Создание онлайн-маркетплейса на платформе Micronaut, Kotlin, Java 11+

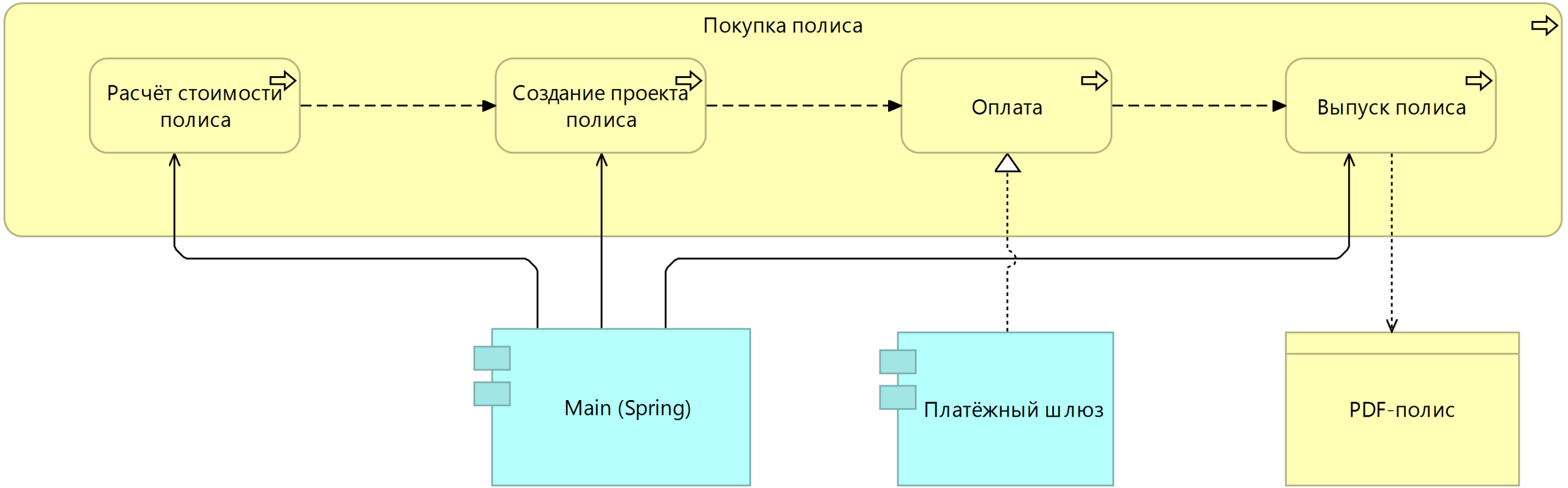
Кочергин Иван
ivan.s.kochergin@gmail.com
Тресоумов Игорь
igortresoumov@gmail.com

Введение

Онлайн-Маркетплейс — платформа электронной коммерции, предоставляющая информацию о продукте или услуге третьих лиц.

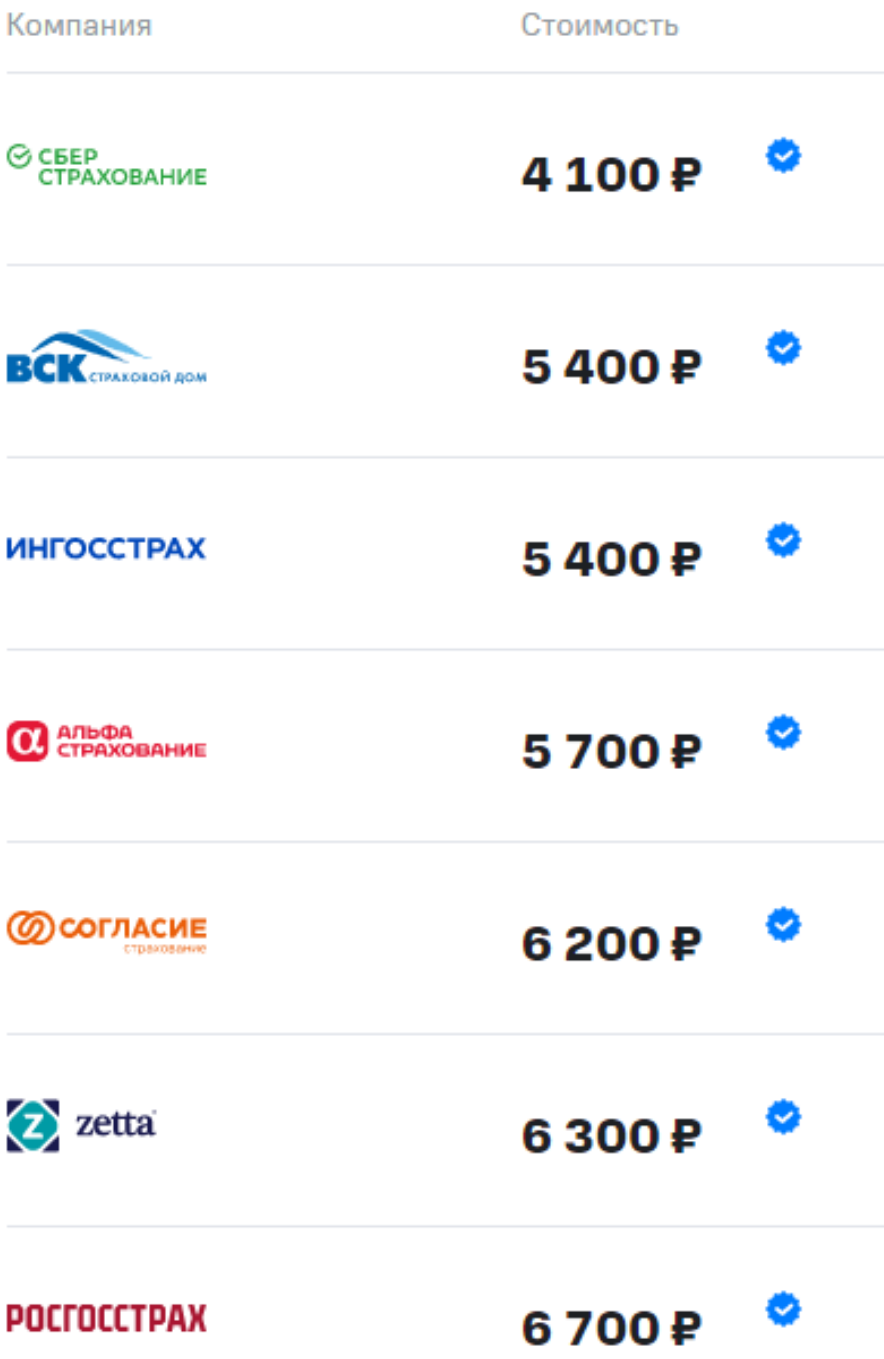
В качестве примера будем использовать маркетплейс страховых продуктов.

Процесс покупки в случае одного поставщика услуг

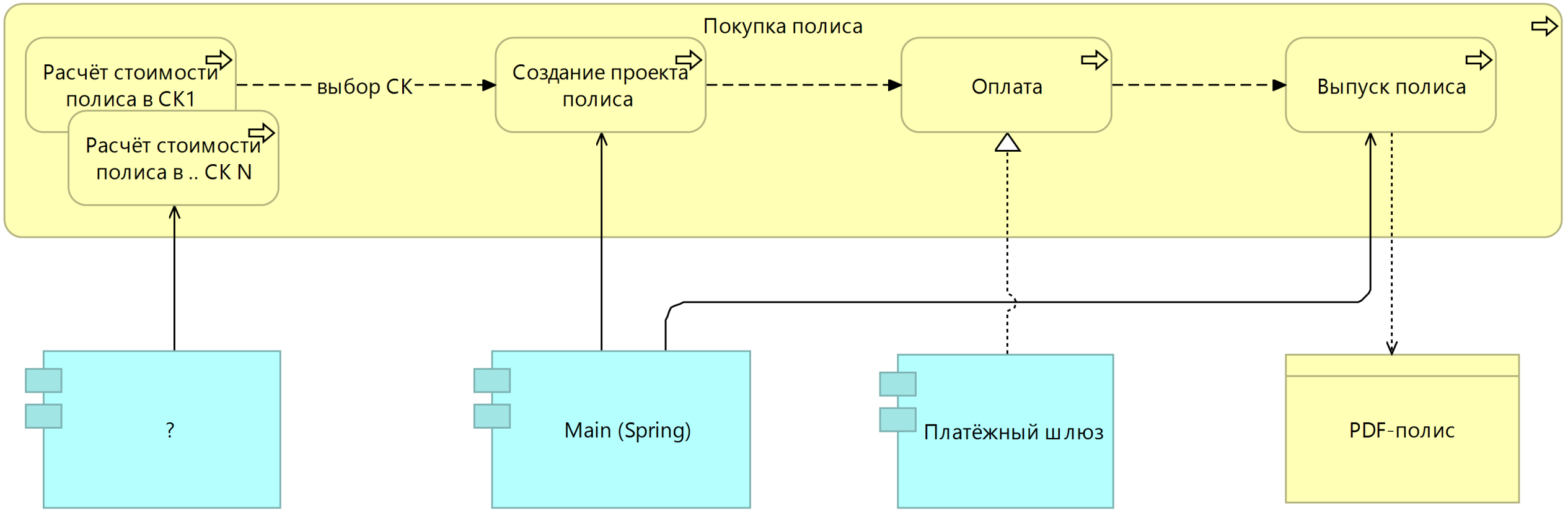


ОСАГО

- 10 крупных участников рынка
- Каждый запрос в Страховую Компанию может длиться до 3 минут



Процесс покупки в случае нескольких поставщиков услуг



Задача

Написать backend приложения, которое сможет совершать максимально возможное количество http-запросов в секунду, каждый из которых может длиться до 3 минут.

Считаем, что frontend у нас есть.

Для обеспечения неблокирующей многопоточности - воспользуемся **Kotlin Coroutines**.
В качестве Framework будем использовать **Micronaut**.

Workshop

https://github.com/koxrel/jpoint-demo-server/tree/master
https://github.com/koxrel/jpoint-demo-server/tree/4_flow

Причины использовать Coroutines

- Сигнатуры методов остаются прежними. Единственное отличие – добавляется модификатор *suspend*
- Пишем код так, как если бы мы писали обычный синхронный код, последовательно. Нет необходимости изучать какой-то особый синтаксис.
- Можно использовать привычные конструкции: циклы, блоки try-catch для обработки исключений и т.д. То есть это просто!

https://kotlinlang.org/docs/async-programming.html#coroutines

Почему выбрали Micronaut*

	Ktor	Micronaut
Поддержка WebSocket	✚	✚
Качественная документация	✚	✚
Похоже на Spring	✖	✚

*Волюнтаризм тех.лида

Небольшая технологическая Δ

Приложение	App 1	App 2	App 3	App 4
Технологии	Java 8, Spring	Java 17, Spring	Java 17, Kotlin, Spring	Java 17, Kotlin, Micronaut
Δ	-	Java 17	Kotlin	Micronaut

Это обеспечивает:

- Низкий порог входа для нового разработчика
- Быстрый выход на рынок (Time-To-Market)
- Низкий отток в команде разработки

Пару слов о WebSocket

Клиенты часто делают несколько расчётов.

Использование WebSocket’ов позволяет не создавать новых соединений.

Можно ли отказаться от json-streaming?

Настройки ОС Linux

Linux:

- net.ipv4.ip_local_port_range = 9000 65535
- ulimit –n 70 000

Docker:

- --sysctl net.ipv4.ip_local_port_range="9000 65000"
- --ulimit nofile=70000:70000

Замеры производительности*

- 31 rps
- 5600 пользователей в моменте

	Потоки	http-соединения
Корутины	20	56 535
Потоки	1352	1352

*При условии, что один клиент создаёт 10 сетевых соединений, каждое из которых длится 180 секунд.

SOAP

Spring-WS	Kotlin-WS
Apache HttpComponents HttpClient	Java 11 httpClient

Полученные артефакты

- Micronaut 3.2.1 Bug Fix
  https://github.com/micronaut-projects/micronaut-core/issues/6582
  https://github.com/micronaut-projects/micronaut-core/releases/tag/v3.2.1

- Kotlin-WS
  В процессе публикации

Спасибо за внимание!

Кочергин Иван
ivan.s.kochergin@gmail.com
Тресоумов Игорь
igortresoumov@gmail.com