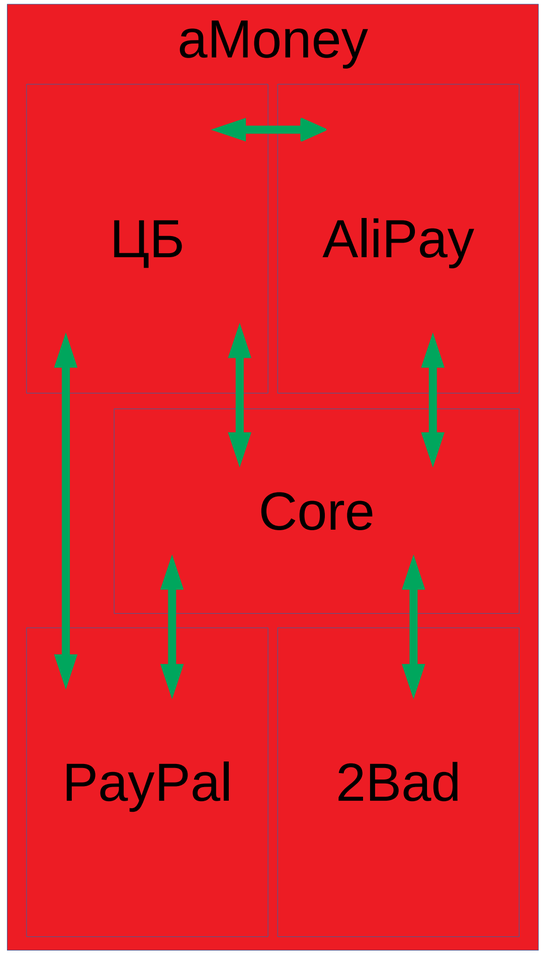

aMoney
ЦБ
AliPay
Core
PayPal
2Bad

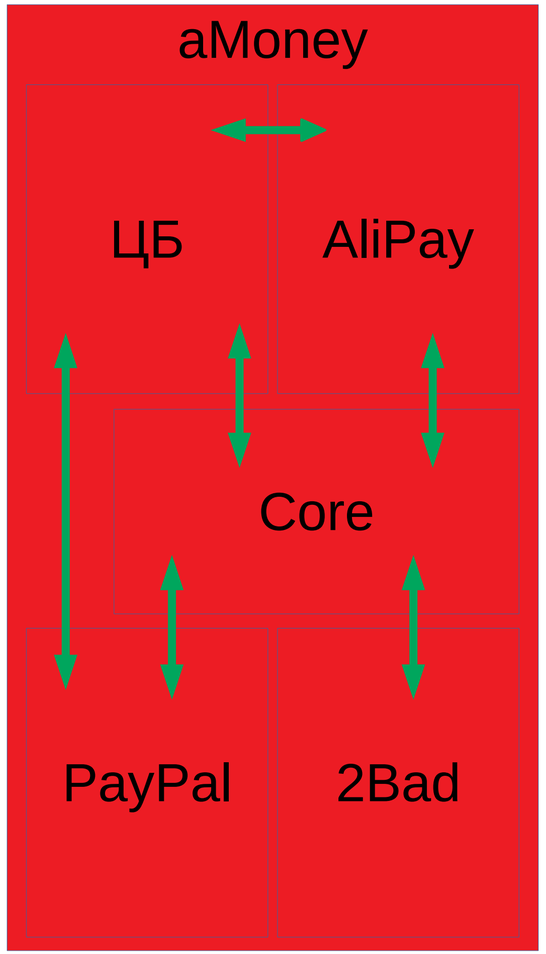

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

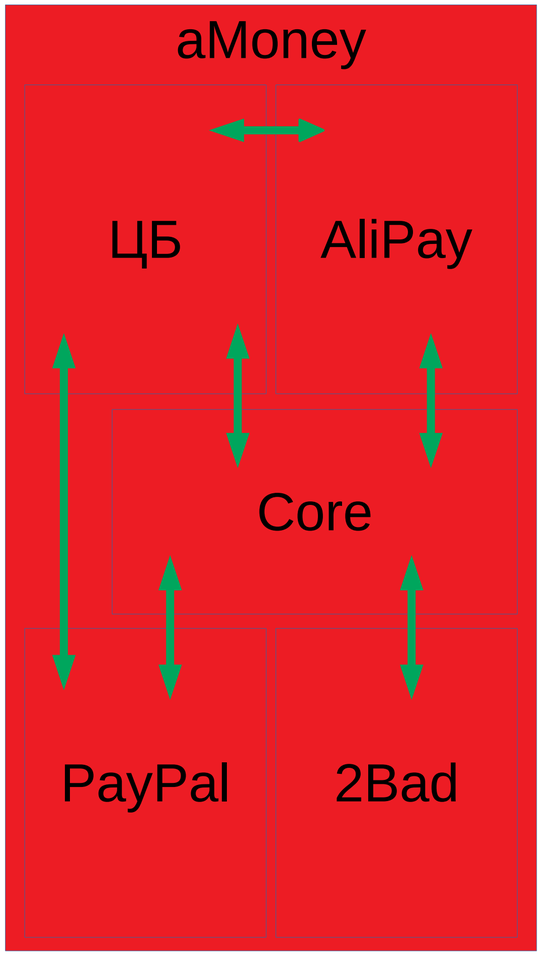
Монолит

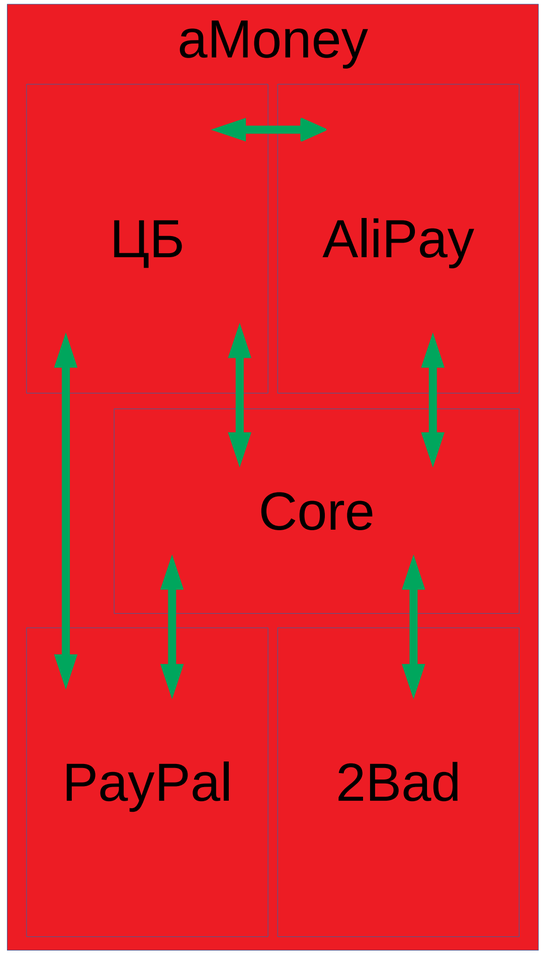

aMoney
ЦБ
AliPay
Core
PayPal
2Bad

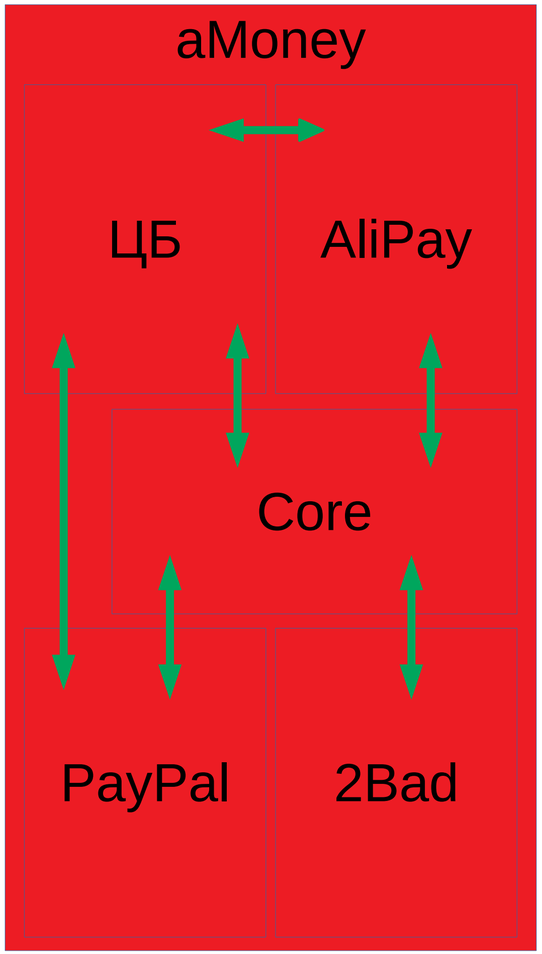

aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

Монолит


aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad


aMoney
ЦБ
AliPay
Core
PayPal
2Bad

Плюсы/минусы для небольшой команды

Плюсы:

- Простой деплой
- Дешёвая передача данных между модулями
- Тесное общение между разработчиками

Минусы:

- Не идеальная надёжность

Плюсы/минусы для небольшой команды

Плюсы:

- Простой деплой
- Дешёвая передача данных между модулями
- Тесное общение между разработчиками

Минусы:

- Не идеальная надёжность
  - и при этом множество кластеров не спасает

Плюсы/минусы для небольшой команды

Плюсы:

- Простой деплой
- Дешёвая передача данных между модулями
- Тесное общение между разработчиками

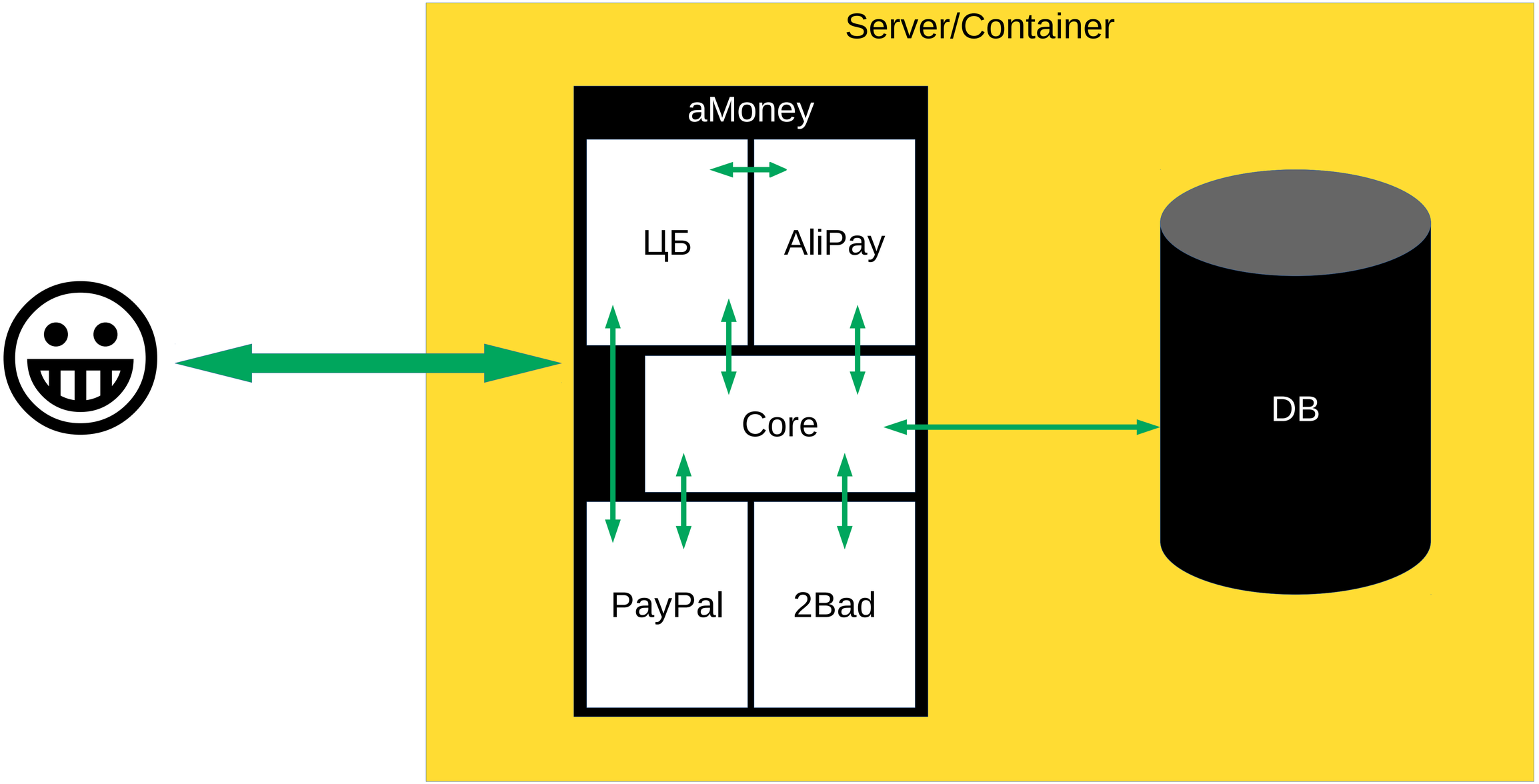
Минусы:

- Не идеальная надёжность
  - и при этом множество кластеров не спасает

Итог: неплохое решение для малых команд и малых кодовых баз

Команда стала больше, больше кода

Монолит



Плюсы/минусы для большой команды

Плюсы/минусы для большой команды

Плюсы:

Плюсы/минусы для большой команды

Плюсы:

- Дешёвая передача данных между модулями

Плюсы/минусы для большой команды

Плюсы:

- Дешёвая передача данных между модулями

Минусы:

Плюсы/минусы для большой команды

Плюсы:

- Дешёвая передача данных между модулями

Минусы:

- Плохая надёжность

Плюсы/минусы для большой команды

Плюсы:

- Дешёвая передача данных между модулями

Минусы:

- Плохая надёжность
- Тесное общение между разработчиками

Плюсы/минусы для большой команды

Плюсы:

- Дешёвая передача данных между модулями

Минусы:

- Плохая надёжность
- Тесное общение между разработчиками
- Долгая сборка

Плюсы/минусы для большой команды

Плюсы:

- Дешёвая передача данных между модулями

Минусы:

- Плохая надёжность
- Тесное общение между разработчиками
- Долгая сборка
  - Долгий деплой

Плюсы/минусы для большой команды

Плюсы:

- Дешёвая передача данных между модулями

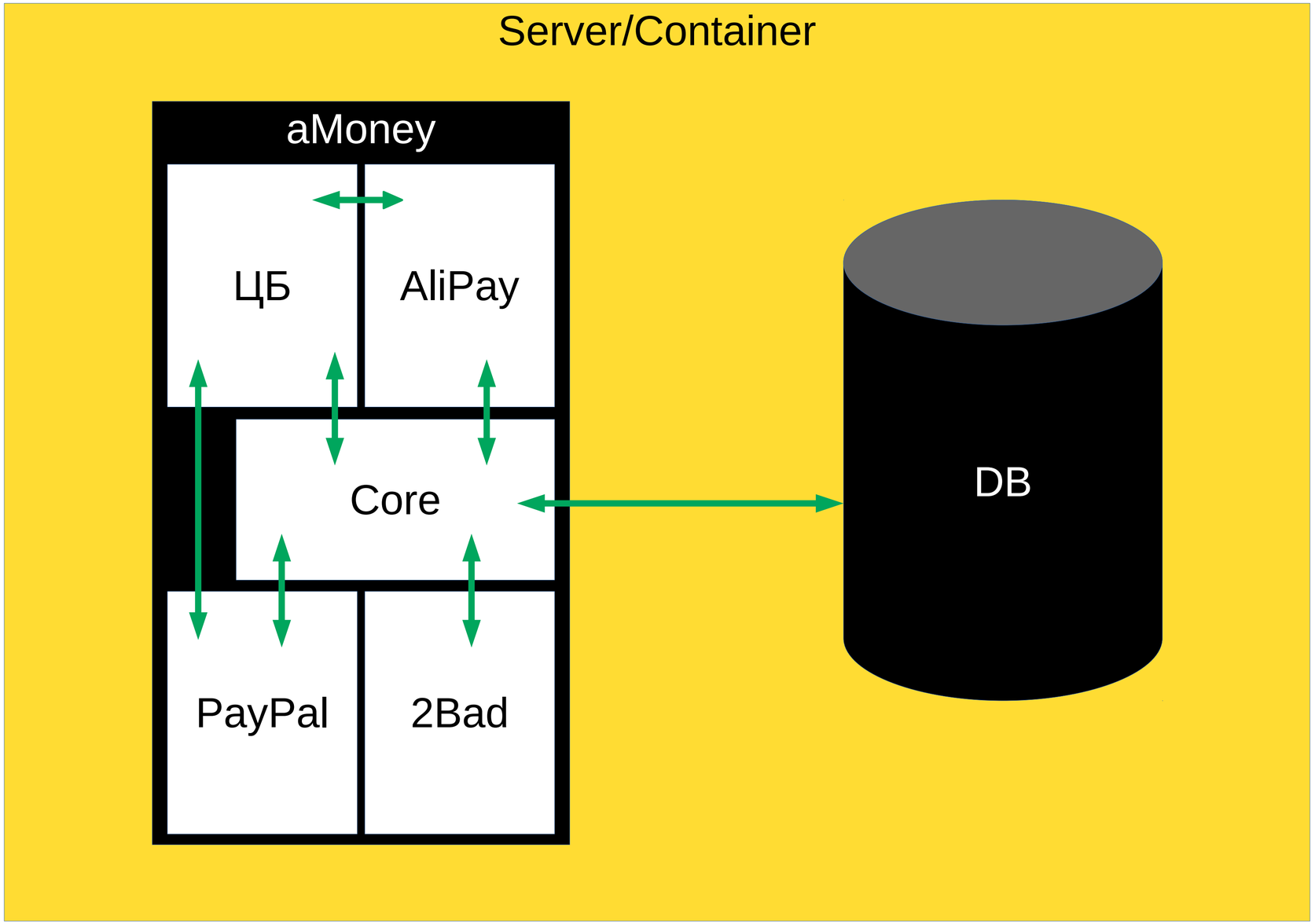
Минусы:

- Плохая надёжность
- Тесное общение между разработчиками
- Долгая сборка
  - Долгий деплой

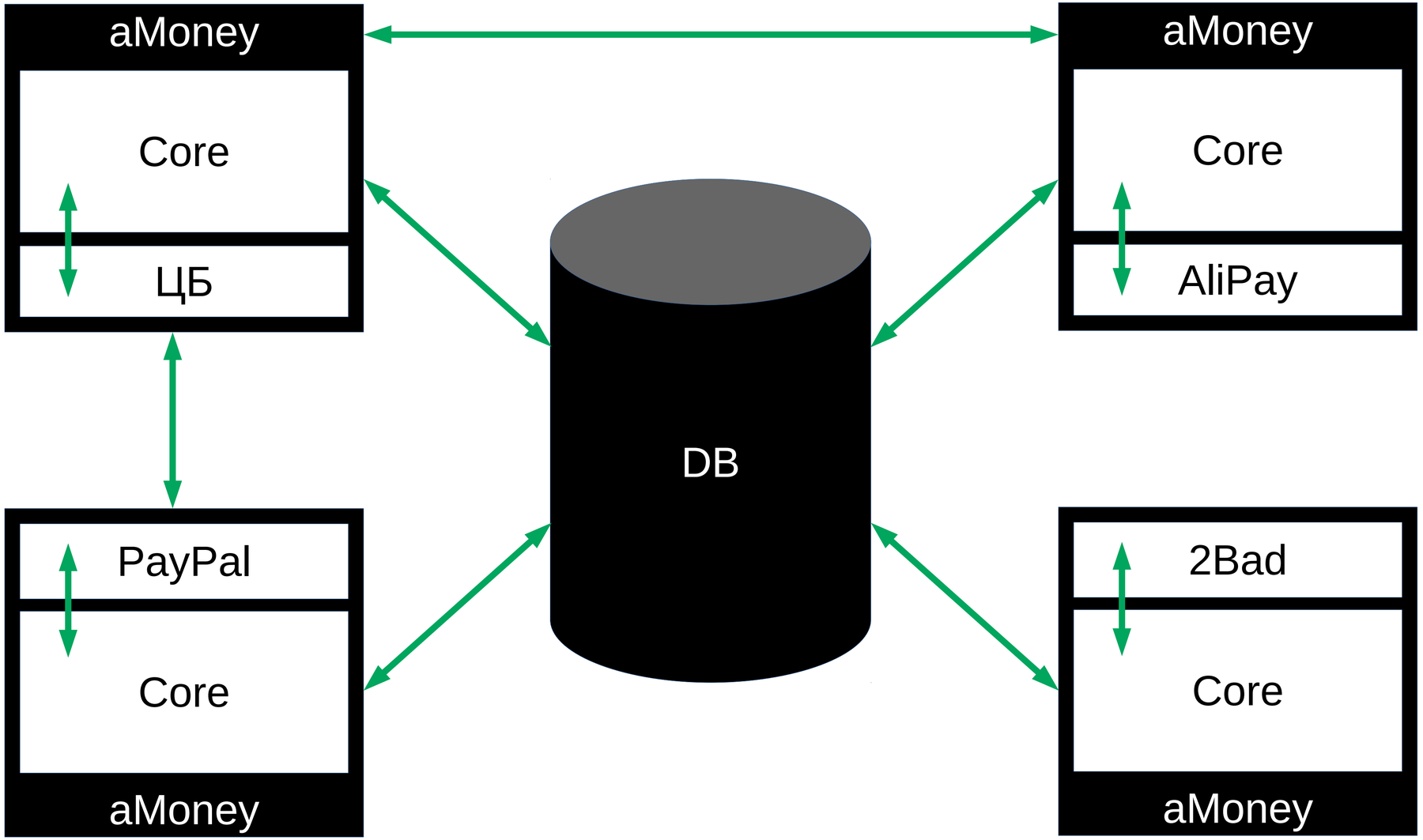
Итог: жить ещё можно

Команда стала ещё больше

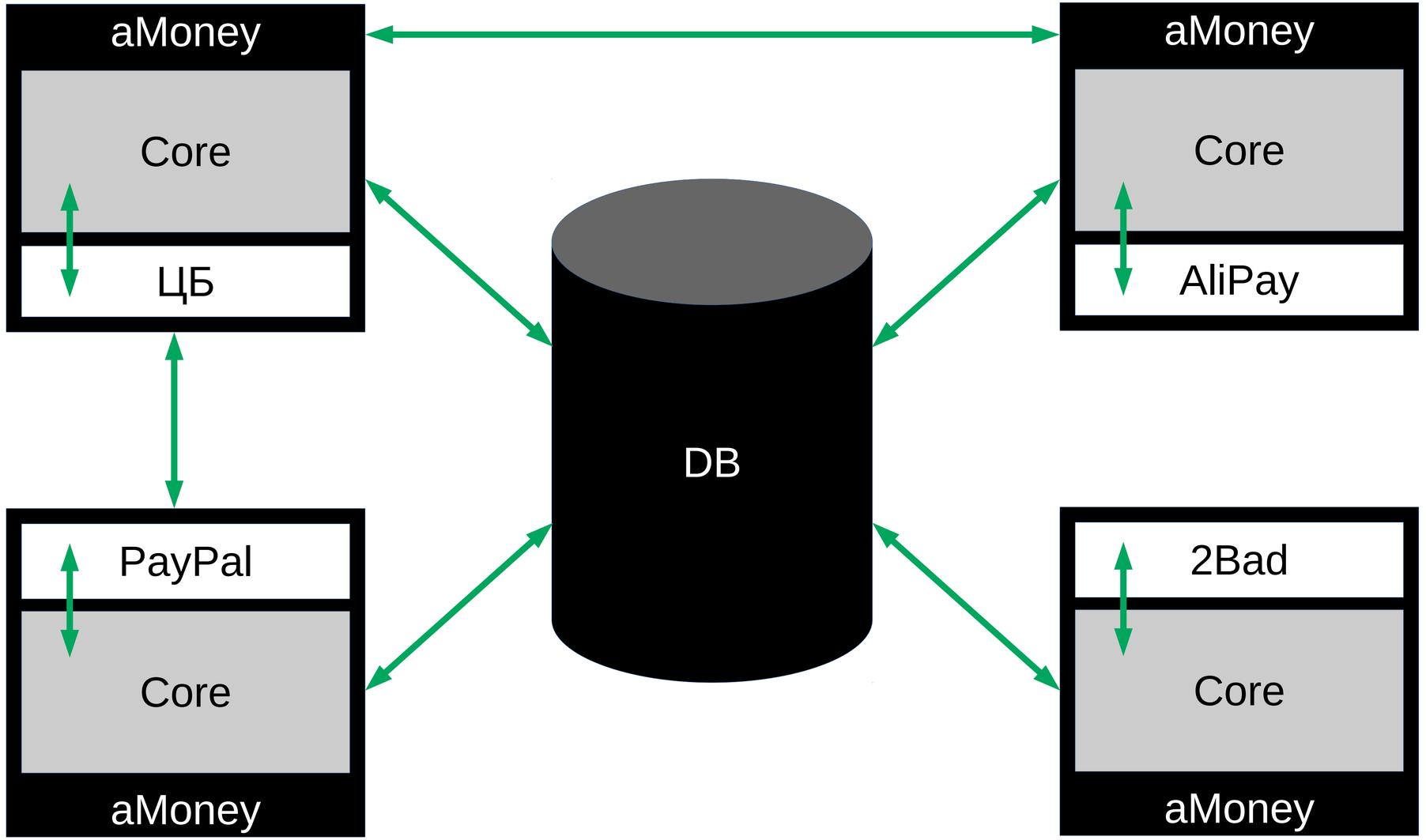
Монолит



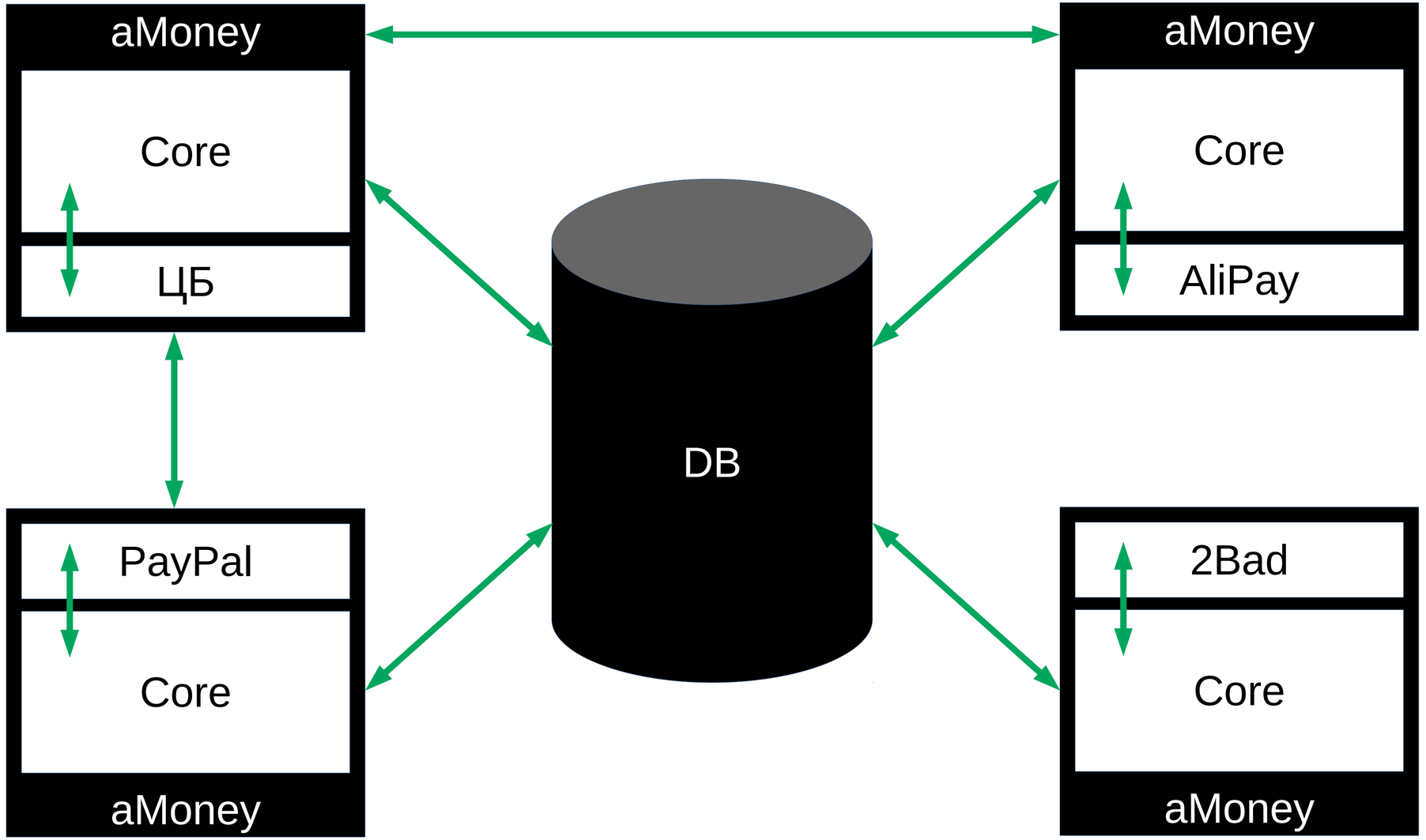
???


aMoney
Core
ЦБ
aMoney
Core
AliPay
DB
PayPal
Core
aMoney
2Bad
Core
aMoney

???


aMoney
Core
ЦБ
aMoney
Core
AliPay
DB
PayPal
Core
aMoney
2Bad
Core
aMoney

???


aMoney
Core
ЦБ
aMoney
Core
AliPay
DB
PayPal
Core
aMoney
2Bad
Core
aMoney

Плюсы/минусы ???

Плюсы/минусы ???

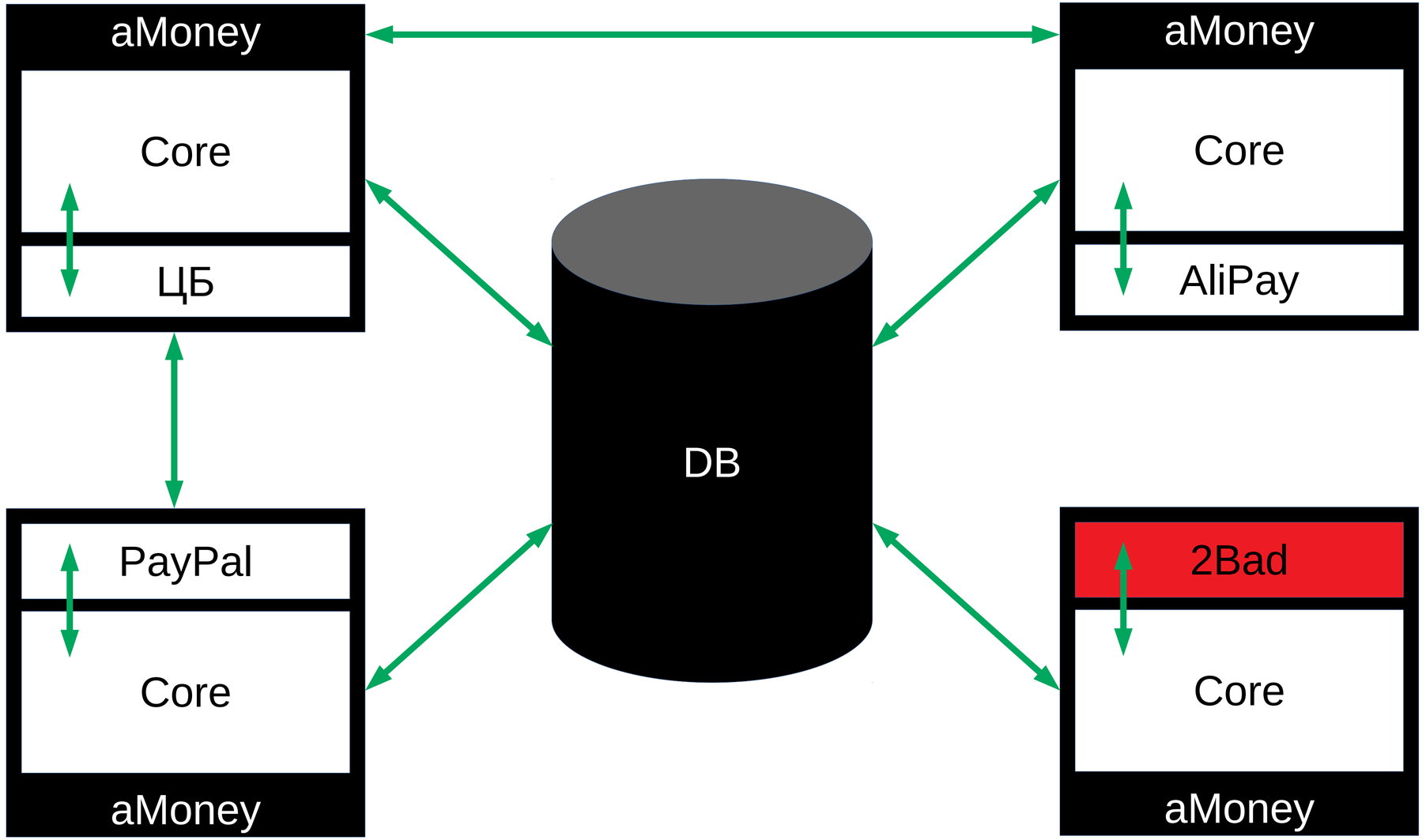
Плюсы:

Плюсы/минусы ???

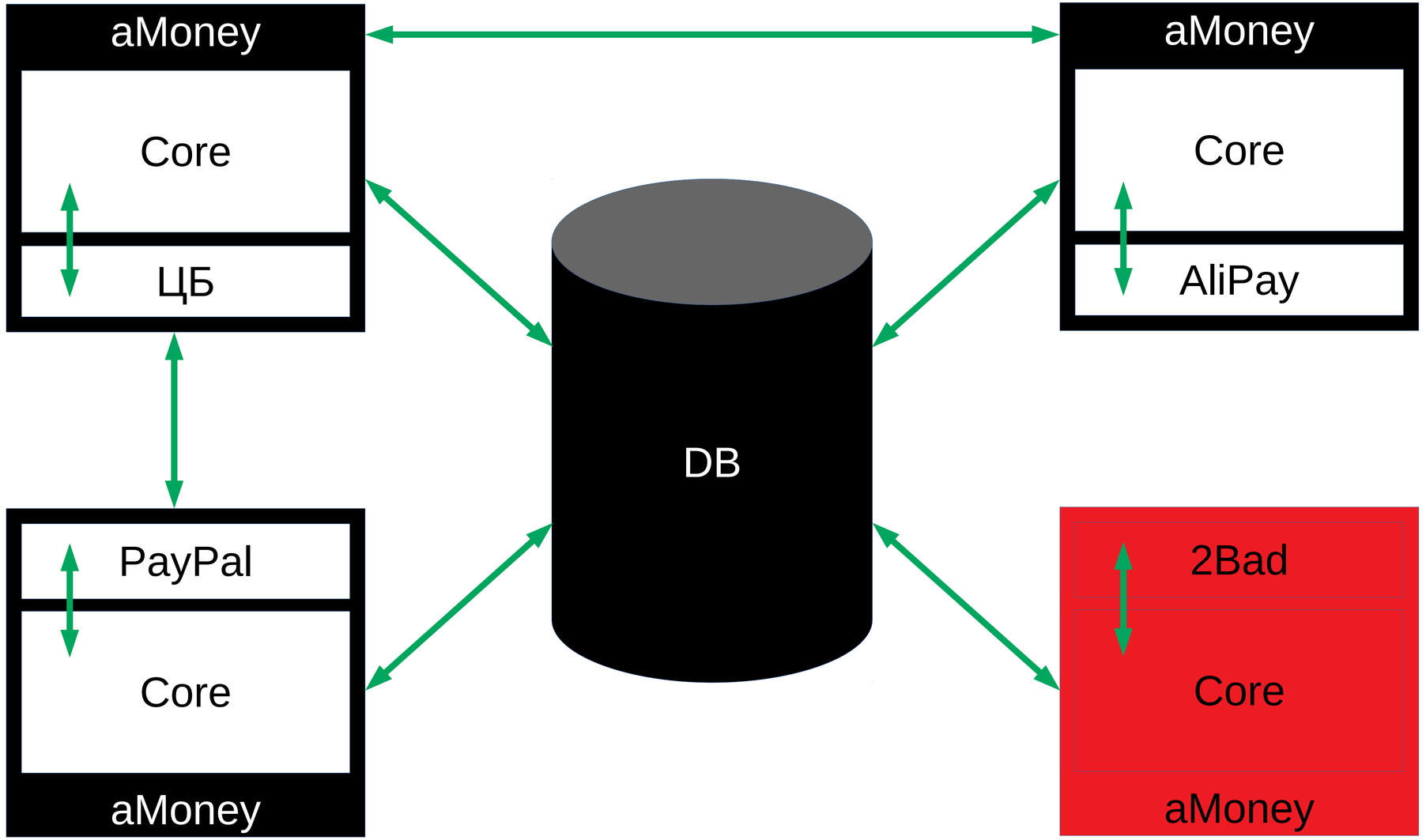
Плюсы:

- Надёжность

???


aMoney
Core
ЦБ
aMoney
Core
AliPay
DB
PayPal
Core
aMoney
2Bad
Core
aMoney

???


aMoney
Core
ЦБ
aMoney
Core
AliPay
DB
PayPal
Core
aMoney
2Bad
Core
aMoney

Плюсы/минусы ???

Плюсы:

- Надёжность

Плюсы/минусы ???

Плюсы:

- Надёжность

Минусы:

Плюсы/минусы ???

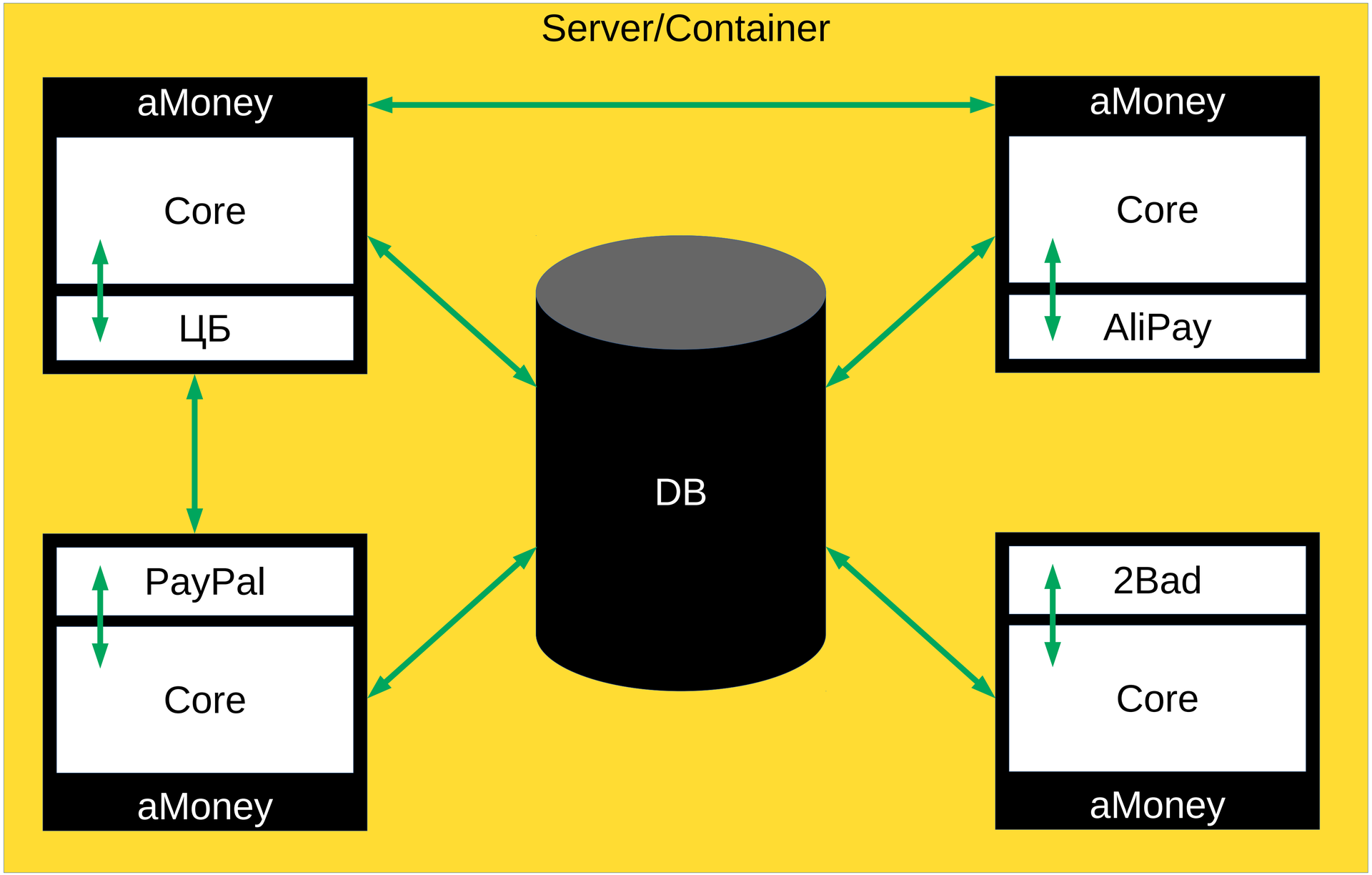
Плюсы:

- Надёжность

Минусы:

- Сложно масштабировать, страдает надёжность

???


Server/Container
aMoney
Core
ЦБ
aMoney
Core
AliPay
DB
PayPal
Core
aMoney
2Bad
Core
aMoney

Плюсы/минусы ???

Плюсы:

- Надёжность

Минусы:

- Сложно масштабировать, страдает надёжность

Плюсы/минусы ???

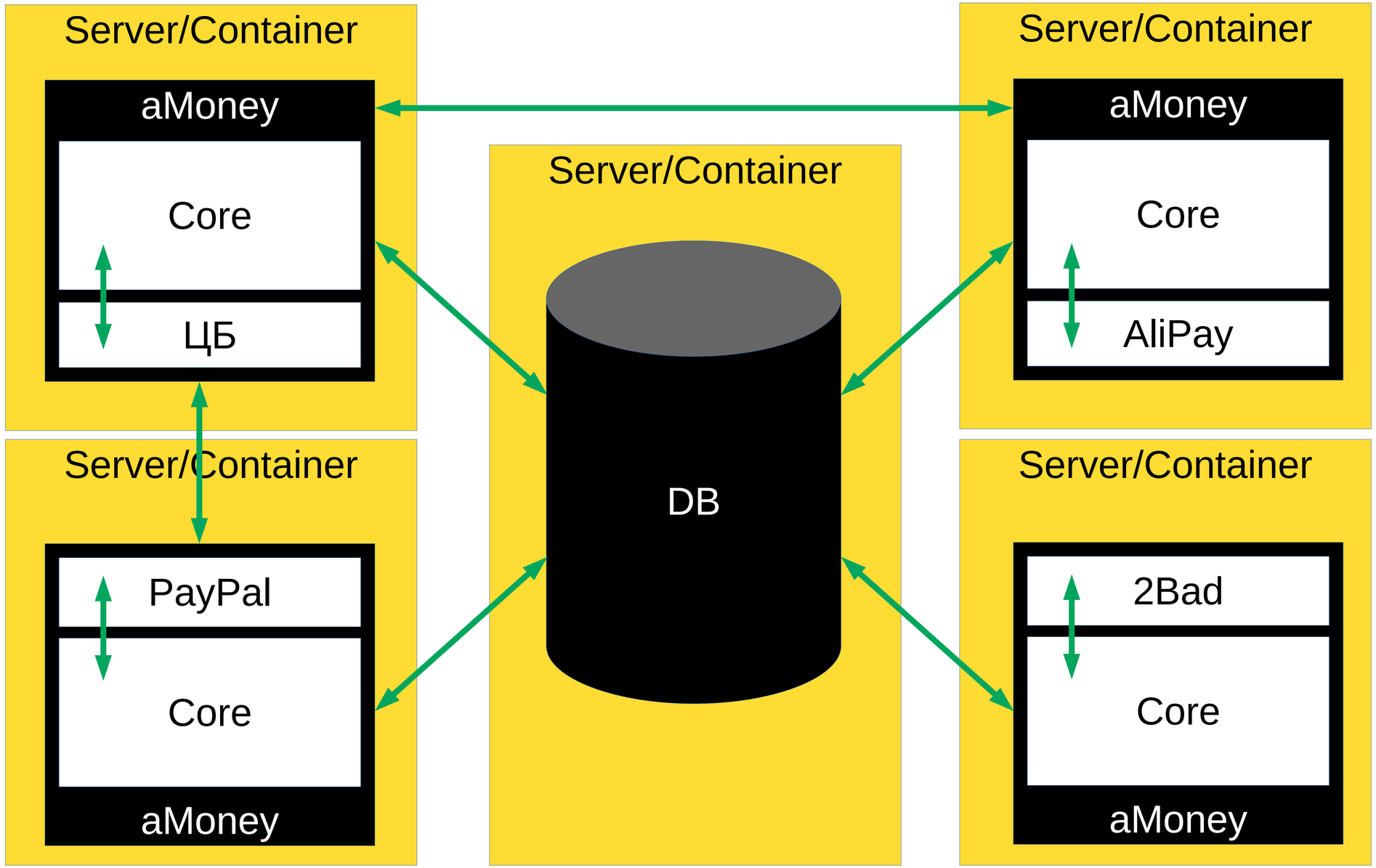
Плюсы:

- Надёжность

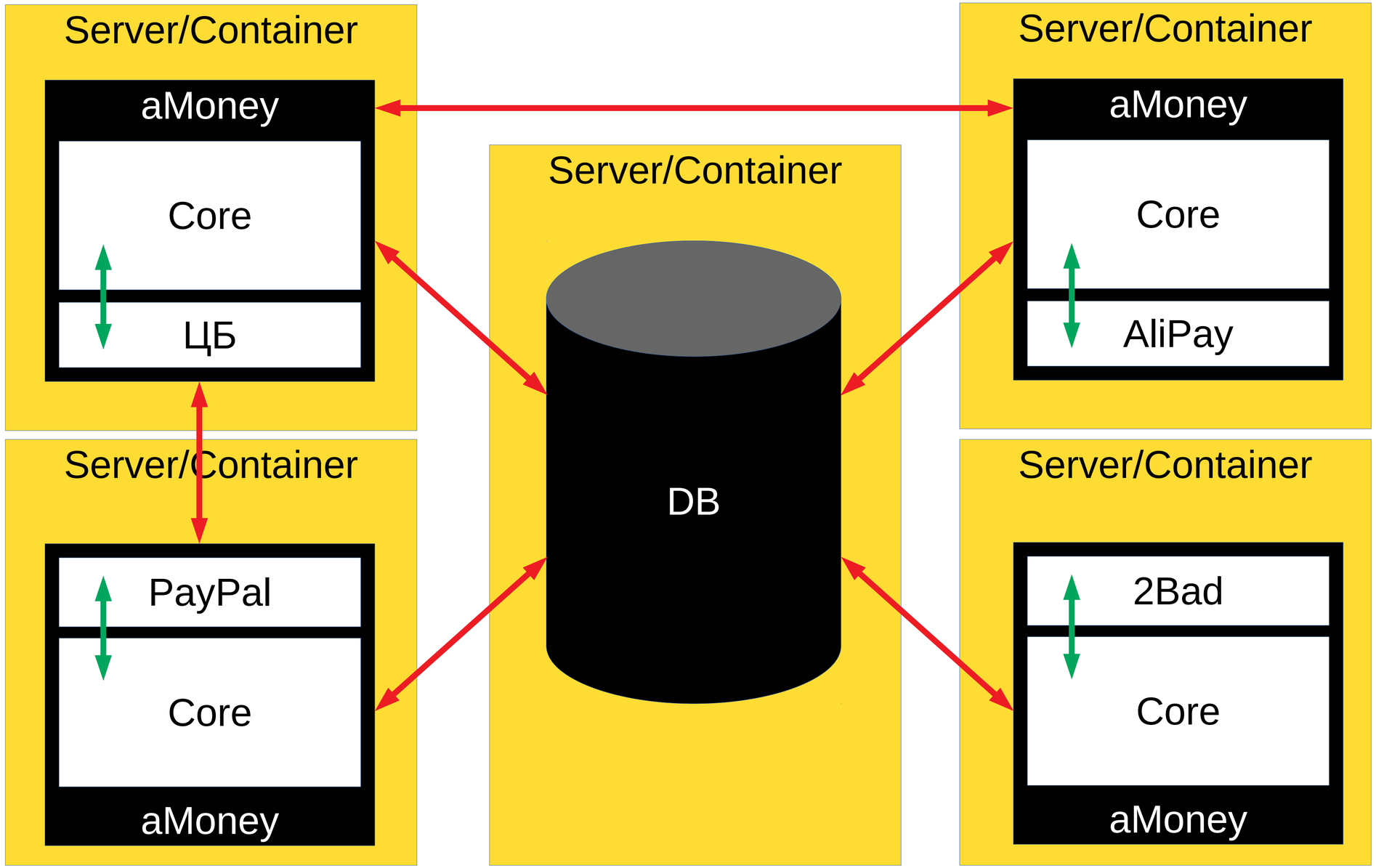
Минусы:

- Сложно масштабировать, страдает надёжность
- Дорогая передача данных между модулями

???


Server/Container
aMoney
Core
ЦБ
Server/Container
aMoney
Core
AliPay
Server/Container
DB
Server/Container
PayPal
Core
aMoney
Server/Container
2Bad
Core
aMoney

???


Server/Container
aMoney
Core
ЦБ
Server/Container
aMoney
Core
AliPay
Server/Container
DB
Server/Container
PayPal
Core
aMoney
Server/Container
2Bad
Core
aMoney

Плюсы/минусы ???

Плюсы:

- Надёжность

Минусы:

- Сложно масштабировать, страдает надёжность
- Дорогая передача данных между модулями

Плюсы/минусы ???

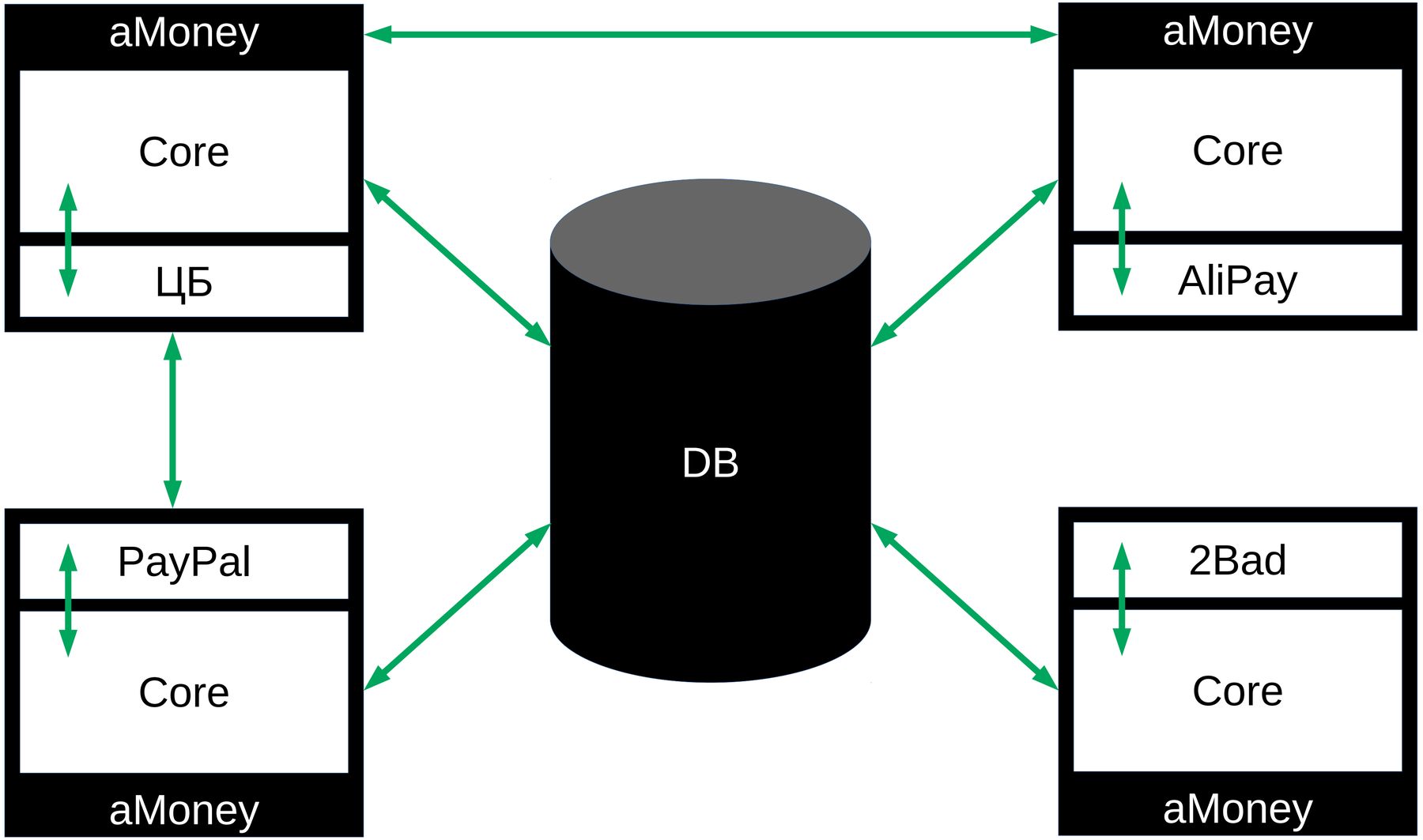
Плюсы:

- Надёжность

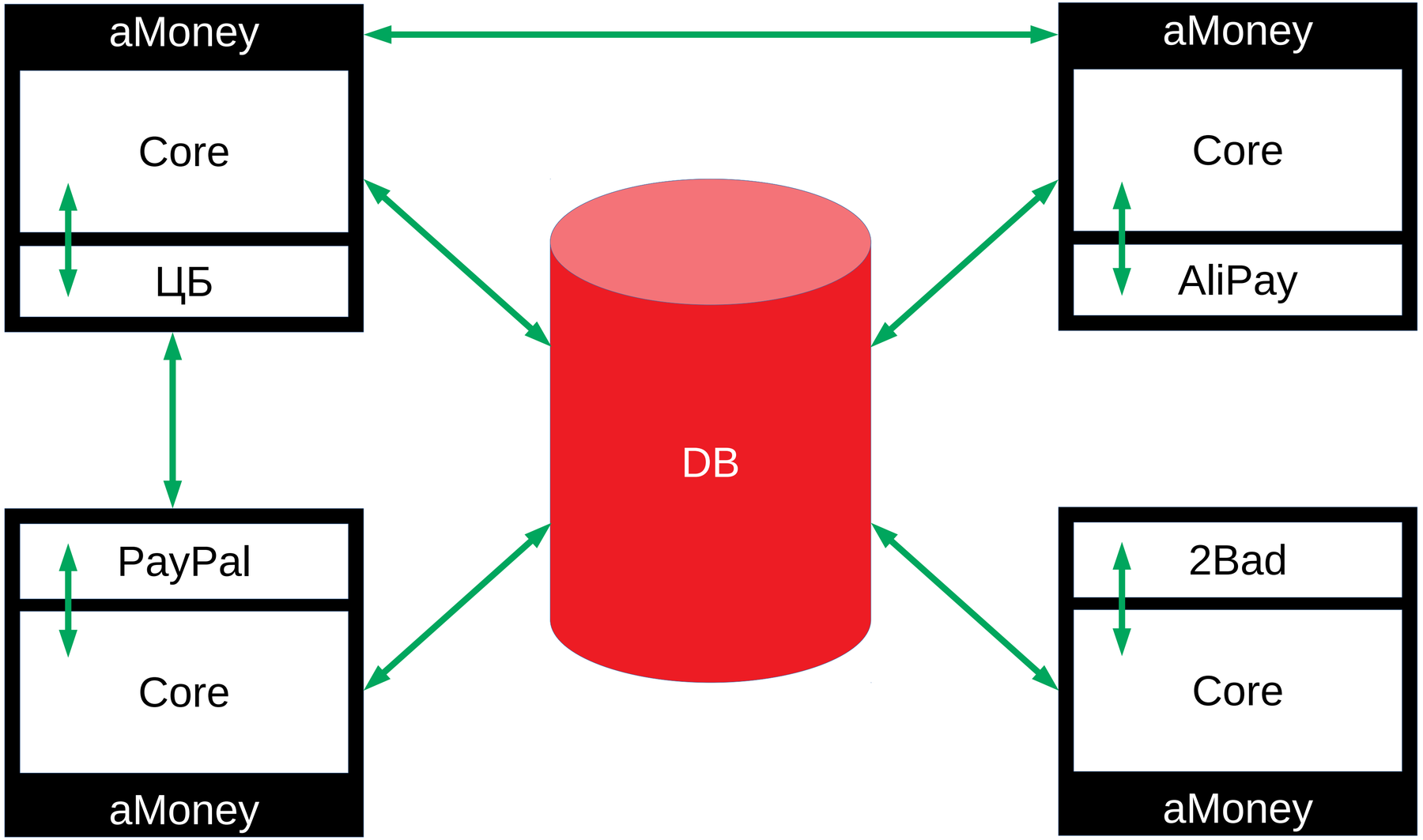
Минусы:

- Сложно масштабировать, страдает надёжность
- Дорогая передача данных между модулями
- Тесное общение между разработчиками

DB


aMoney
Core
ЦБ
aMoney
Core
AliPay
DB
PayPal
Core
aMoney
2Bad
Core
aMoney

DB — единая точка для всех команд



Плюсы/минусы ???

Плюсы:

- Надёжность

Минусы:

- Сложно масштабировать, страдает надёжность
- Дорогая передача данных между модулями
- Тесное общение между разработчиками

Плюсы/минусы ???

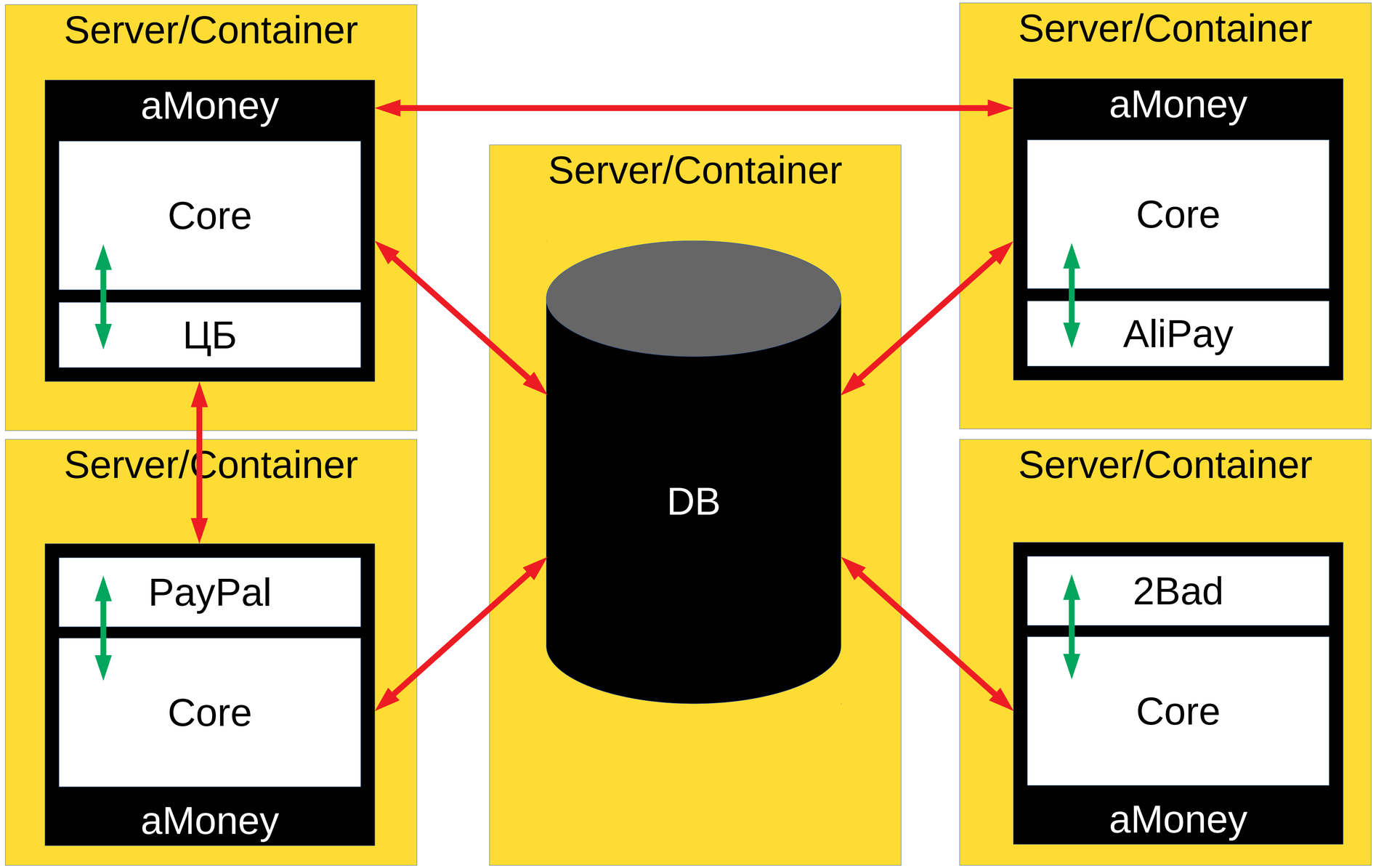
Плюсы:

- Надёжность

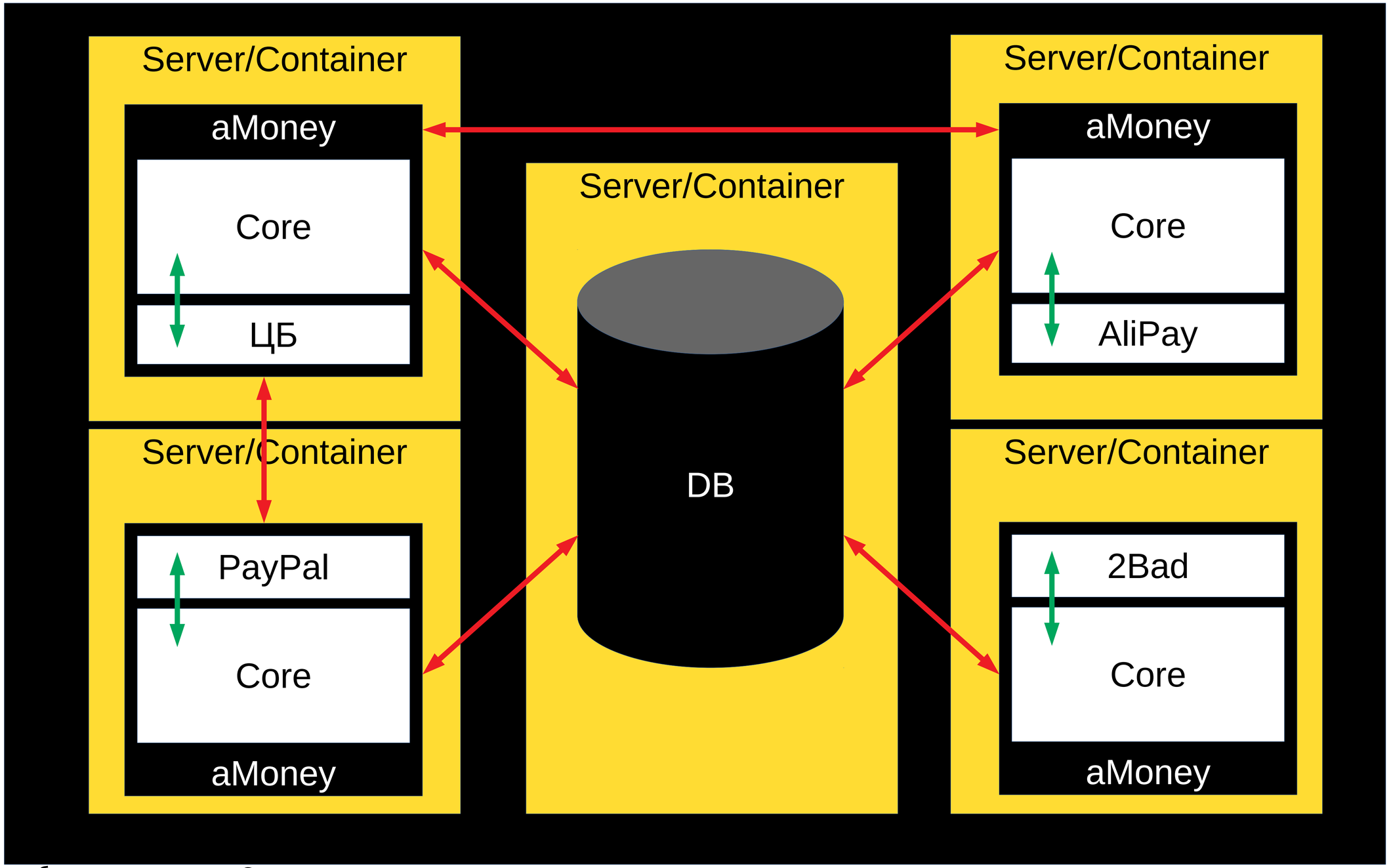
Минусы:

- Сложно масштабировать, страдает надёжность
- Дорогая передача данных между модулями
- Тесное общение между разработчиками
- Страшный деплой

???


Server/Container
aMoney
Core
ЦБ
Server/Container
aMoney
Core
AliPay
Server/Container
DB
Server/Container
PayPal
Core
aMoney
Server/Container
2Bad
Core
aMoney

???


Server/Container
aMoney
Core
ЦБ
Server/Container
aMoney
Core
AliPay
Server/Container
DB
Server/Container
PayPal
Core
aMoney
Server/Container
2Bad
Core
aMoney

Плюсы/минусы ???

Плюсы:

- Надёжность

Минусы:

- Сложно масштабировать, страдает надёжность
- Дорогая передача данных между модулями
- Тесное общение между разработчиками
- Страшный деплой

Плюсы/минусы ???

Плюсы:

- Надёжность

Минусы:

- Сложно масштабировать, страдает надёжность
- Дорогая передача данных между модулями
- Тесное общение между разработчиками
- Страшный деплой
- Затраты на железо

Плюсы/минусы ???

Плюсы:

- Надёжность

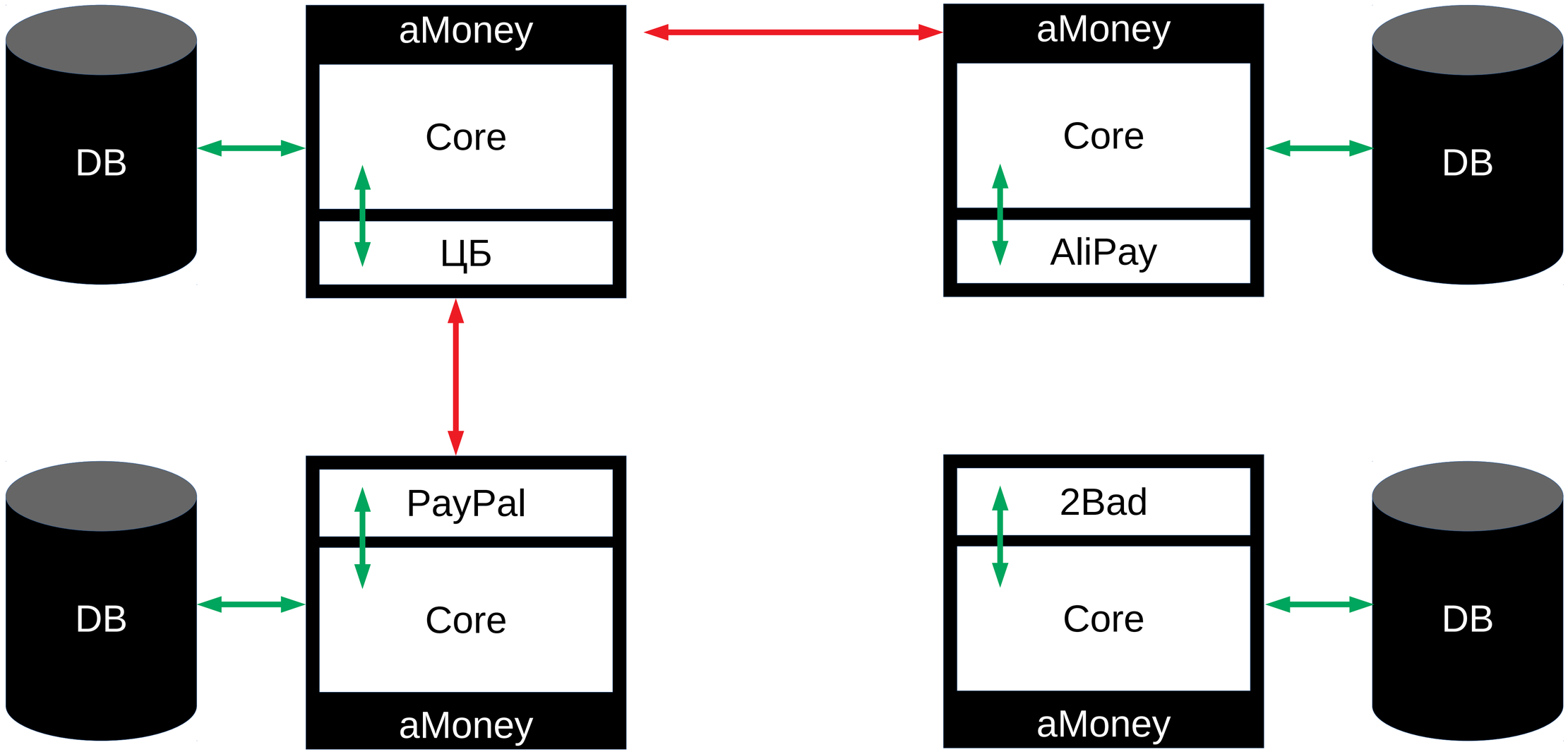
Минусы:

- Сложно масштабировать, страдает надёжность
- Дорогая передача данных между модулями
- Тесное общение между разработчиками
- Страшный деплой
- Затраты на железо

Итог: быстрое решение для увеличения надежности

Правильные микросервисы

Микросервисы



Плюсы/минусы микросервисов для большой команды

Плюсы/минусы микросервисов для большой команды

Плюсы:

Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность

Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность
- Быстрый деплой

Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность
- Быстрый деплой
- Быстрая сборка

Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность
- Быстрый деплой
- Быстрая сборка
- Тесное общение разработчиков модуля

Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность
- Быстрый деплой
- Быстрая сборка
- Тесное общение разработчиков модуля

Минусы:

Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность
- Быстрый деплой
- Быстрая сборка
- Тесное общение разработчиков модуля

Минусы:

- Дорогая передача данных между модулями

Плюсы/минусы микросервисов для большой команды

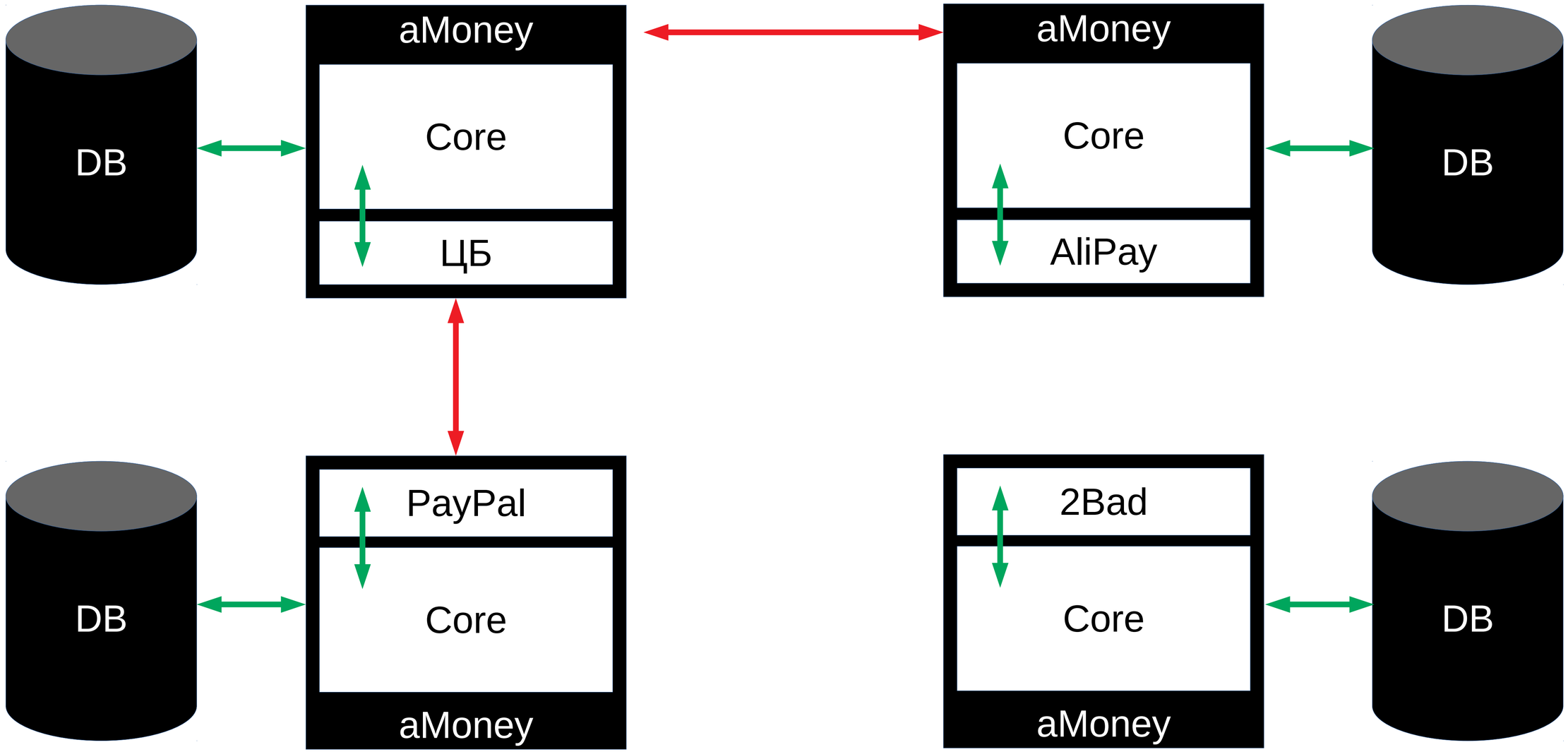
Плюсы:

- Надёжность
- Быстрый деплой
- Быстрая сборка
- Тесное общение разработчиков модуля

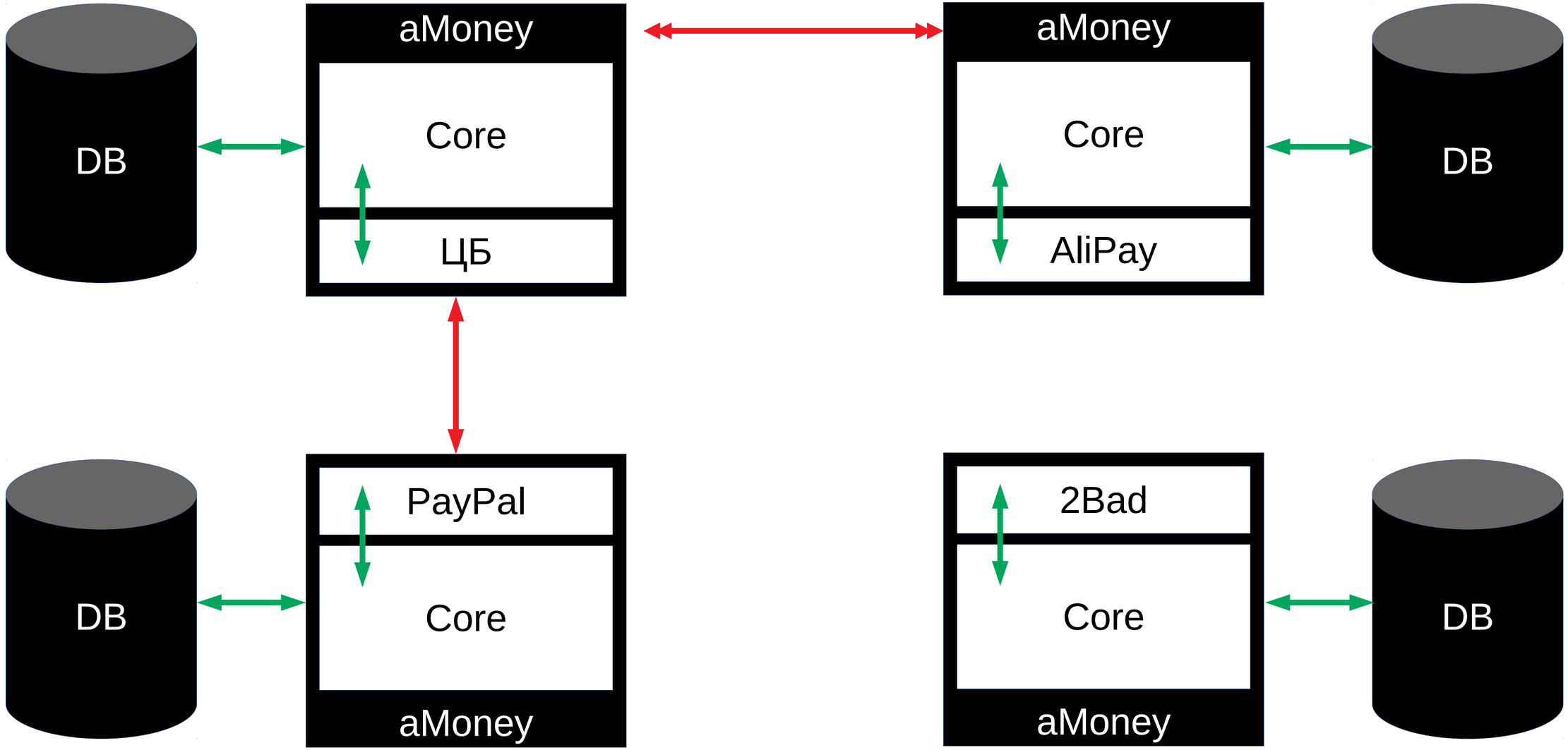
Минусы:

- Дорогая передача данных между модулями
- Обязательное версионирование и поддержка старых версий

Микросервисы



Микросервисы



Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность
- Быстрый деплой
- Быстрая сборка
- Тесное общение разработчиков модуля

Минусы:

- Дорогая передача данных между модулями
- Обязательное версионирование и поддержка старых версий

Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность
- Быстрый деплой
- Быстрая сборка
- Тесное общение разработчиков модуля

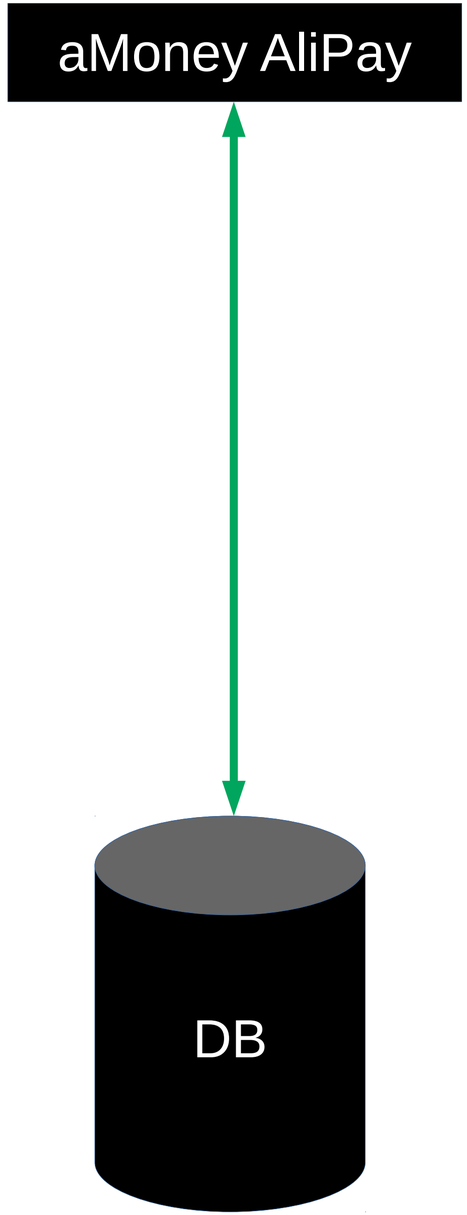
Минусы:

- Дорогая передача данных между модулями
- Обязательное версионирование и поддержка старых версий
- Большие траты на железо

А при чём тут балансеры?

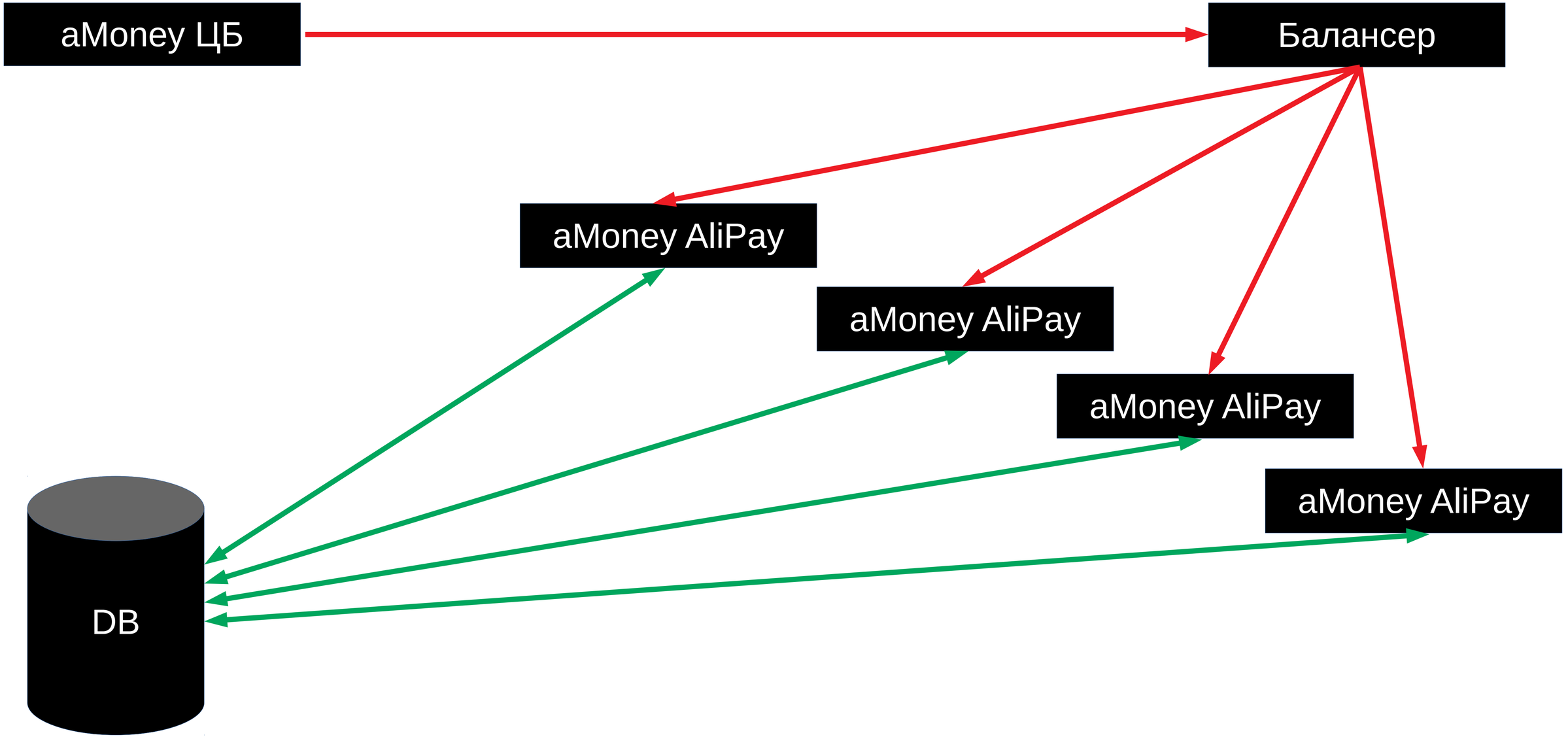
Микросервисы

aMoney ЦБ

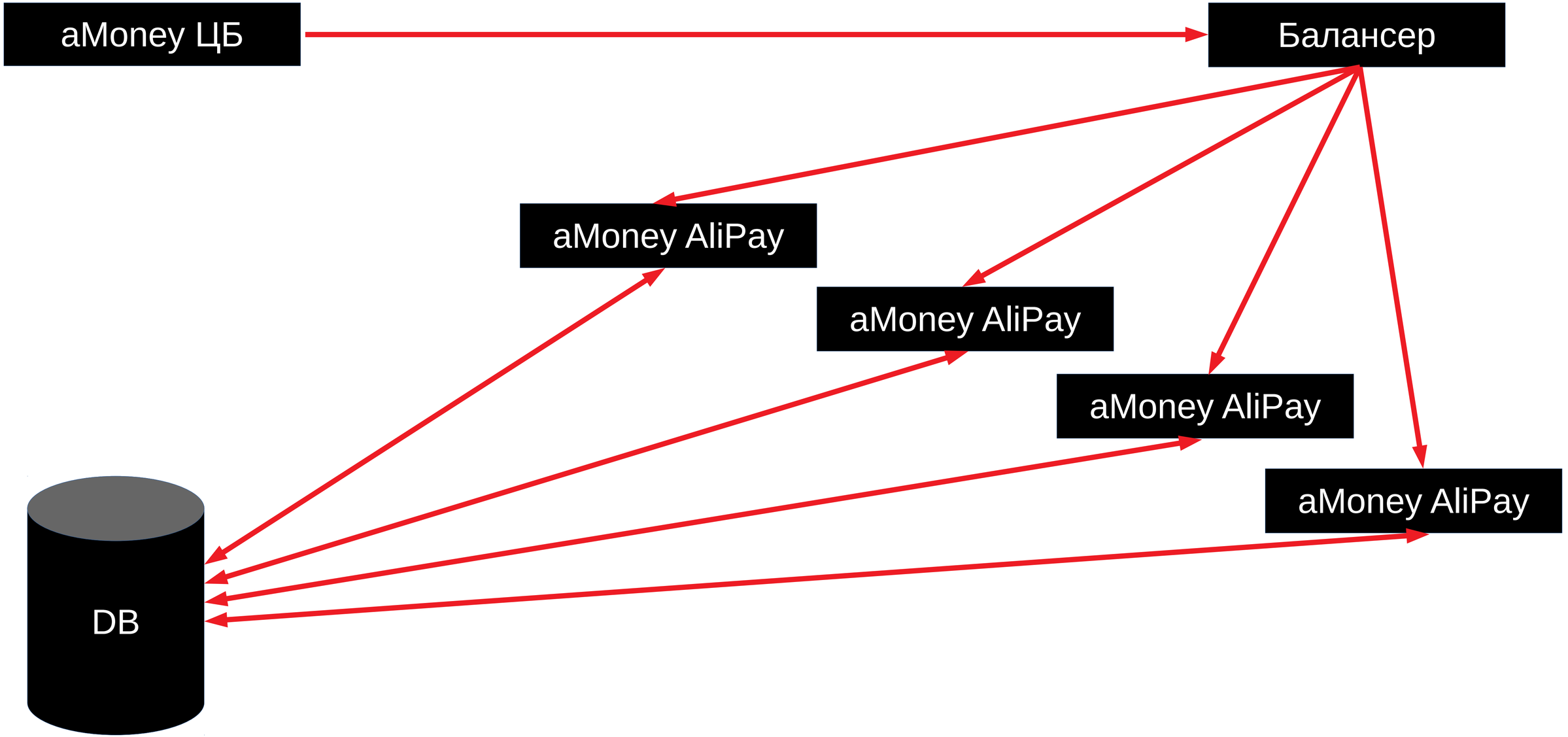


Микросервисы

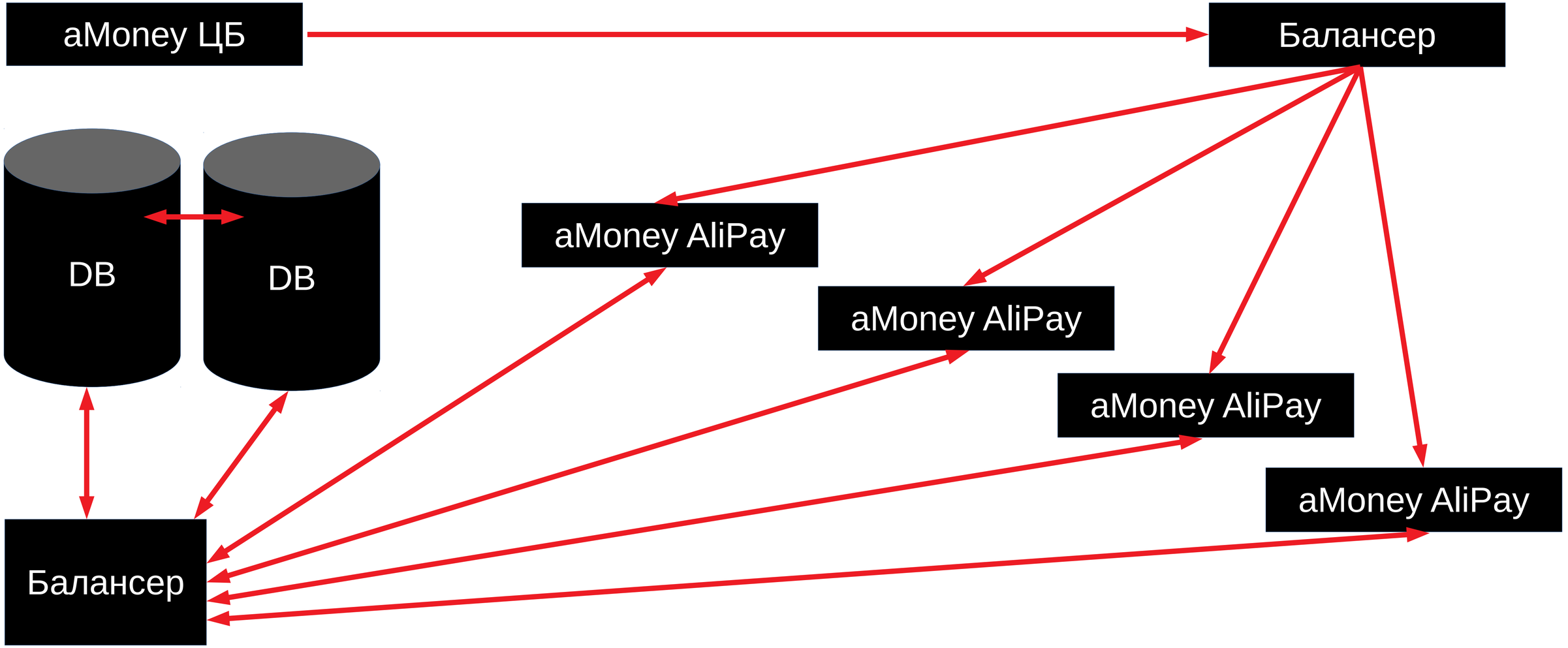
Микросервисы



Микросервисы

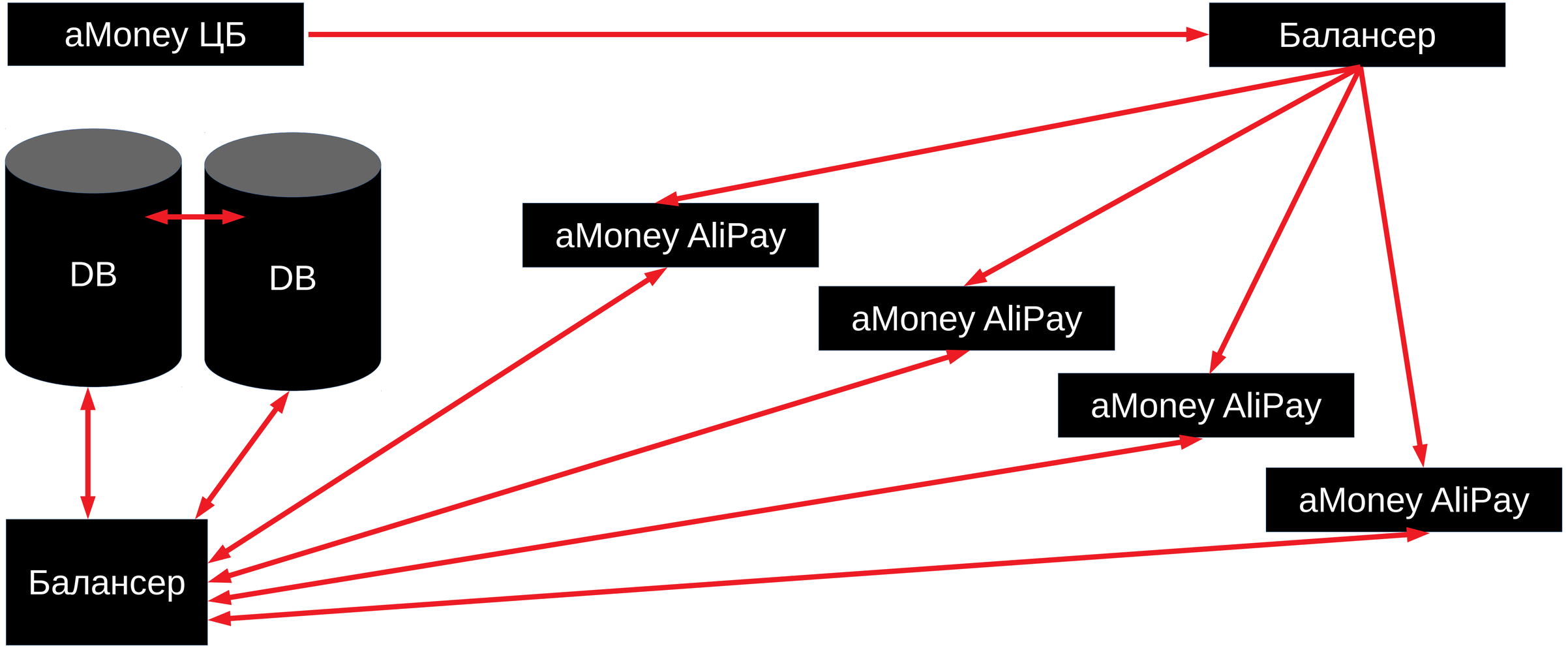


Микросервисы

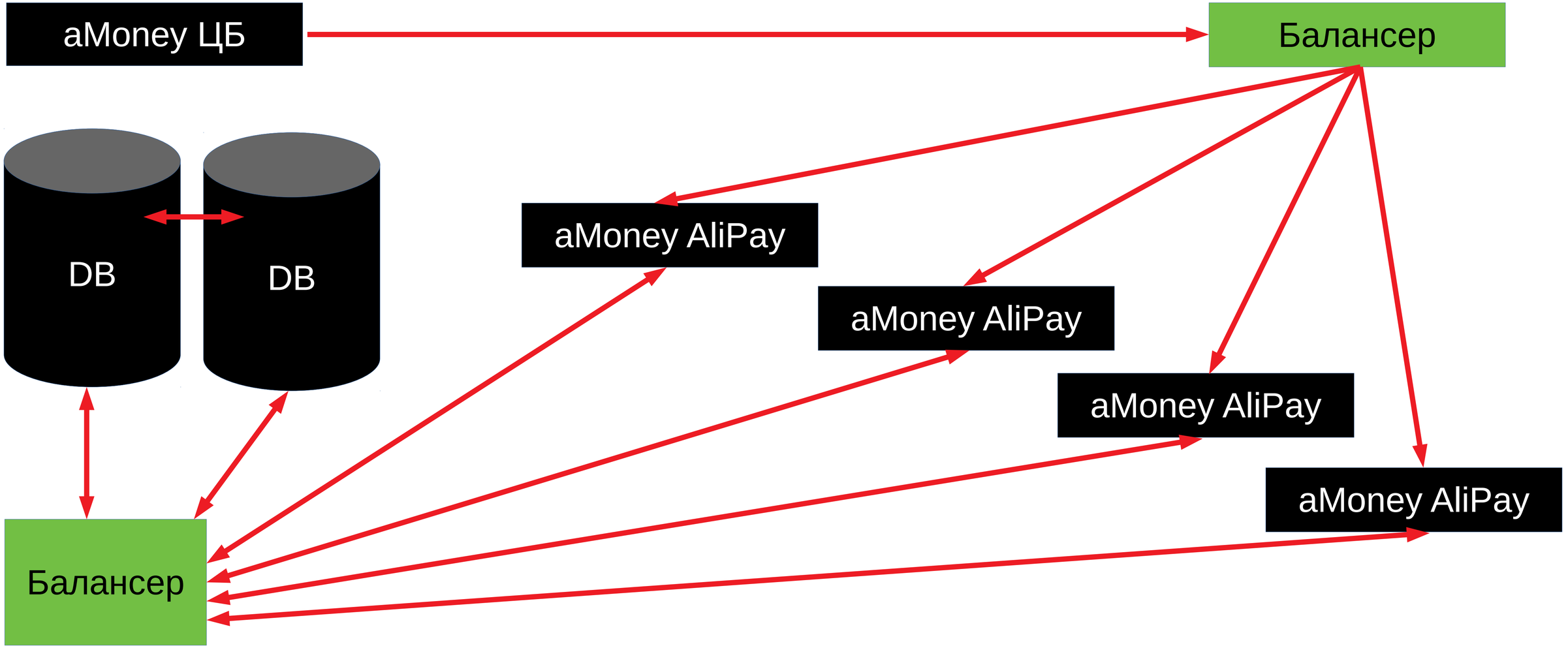


Как ускорить?

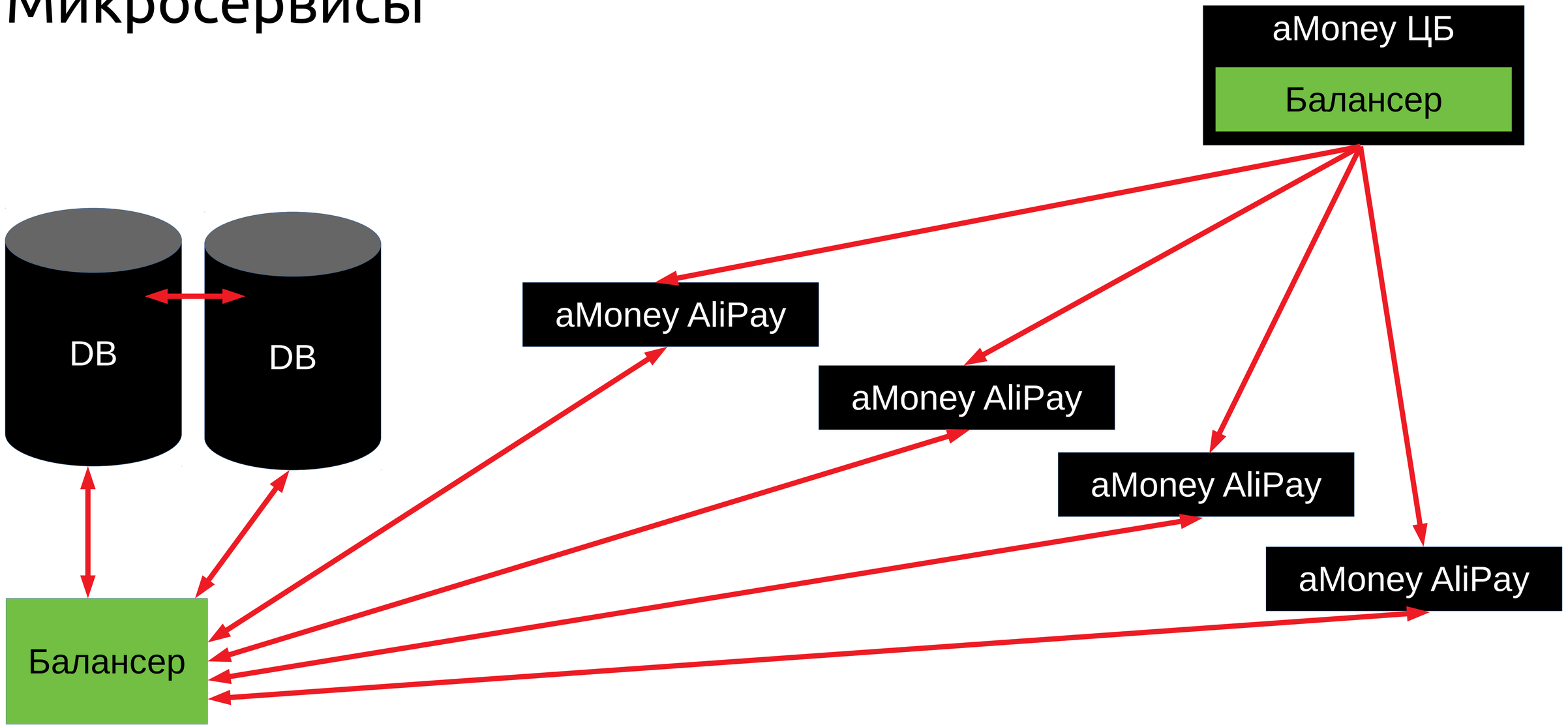
Микросервисы



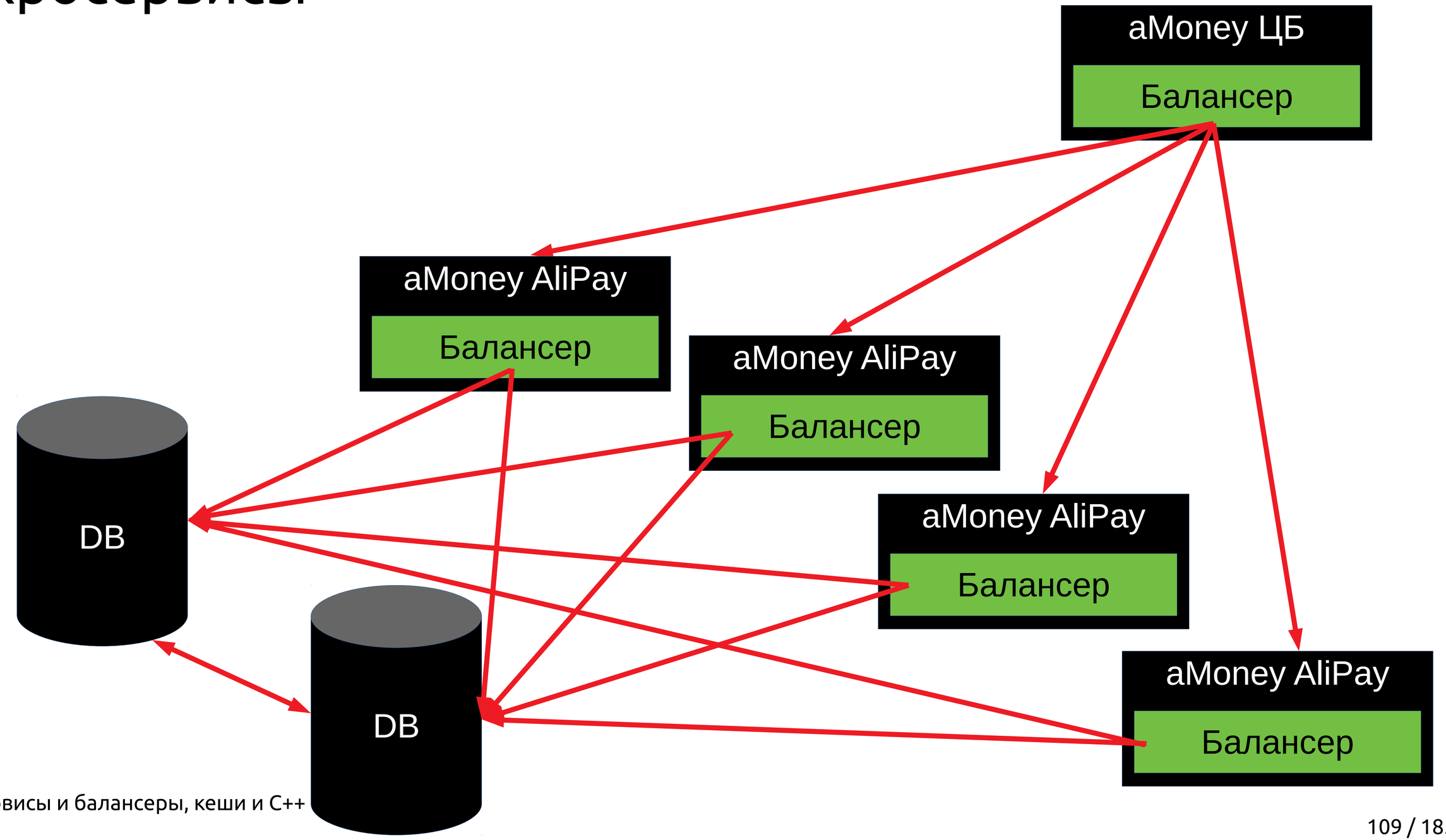
Микросервисы



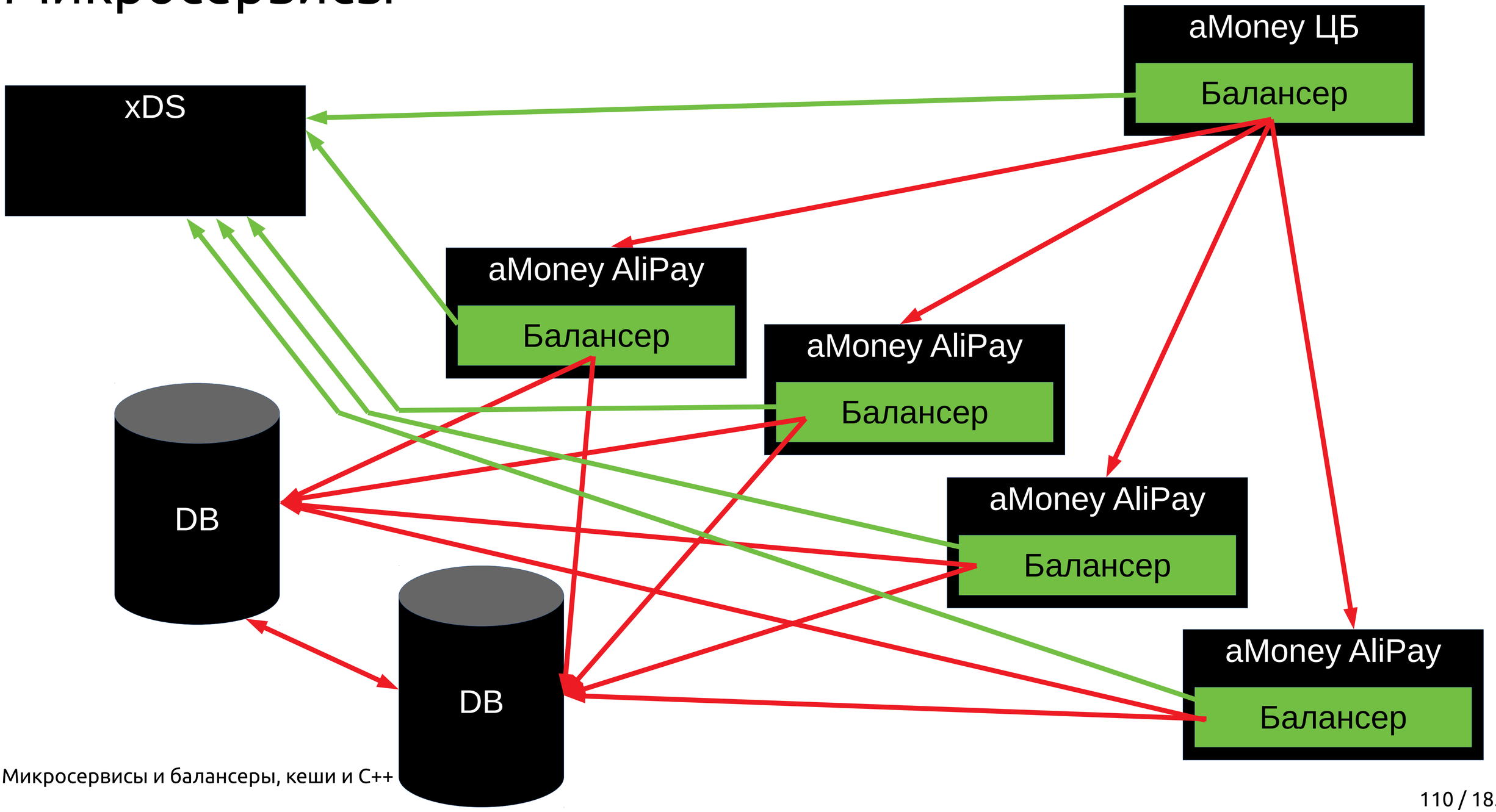
Микросервисы



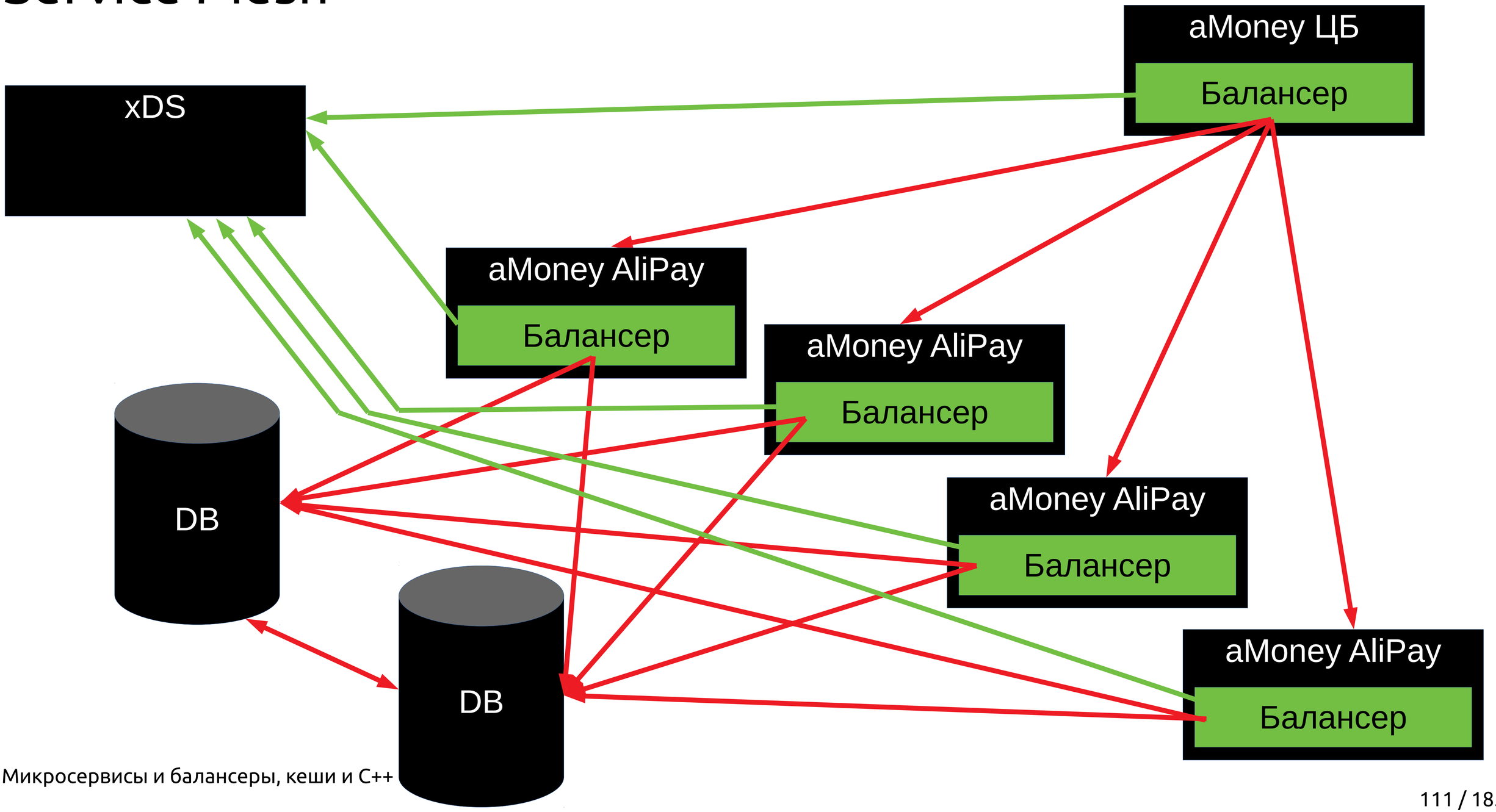
Микросервисы



Микросервисы



Service Mesh

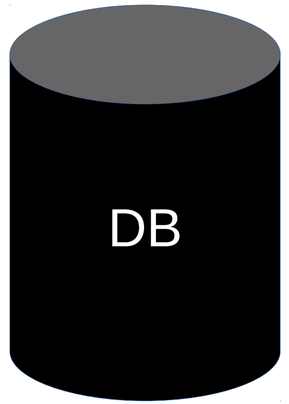


Как ещё сильнее ускорить?

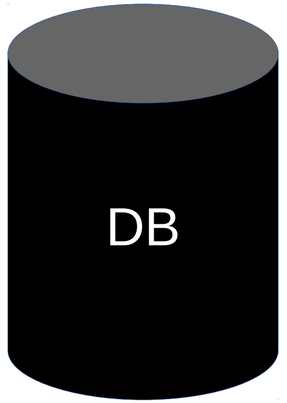
Запрос

aMoney ЦБ

aMoney AliPay

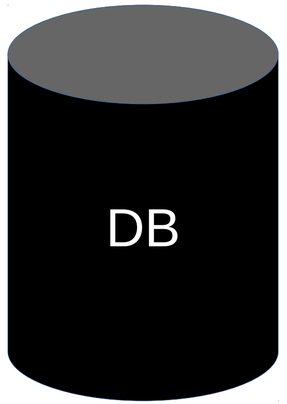


Запрос

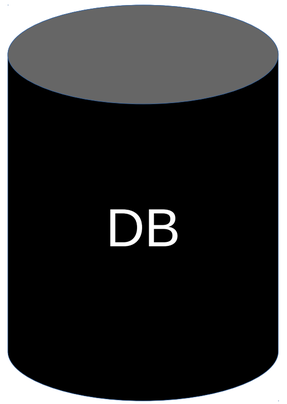


Запрос

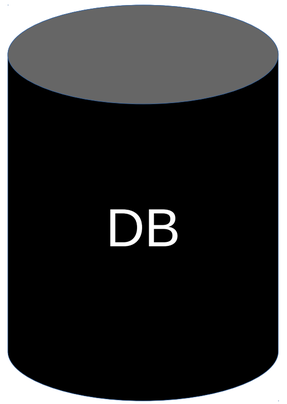
Запрос



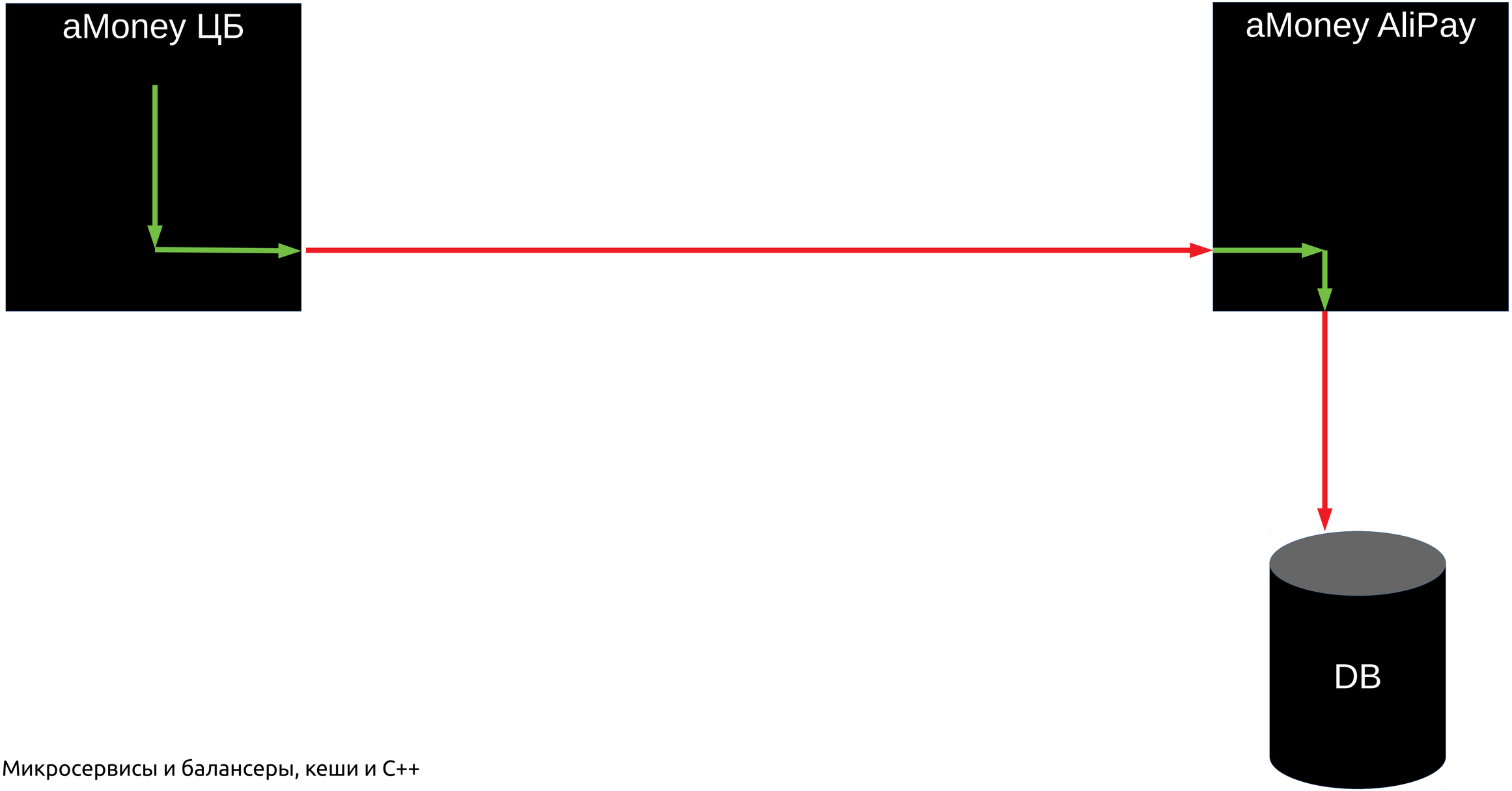
Запрос



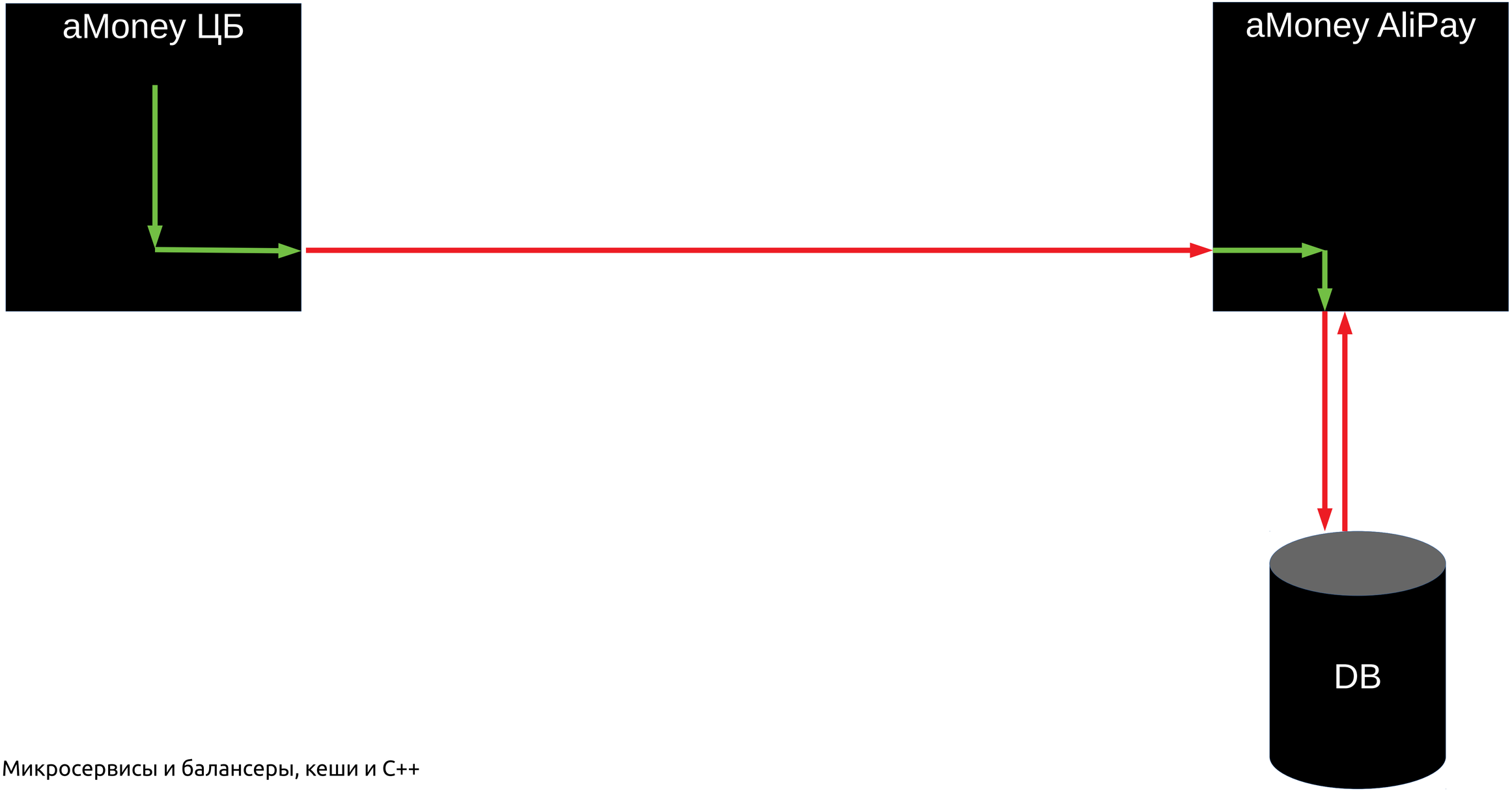
Запрос



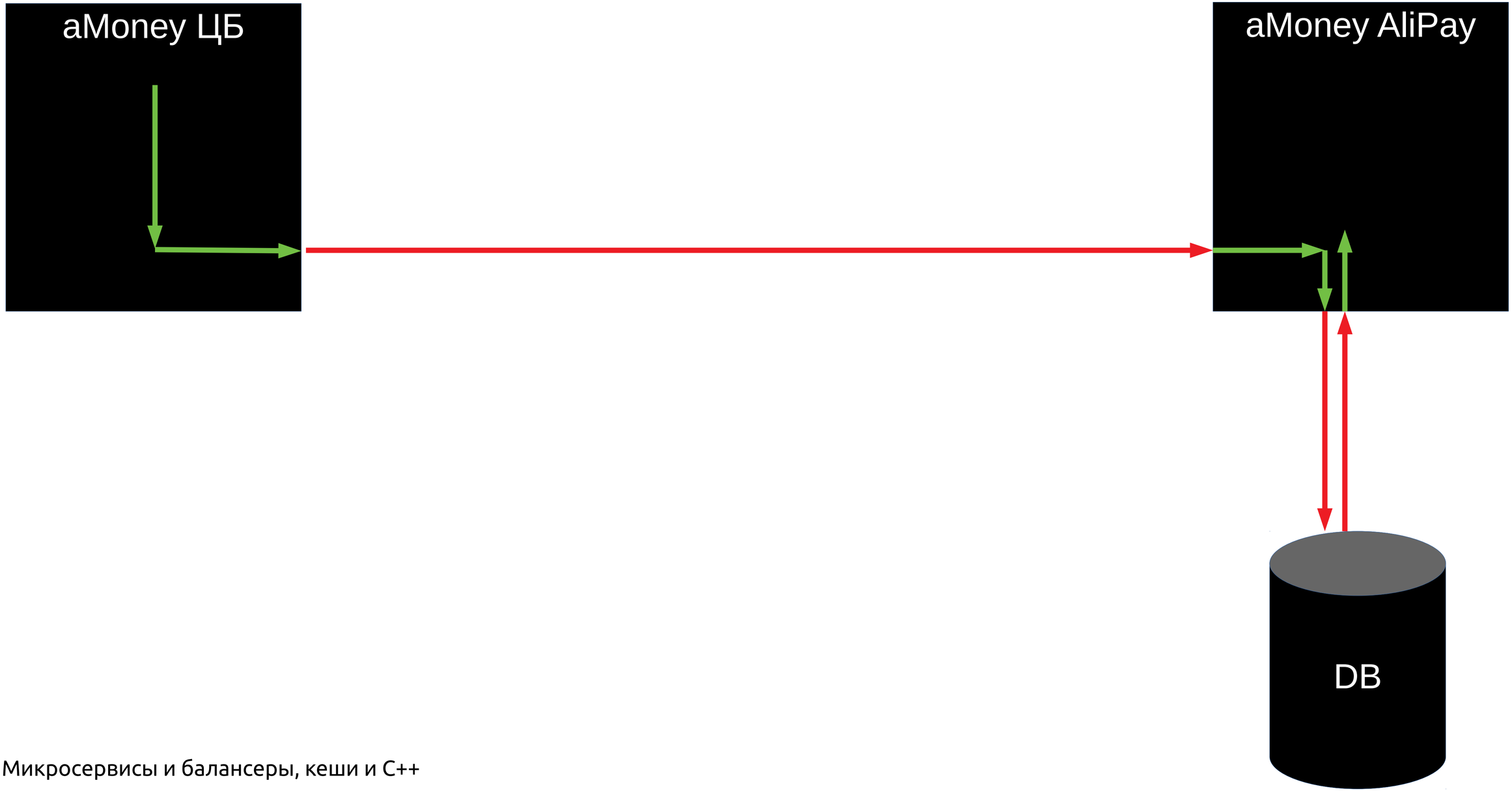
Запрос



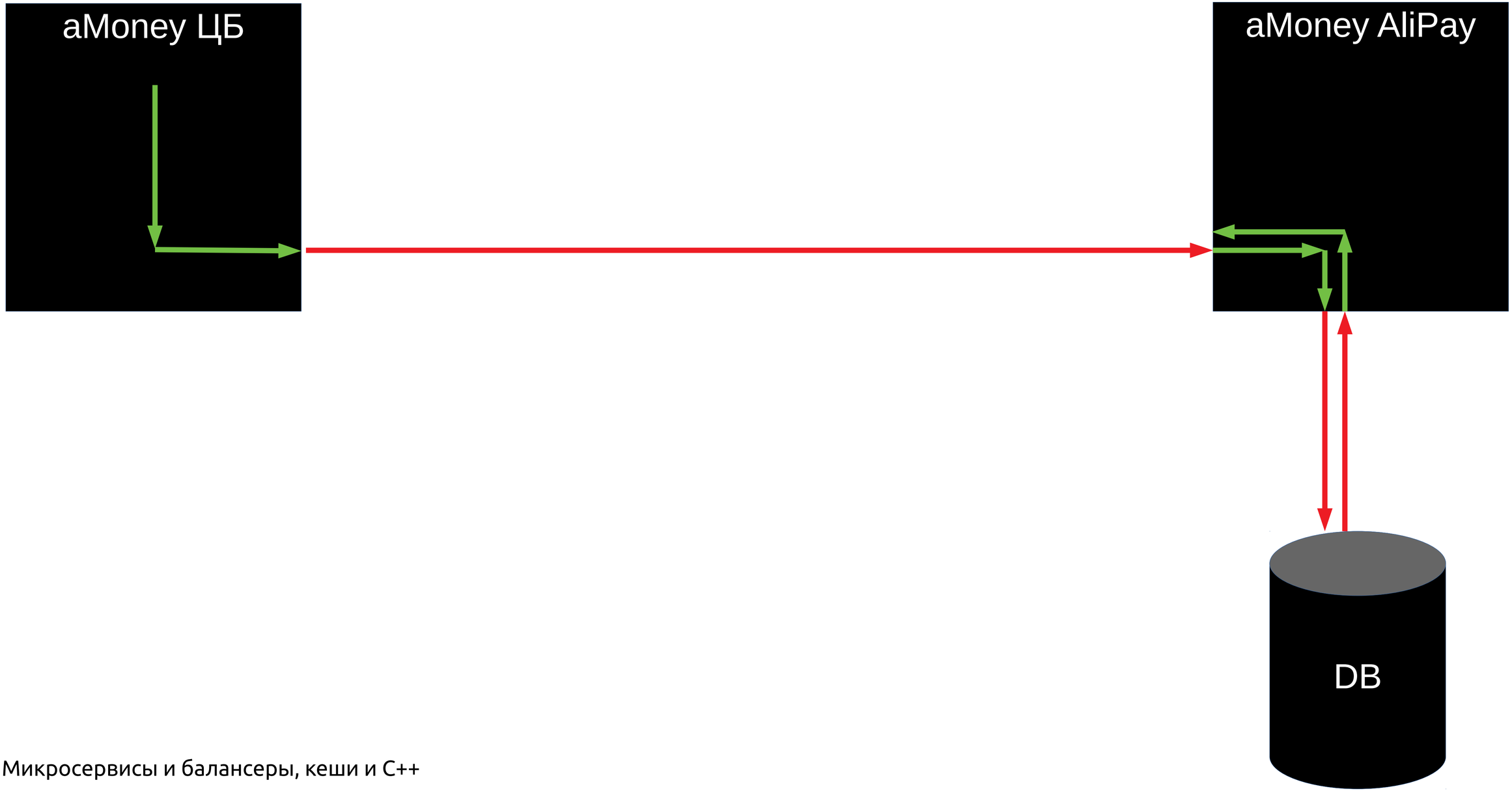
Запрос



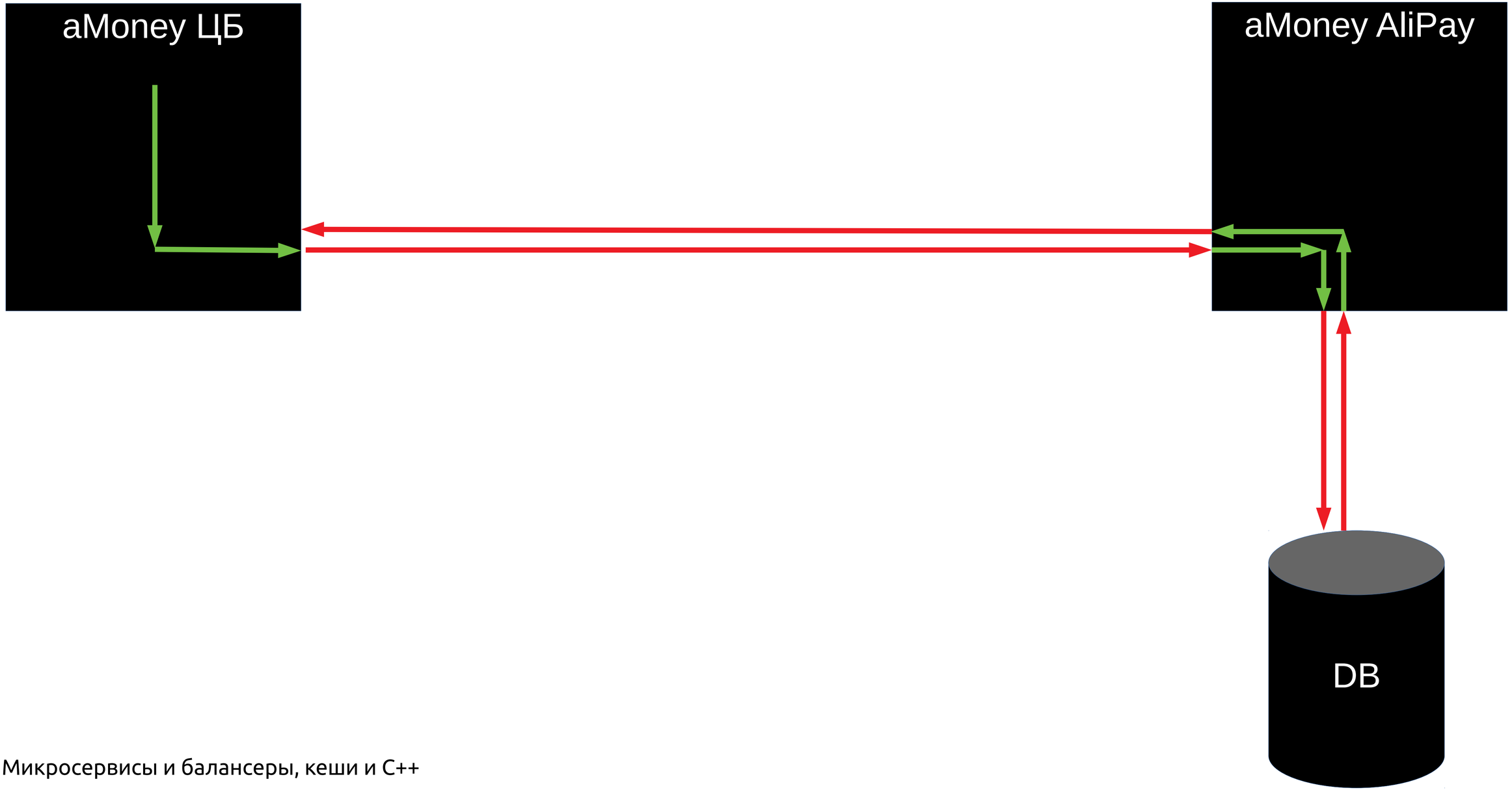
Запрос



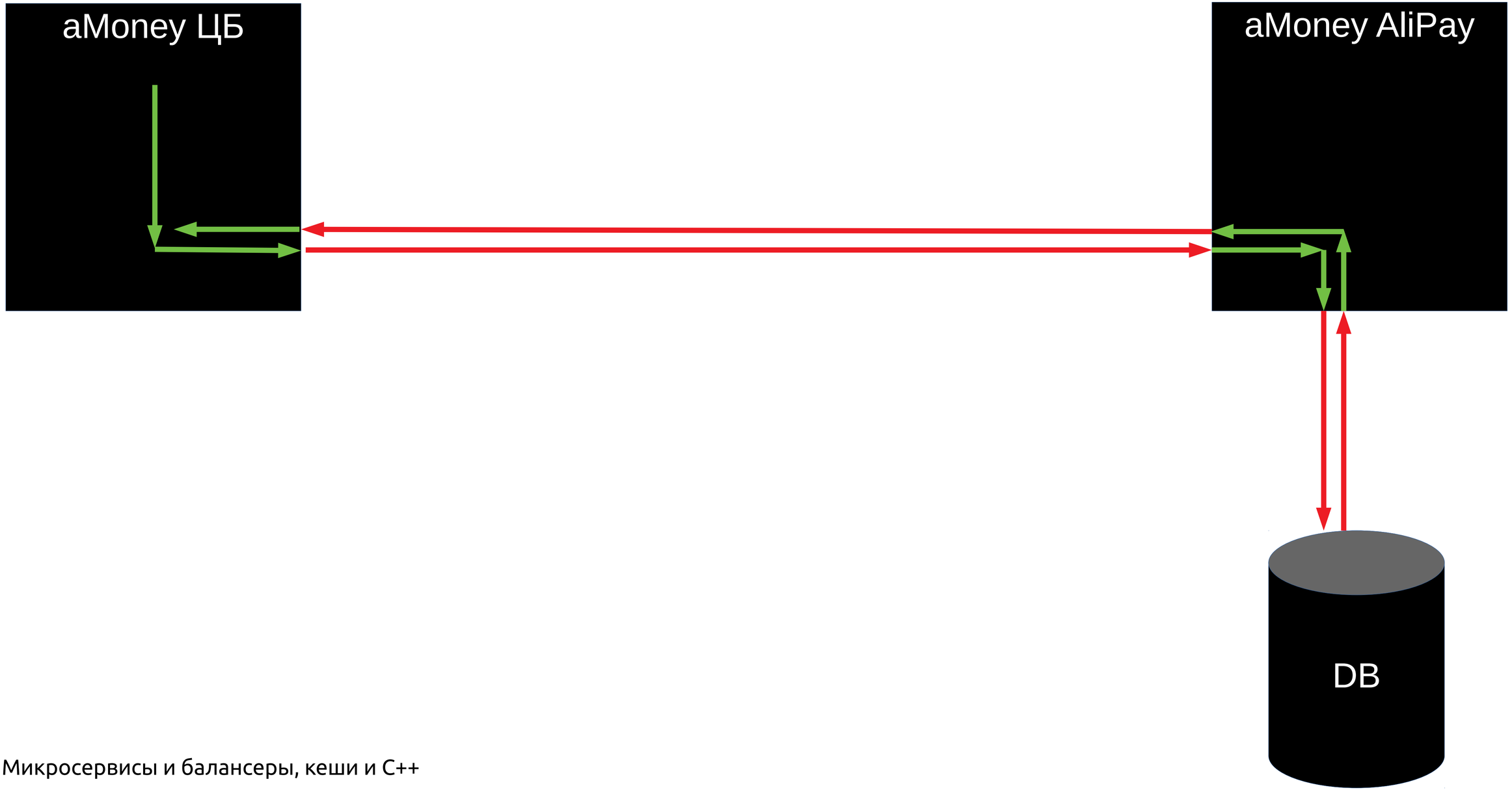
Запрос



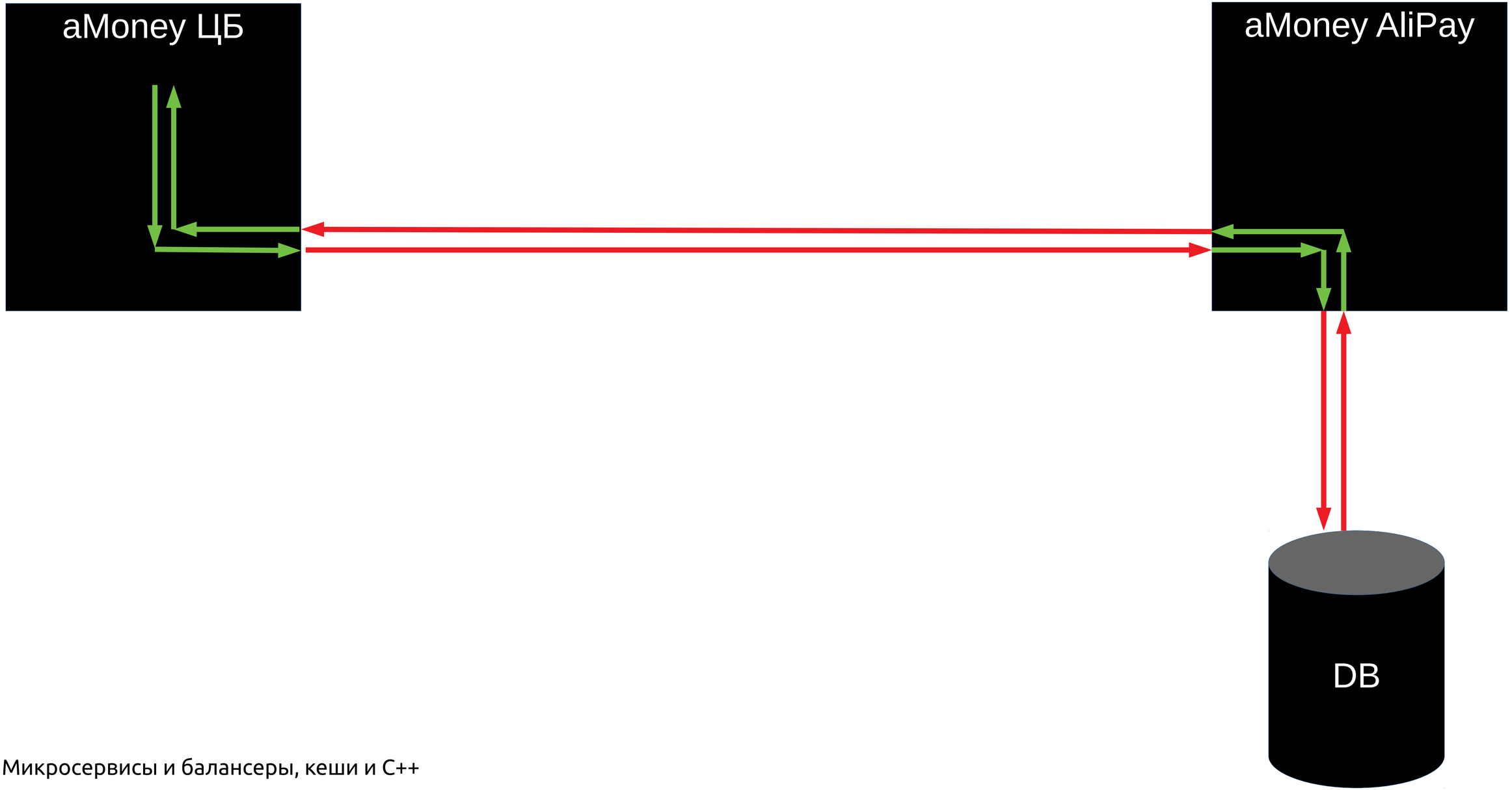
Запрос



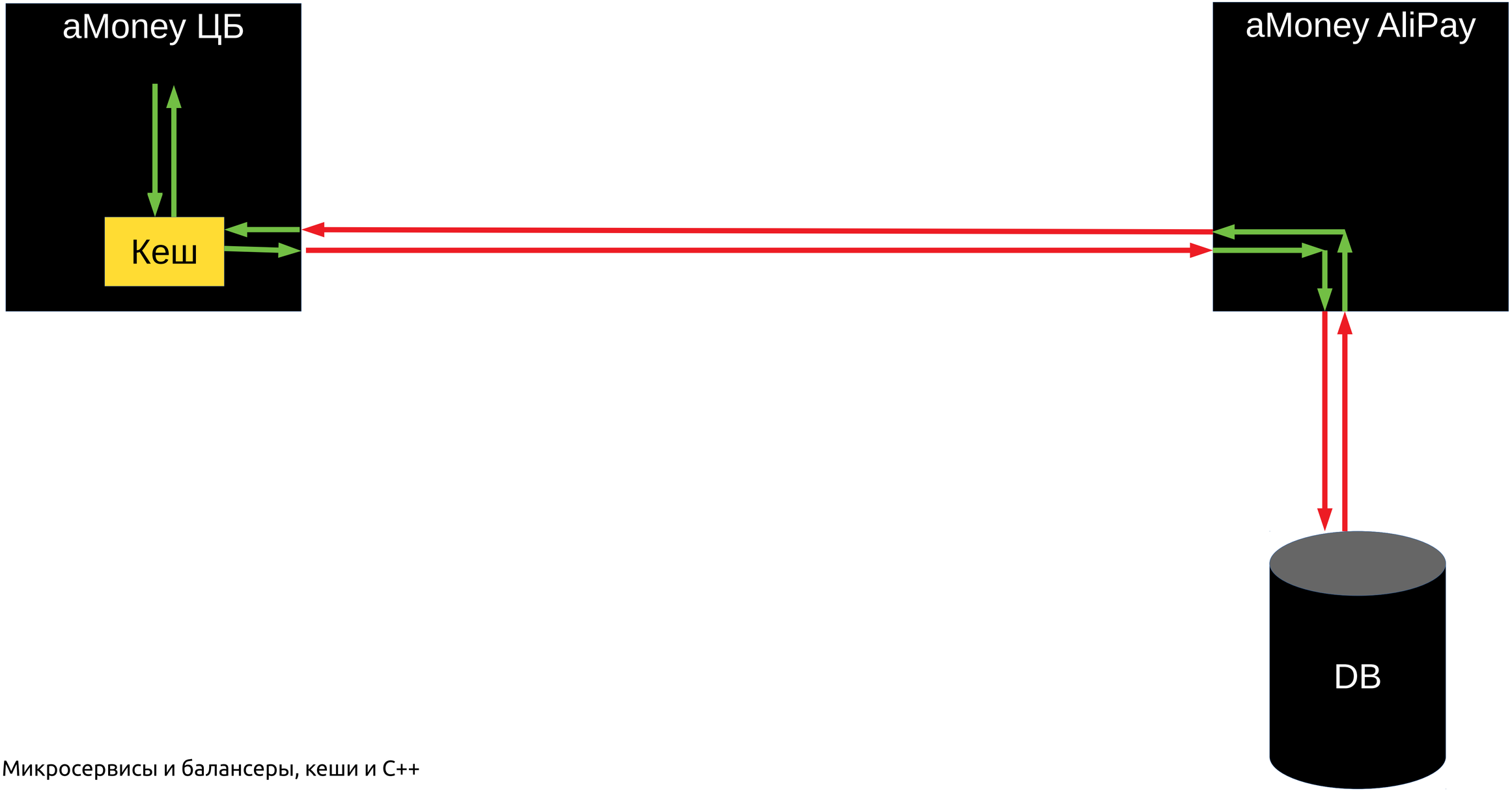
Запрос



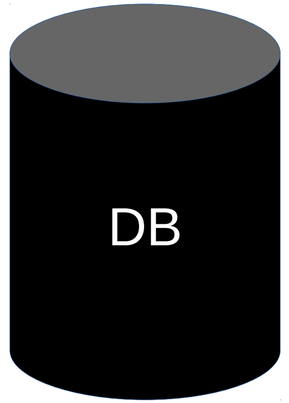
Запрос



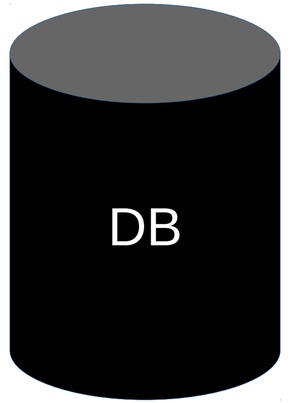
Кеши



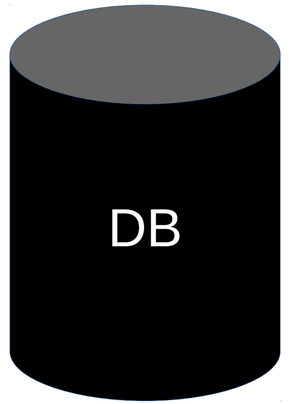
Кеши



Кеши



Кеши



Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность
- Быстрый деплой
- Быстрая сборка
- Тесное общение разработчиков модуля

Минусы:

- Дорогая передача данных между модулями
- Обязательное версионирование и поддержка старых версий
- Большие траты на железо

Плюсы/минусы микросервисов для большой команды

Плюсы:

- Надёжность
- Быстрый деплой
- Быстрая сборка
- Тесное общение разработчиков модуля

Минусы:

- ~~Дорогая передача данных между модулями~~
- Обязательное версионирование и поддержка старых версий
- Большие траты на железо

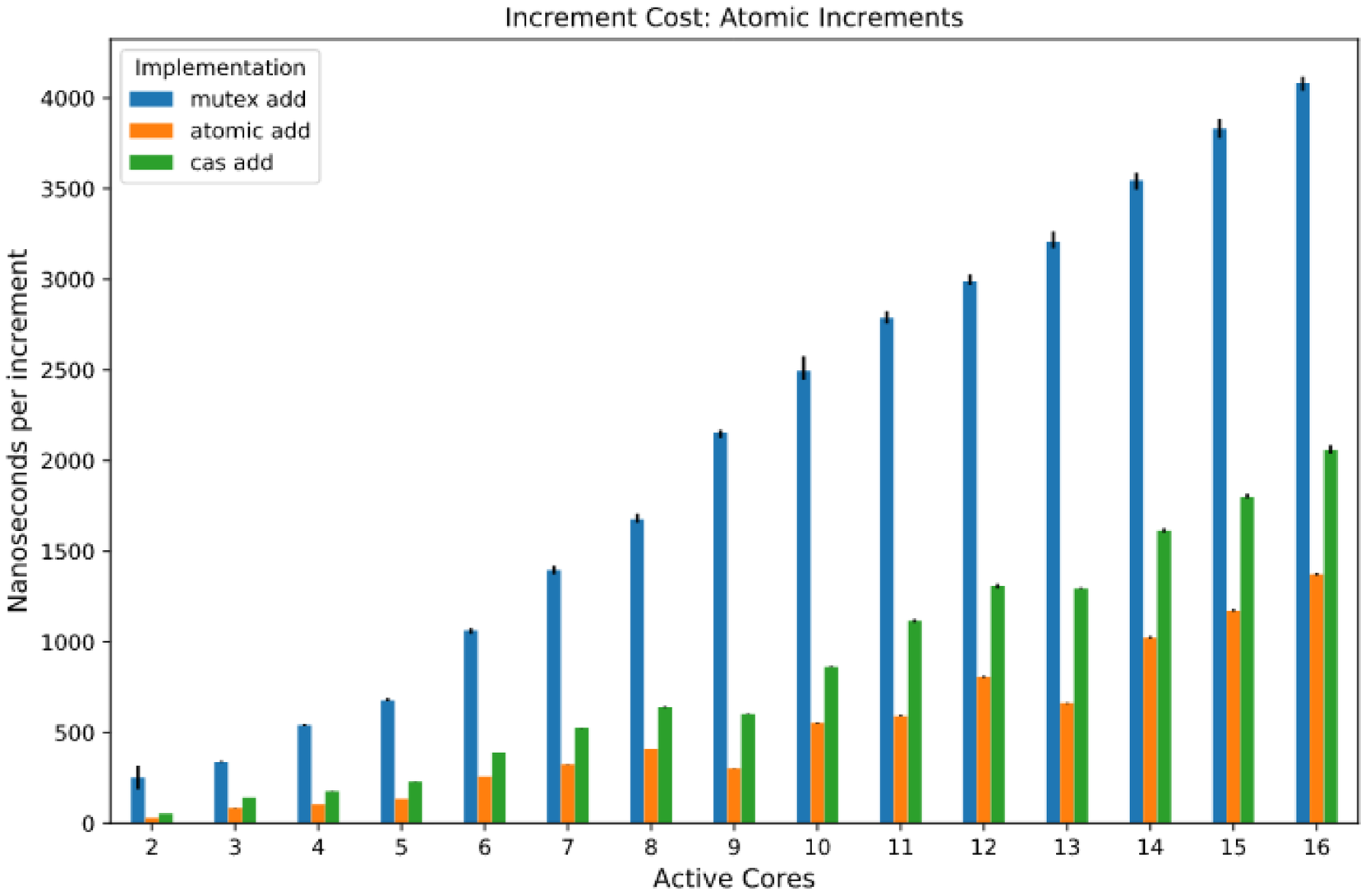
C++ и кеши

Кеши

Кеши

```
std::mutex my_cache_mutex;
std::shared_ptr<const Data> my_cache;
```

Особенности кешей



https://travisdowns.github.io/blog/2020/07/06/concurrency-costs.html

Кеши

Кеши

10kRPS

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms

Как сделать получше?

Кеши

```
std::atomic<std::shared_ptr<const Data>> my_cache;
```

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms
- atomic<shared_ptr>: 1us * 10 000 * 2 == 20ms

Как сделать идеально?

Получше

Получше

Основные тормоза — rmw атомарная операция

Получше

Основные тормоза — rmw атомарная операция

Надо просто её убрать с горячего пути!

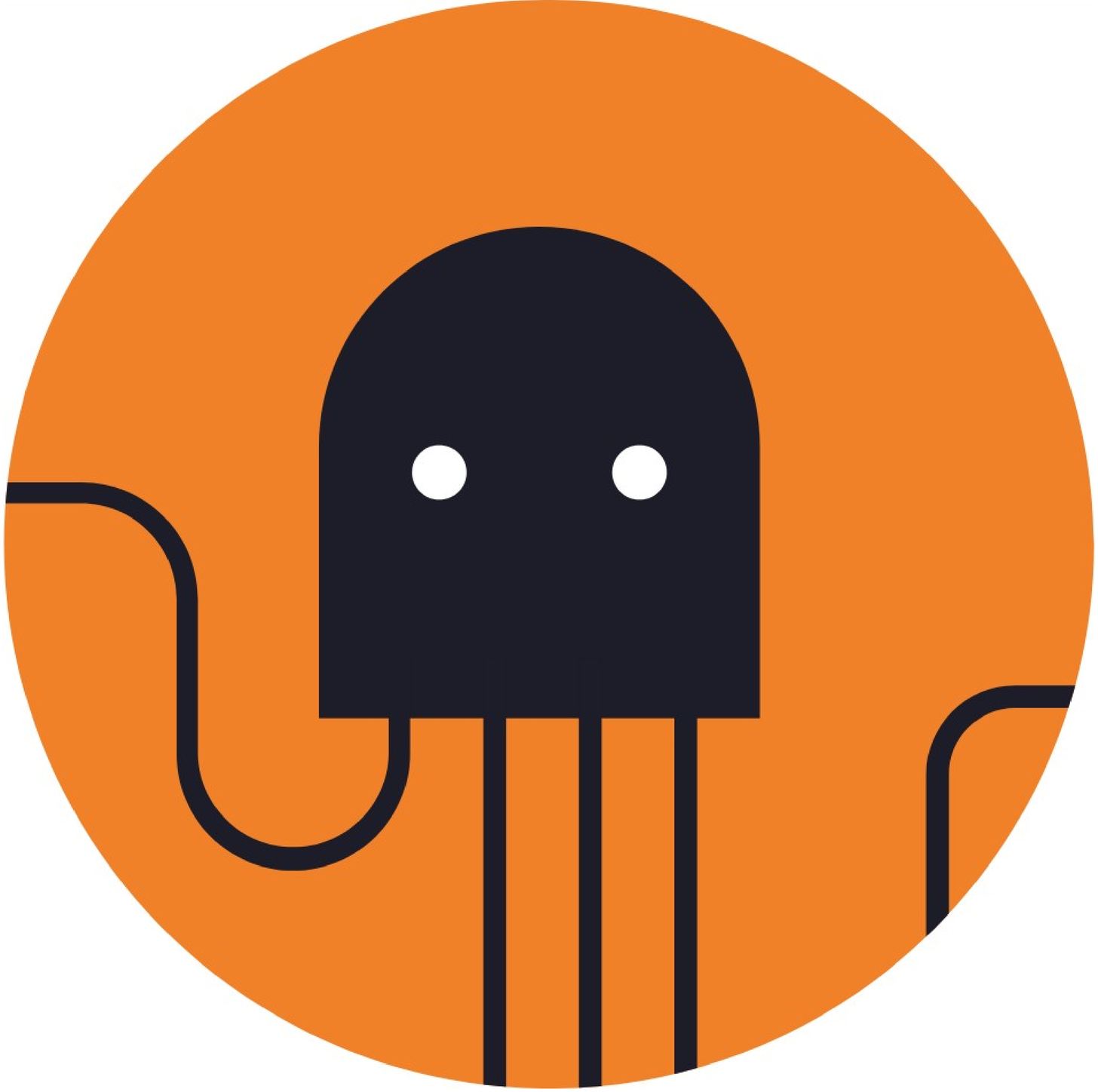
Получше

Основные тормоза — rmw атомарная операция

Надо просто её убрать с горячего пути!

А горячий путь — чтение данных

Hazard Pointer, Concurrency TS 2



Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms
- atomic<shared_ptr>: 1us * 10 000 * 2 == 20ms

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms
- atomic<shared_ptr>: 1us * 10 000 * 2 == 20ms
- rcu: 20ns * 10 000 * 2 == <1ms

Hazard Pointer

Hazard Pointer

Плюсы:

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

- Дорогая запись или обновление данных

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

- Дорогая запись или обновление данных
- Приличные затраты на внутренние нужды

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

- Дорогая запись или обновление данных
- Приличные затраты на внутренние нужды
- В памяти может находиться сразу несколько разных поколений данных

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

- Дорогая запись или обновление данных
- Приличные затраты на внутренние нужды
- В памяти может находиться сразу несколько разных поколений данных

Итог: хорошее решения для задач с количеством чтения данных ≫ записи

Итог

Итого

Итого

- Все архитектуры хороши по своему

Итого

- Все архитектуры хороши по своему
  - Монолит весьма неплох

Итого

- Все архитектуры хороши по своему
  - Монолит весьма неплох
  - Микросервисы тоже норм

Итого

- Все архитектуры хороши по своему
  - Монолит весьма неплох
  - Микросервисы тоже норм
  - Промежуточный шаг может быть в тему

Итого

- Все архитектуры хороши по своему
  - Монолит весьма неплох
  - Микросервисы тоже норм
  - Промежуточный шаг может быть в тему
- Думайте над архитектурой балансеров

Итого

- Все архитектуры хороши по своему
  - Монолит весьма неплох
  - Микросервисы тоже норм
  - Промежуточный шаг может быть в тему
- Думайте над архитектурой балансеров
- Кеши + микросервисы = ❤

Итого

- Все архитектуры хороши по своему
  - Монолит весьма неплох
  - Микросервисы тоже норм
  - Промежуточный шаг может быть в тему
- Думайте над архитектурой балансеров
- Кеши + микросервисы = ❤
- Hazard Pointer — занятная вещь

Спасибо

Полухин Антон

Эксперт-разработчик C++

antoshkka@gmail.com

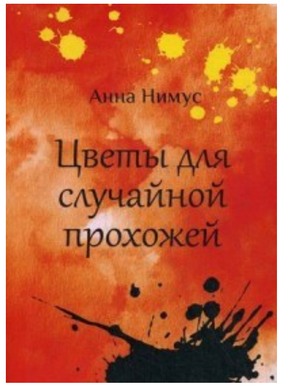
antoshkka@yandex-team.ru

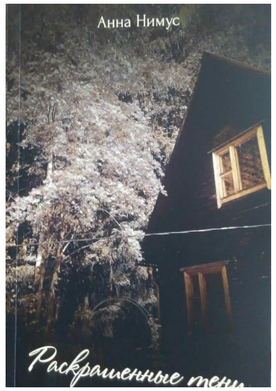
https://github.com/apolukhin

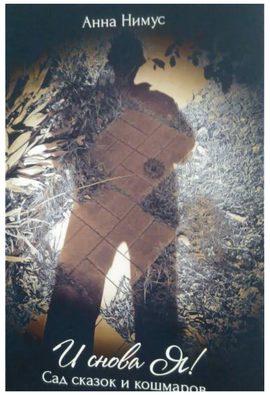
https://stdcpp.ru/

РГ21 C++ РОССИЯ

Спасибо


Анна Нимус
Цветы для случайной прохожей


Анна Нимус


Анна Нимус
И снова Я!
Сад сказок и кошмаров