Использование неблокирующих веб-серверов в highload проектах

Александр Цветков

Eastwood lab

План доклада

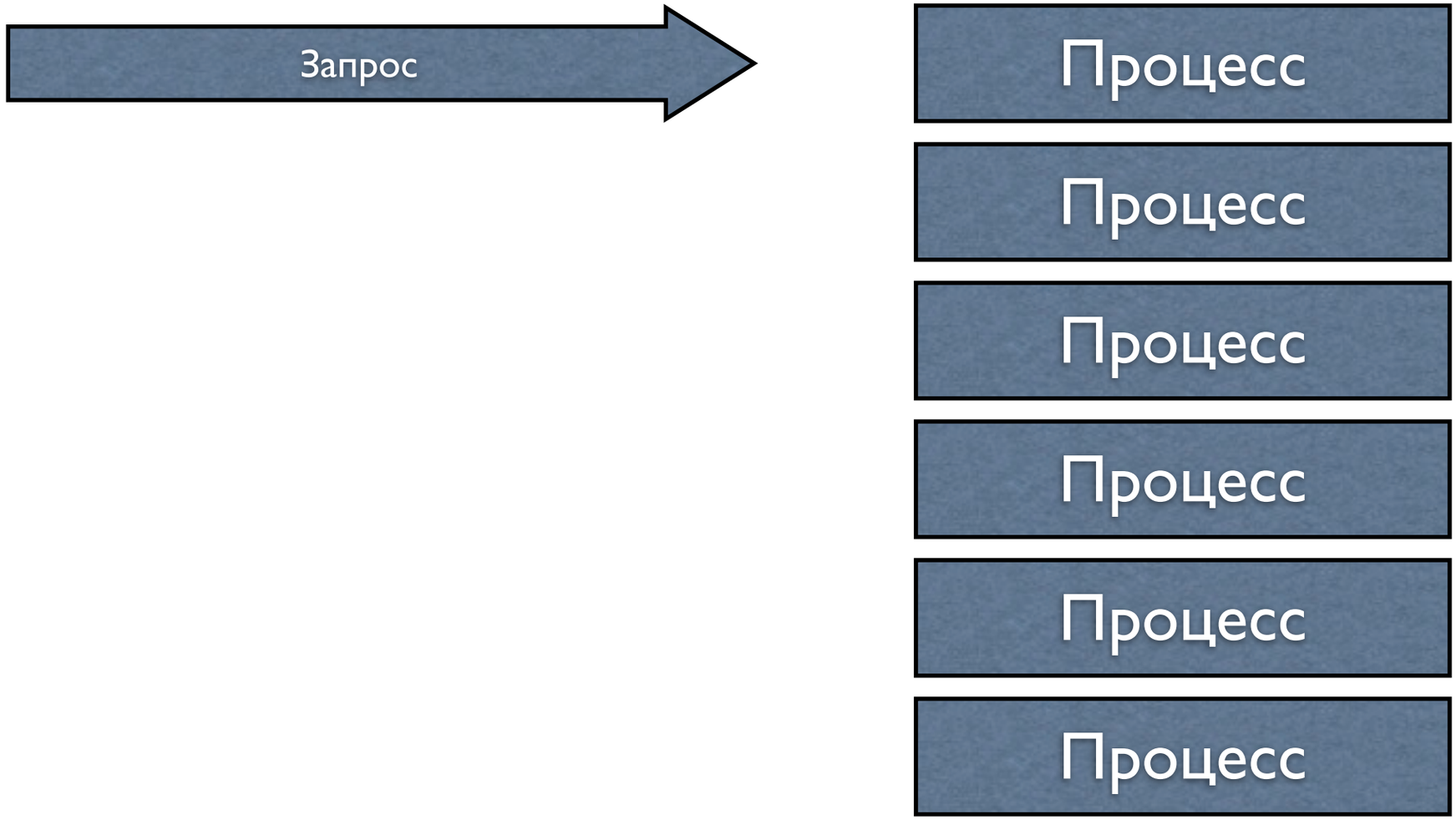
- Что такое async event loop?
- Почему не NodeJS?
- Tornado — асинхронный веб-сервер на Python
- Примениение Tornado в больших реальных проектах на примере Frontik в инфрасруктуре hh.ru

Асинхронная модель веб-сервера

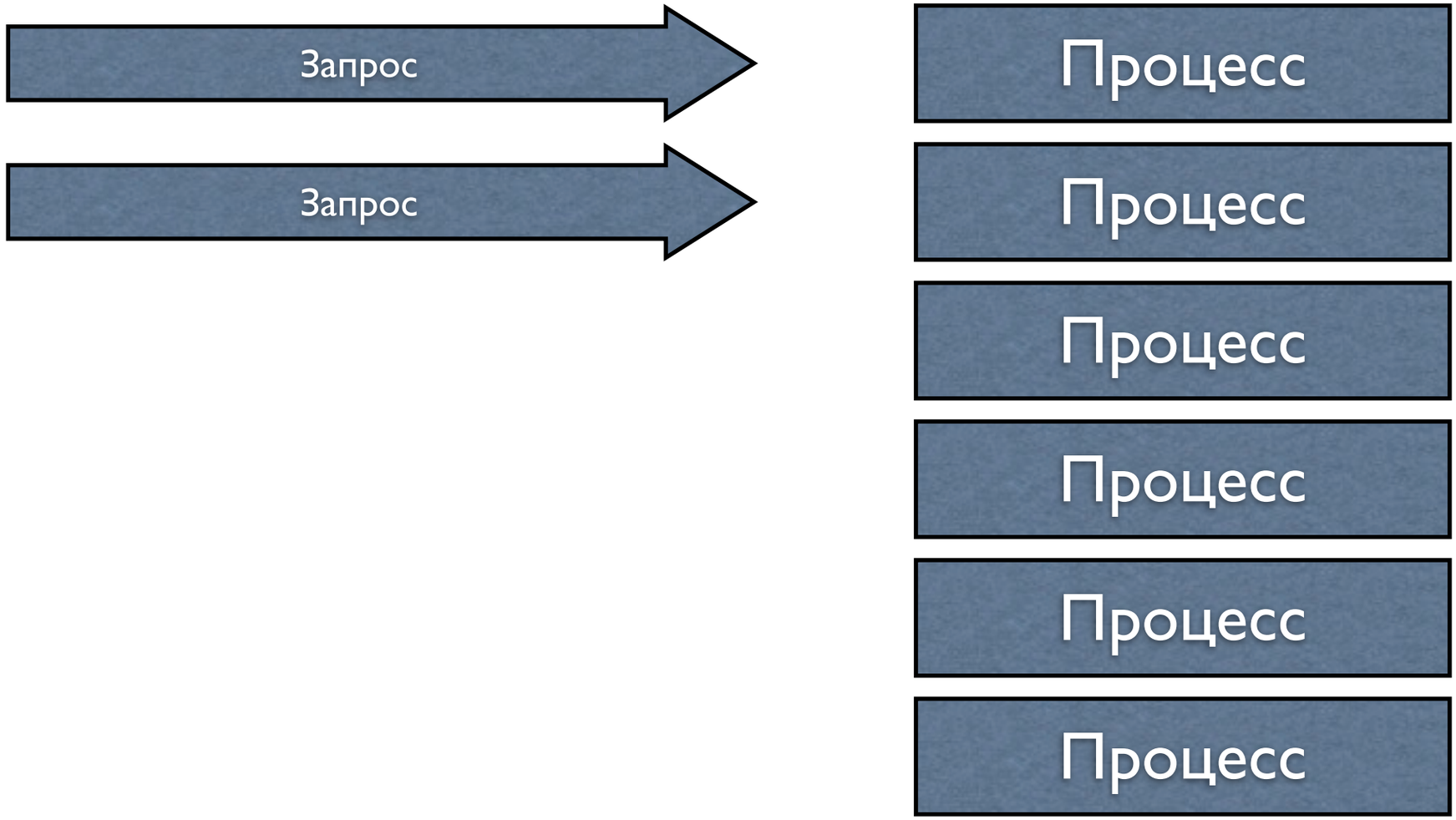
Как устроена, чем отличается от обычной, какие задачи решает.

Как работает обычный веб-сервер

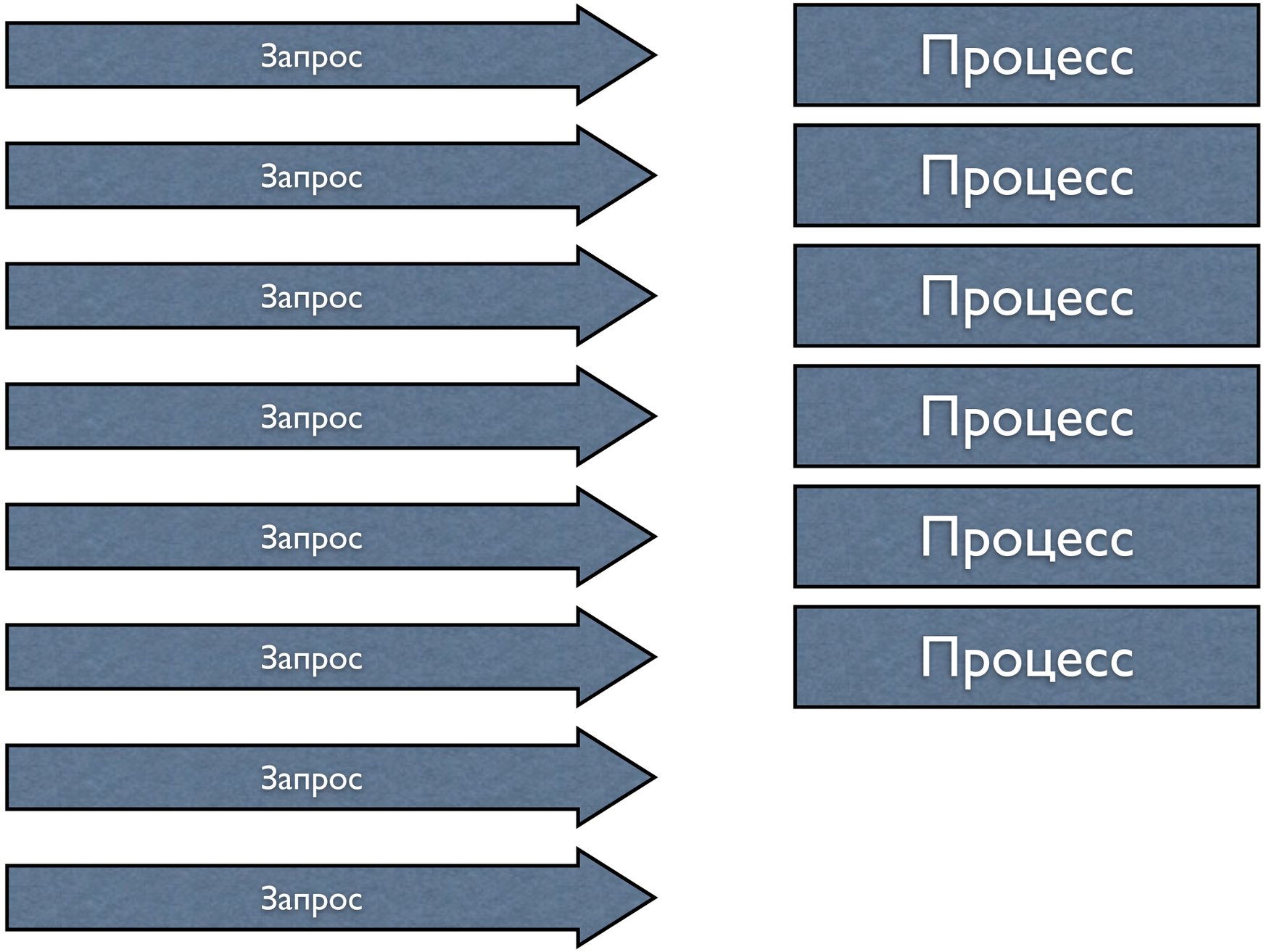
Как работает обычный веб-сервер



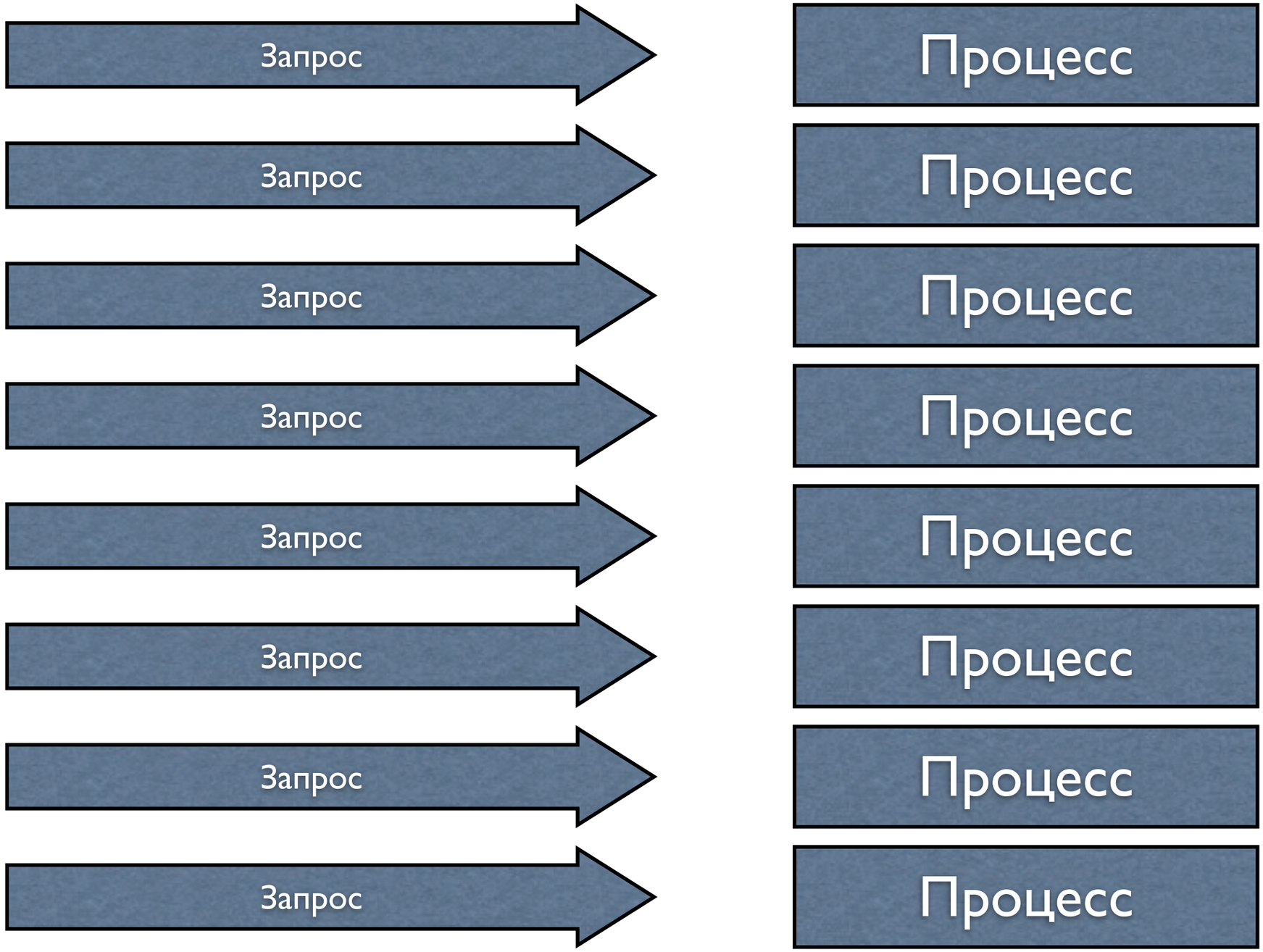
Как работает обычный веб-сервер



Как работает обычный веб-сервер



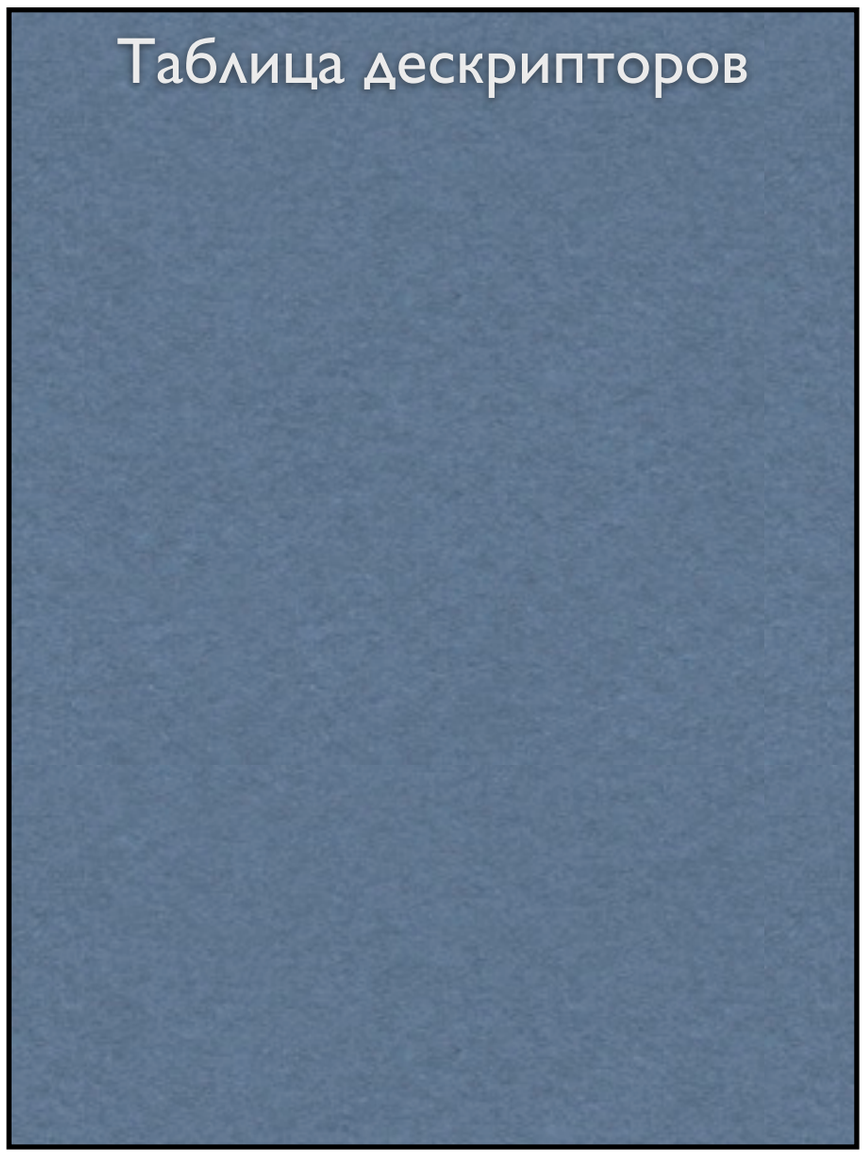
Как работает обычный веб-сервер



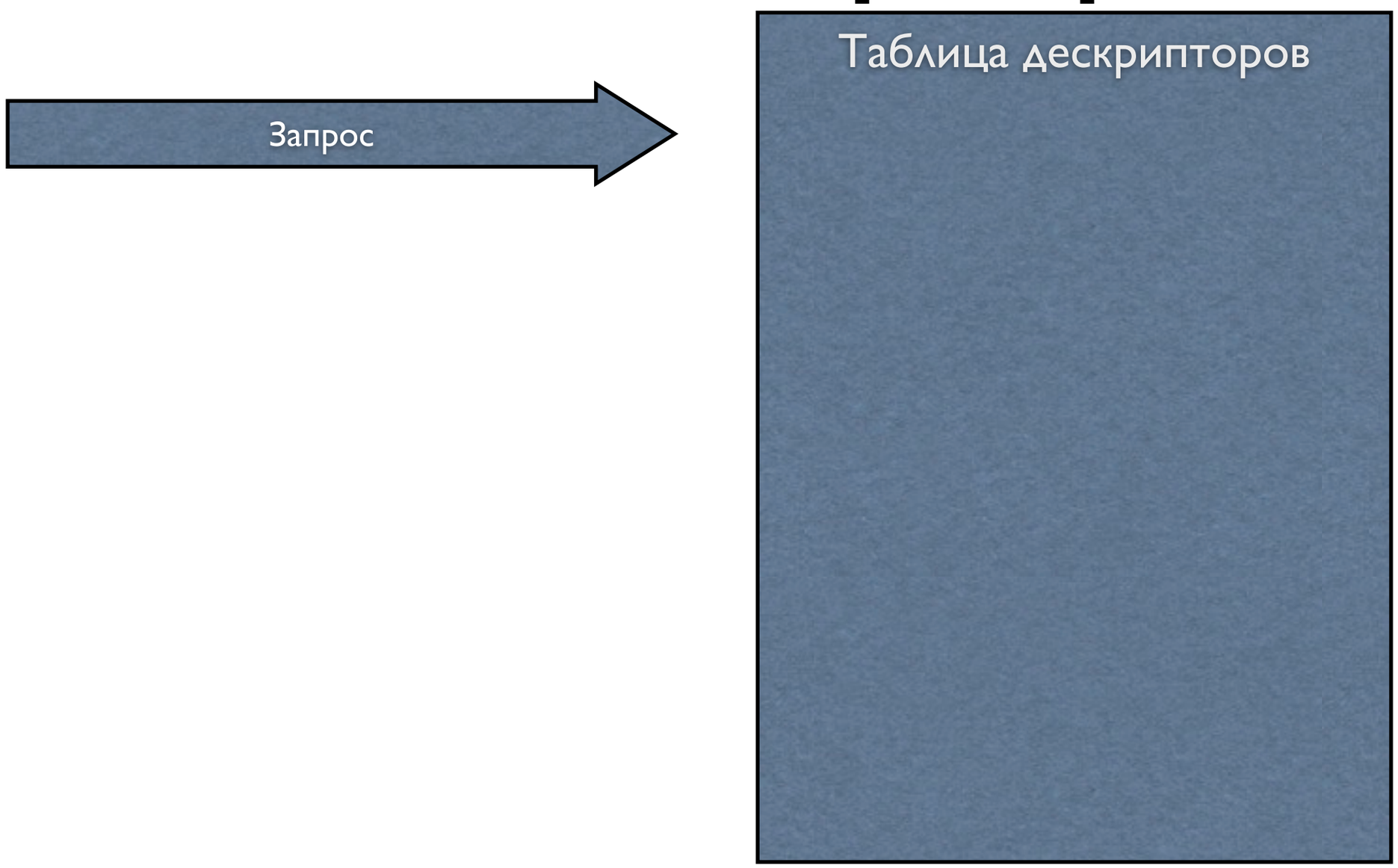
Чем плохо?

- Создание процесса/треда – достаточно дорогостоящая операция
- Выделенные процессу ресурсы часто простаивают

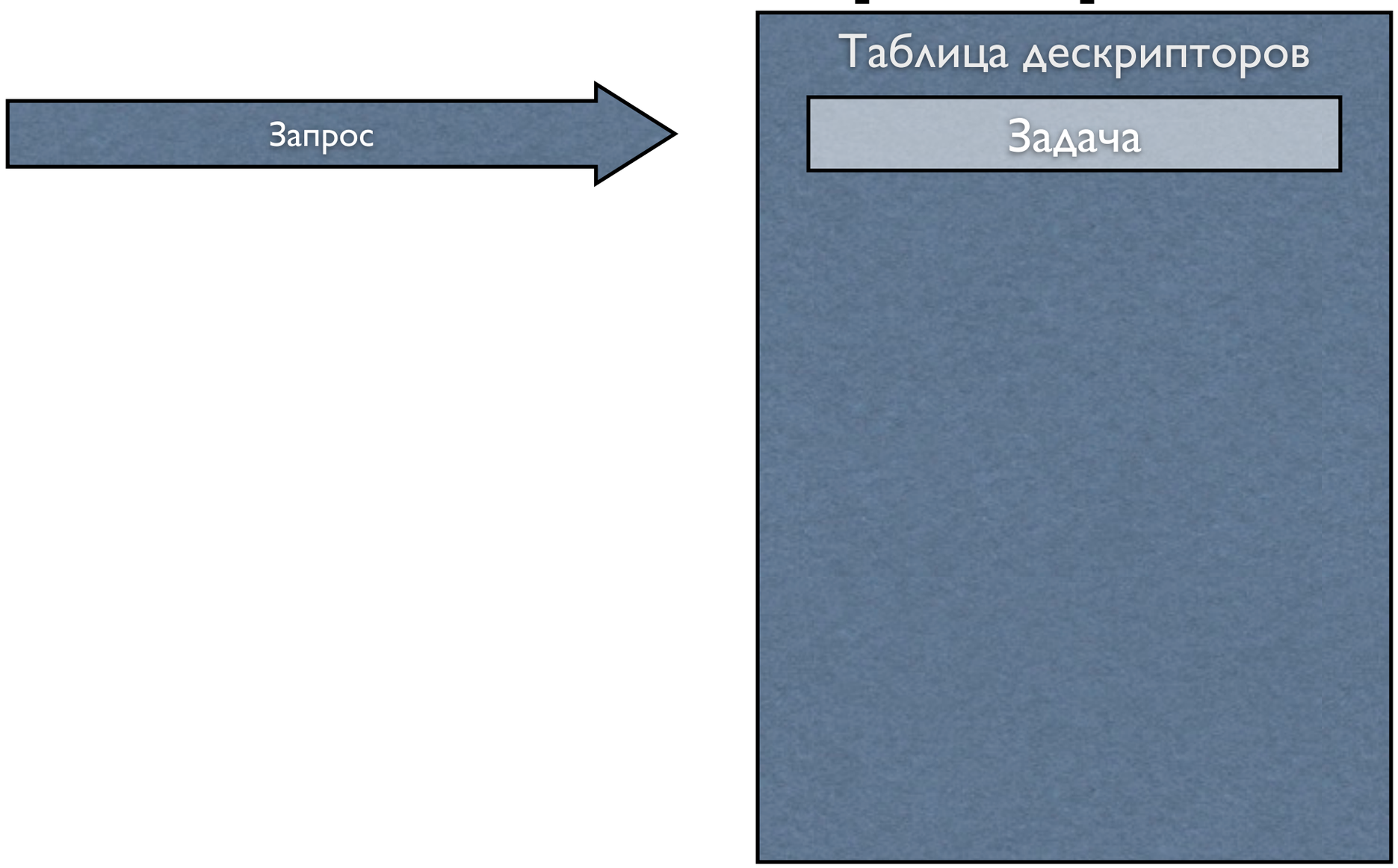
Не блокирующая модель сервера



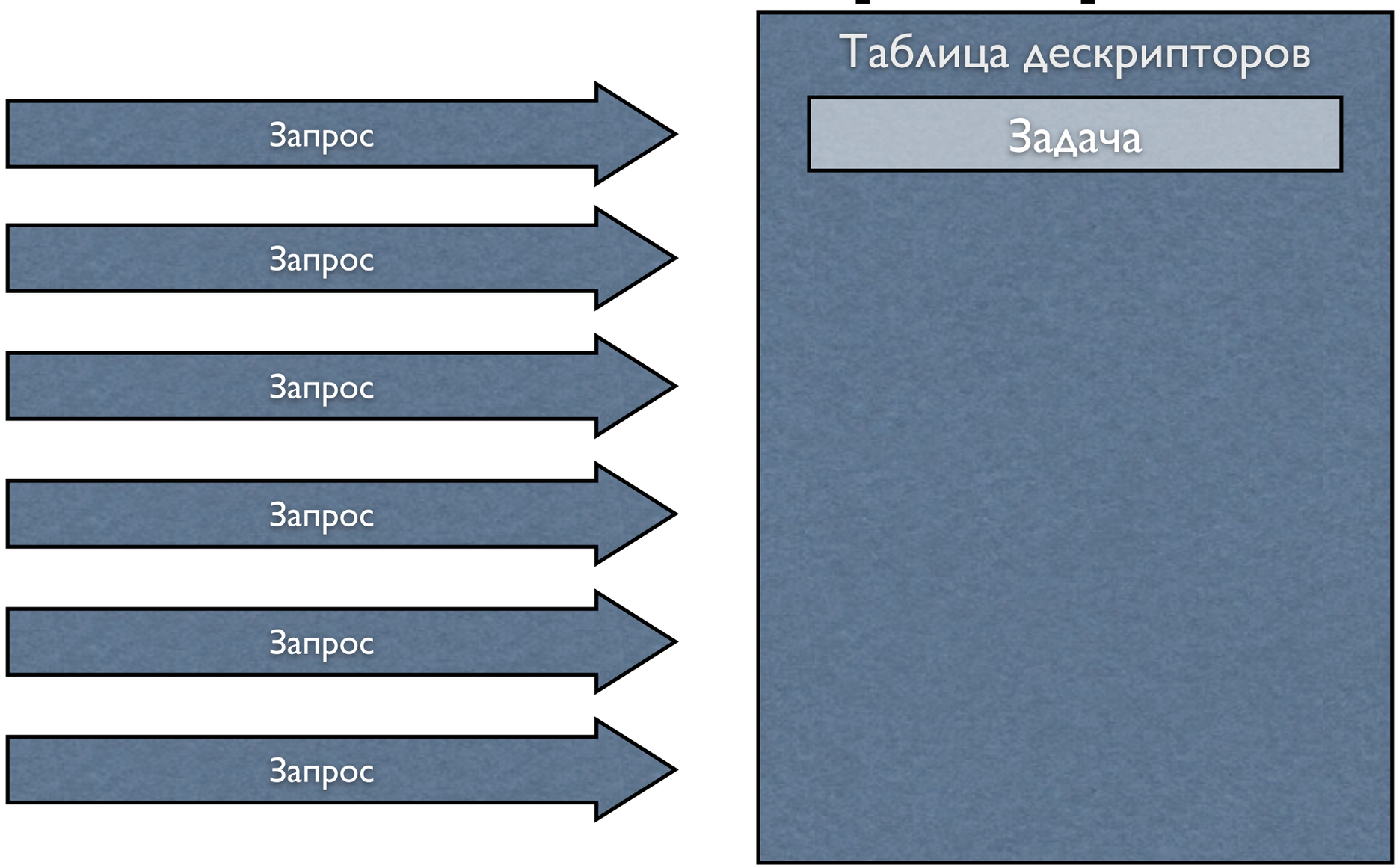
Не блокирующая модель сервера



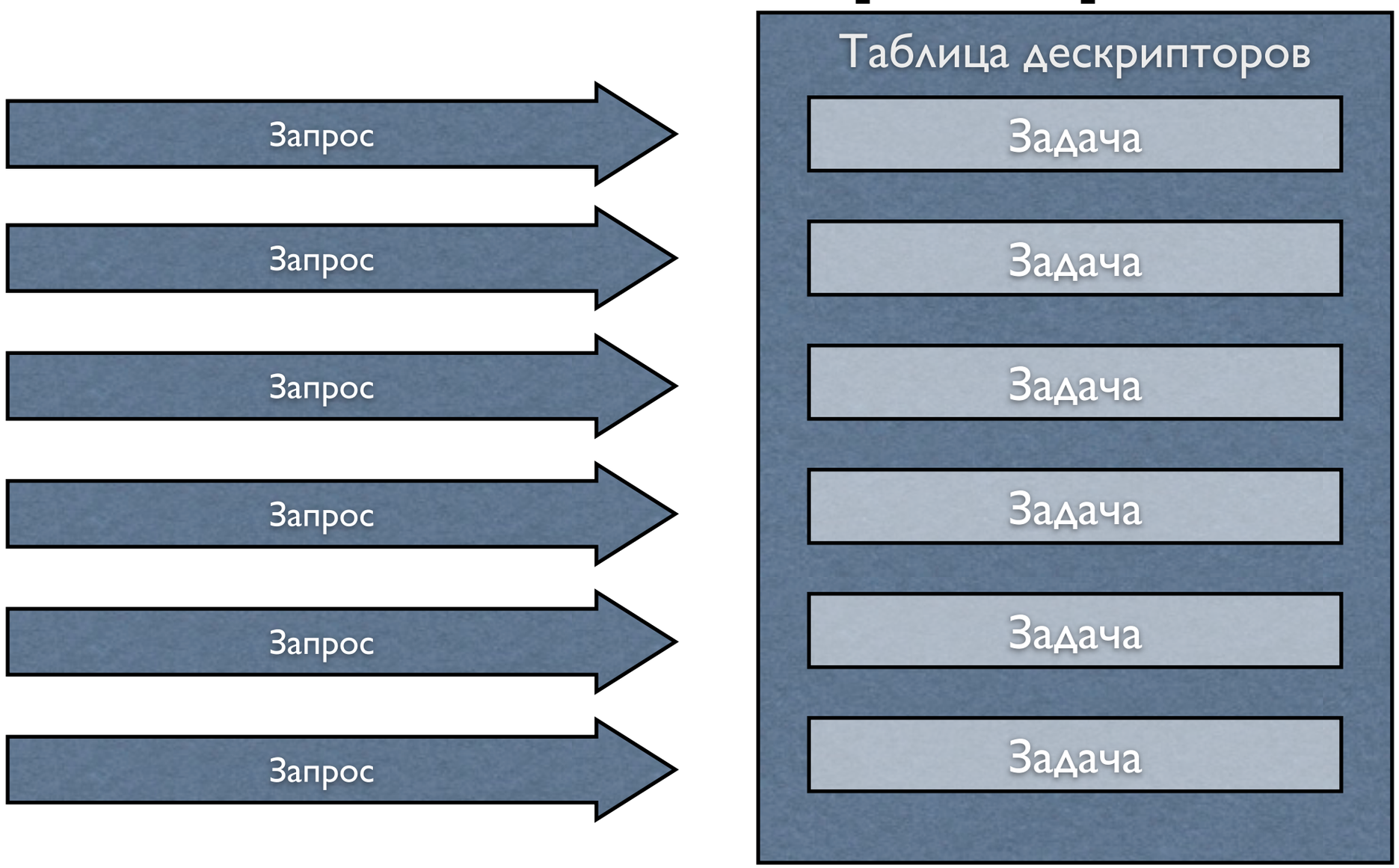
Не блокирующая модель сервера



Не блокирующая модель сервера



Не блокирующая модель сервера



Асинхронный application-сервер

Хорошо подходит

- Интерактивные веб-приложения (websockets, long pooling)
- API - сервисы
- Много независимых задач

Плохо подходит

- Приложения с большим количеством вычислений
- Зависимые друг от друга задачи

Node JS

Первый широко распространённый асинхронный application сервер

Что такое Node JS?

- Полноценный HTTP (и не только) сервер
- Асинхронная не блокирующая модель
- Сценарии на Javascript (движок google V8)

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

Демо

Чем же плох Node JS?

- Нет набора инструментов для быстрой разработки прототипа
- До сих пор активно развивается (сейчас 0.6)
- Javascript - плохой язык для server-side
- До сих пор мало библиотек*

Tornado

Асинхронный web-сервер на Python

Чем хорош Python для асинхронной модели?

- Активно используется в server-side
- Много возможностей для функционального программирования
- Вместе с тем все возможности ООП

Чем хорош Tornado?

- Разработан для FriendFeed, сейчас поддерживается Facebook
- Сценарии на Python
- Мини-фреймворк в комплекте

```
import tornado.ioloop
import tornado.web

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, world")

application = tornado.web.Application([
    (r"/", MainHandler),
])

if __name__ == "__main__":
    application.listen(8888)
    tornado.ioloop.IOLoop.instance().start()
```

Демо

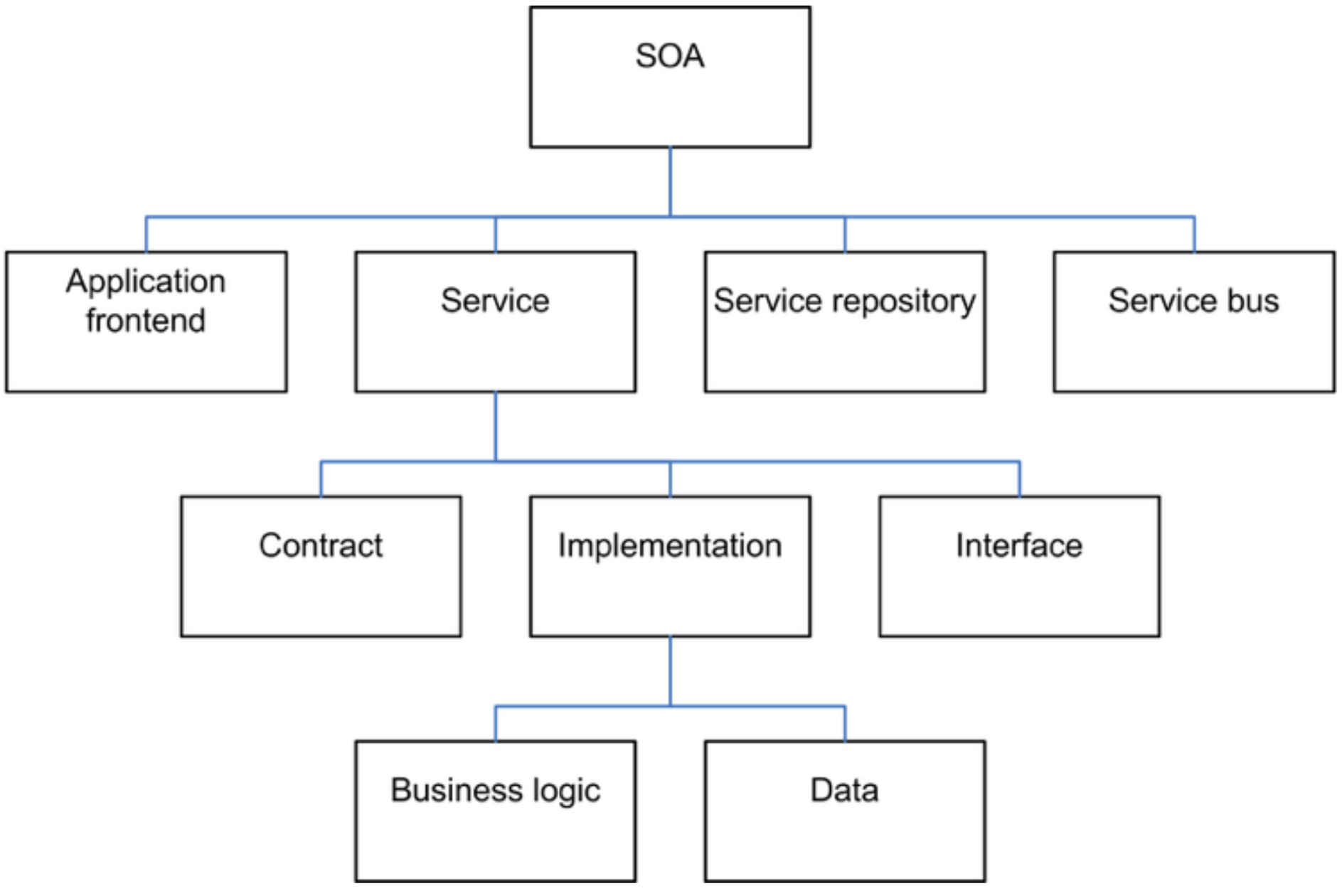
Что нужно учитывать?

- Использовать сторонние библиотеки с осторожностью (весь ввод-вывод должен быть не блокирующим)
- Избегать тяжёлой логики

Frontik

XML агрегатор на основе Tornado.
Применение для построения высоконагруженных сервисов

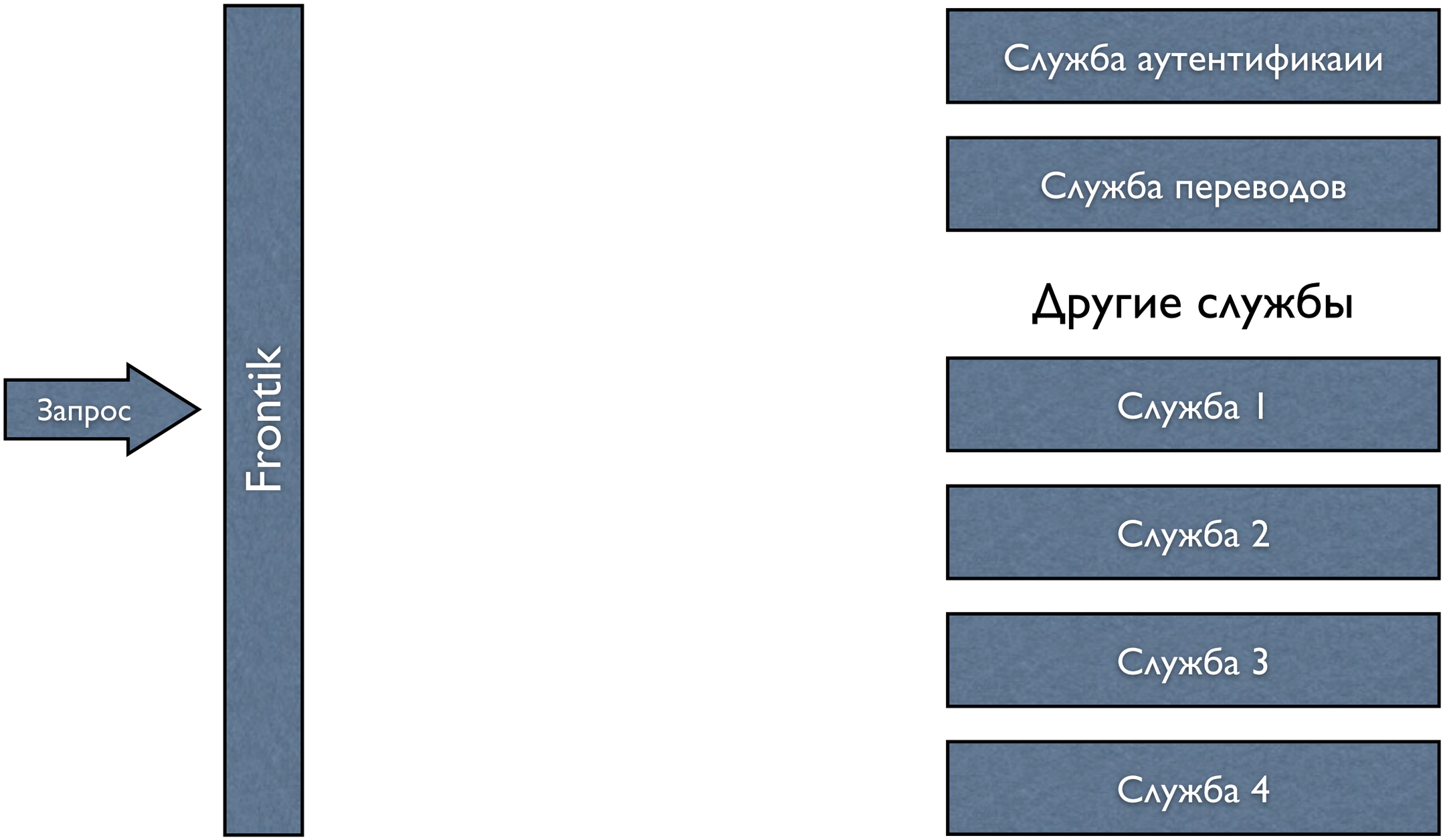
Сервис-ориентированная архитектура



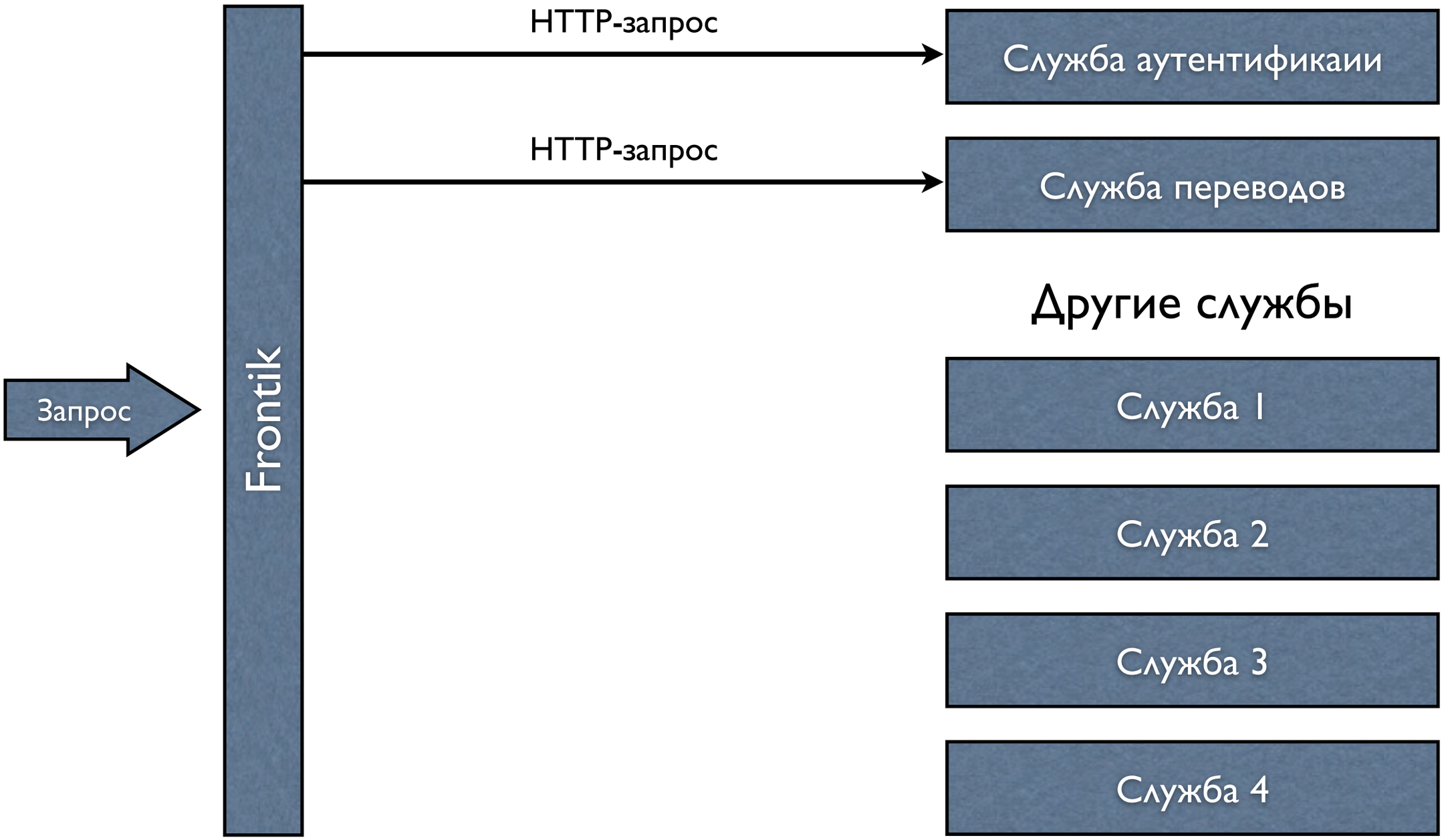
Преущества SOA

- Сервисы слабо связаны (или не связаны вообще)
- Гетерогенная среда, на реализацию сервиса минимум внешних ограничений
- Минимальные затраты на встраивание нового сервиса в архитектуру

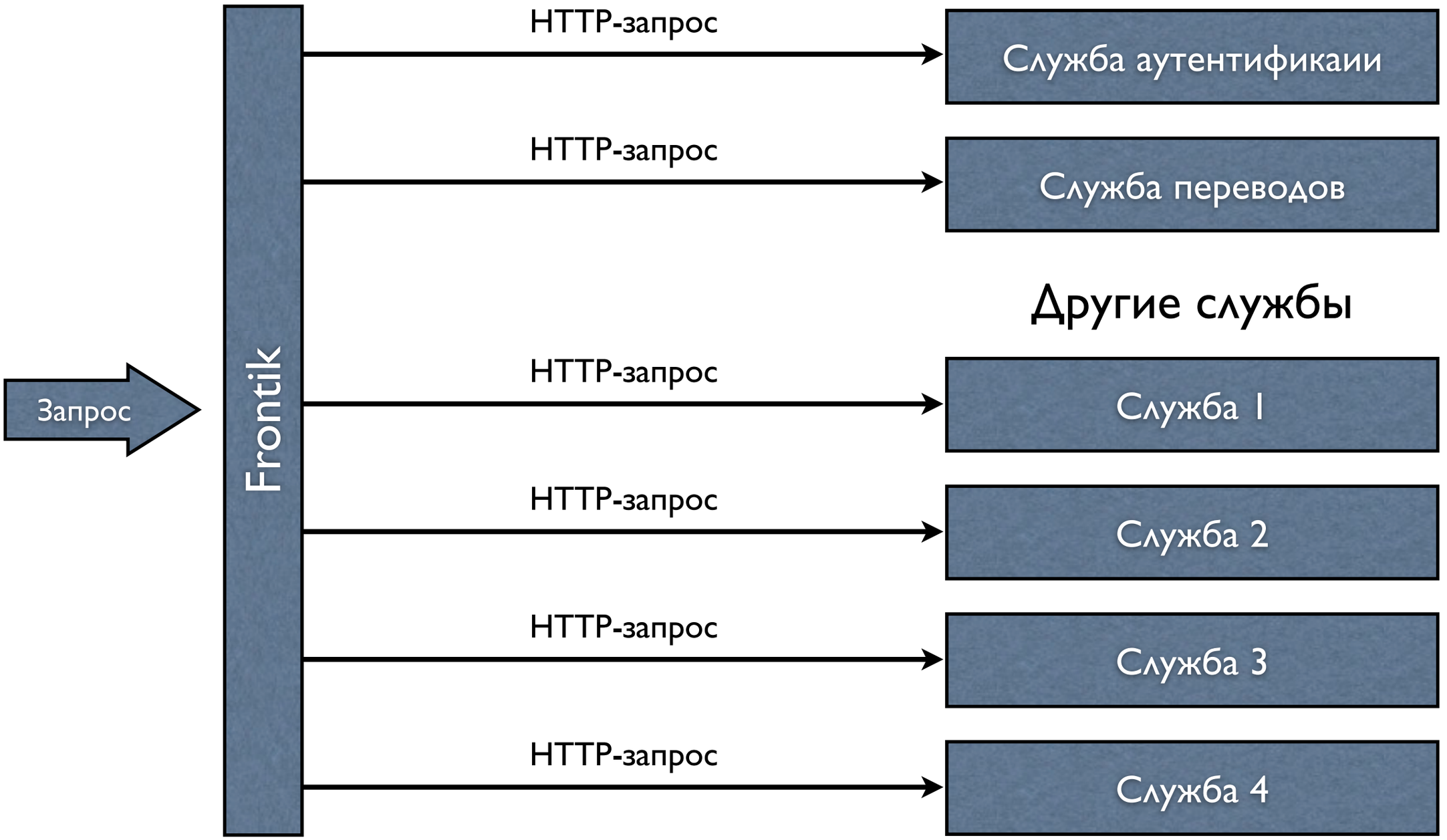
Организация приложения на основе Frontik



Организация приложения на основе Frontik



Организация приложения на основе Frontik



О чём важно помнить

- На frontend должно быть минимум логики
- Даже если логики минимум, блокировки могут возникать (слишком большой xml от сервиса, сложный xsl, и т.д.)

Ресурсы

- Node JS: http://nodejs.org/
- Tornado: http://www.tornadoweb.org
- Frontik: https://github.com/hhru/frontik

Как меня найти?

- Email: sickuenser@gmail.com
- Facebook: http://facebook.com/tsvetkov.al
- Twitter: http://twitter.com/Venya_sick

Вопросы