ТИНЬКОФФ

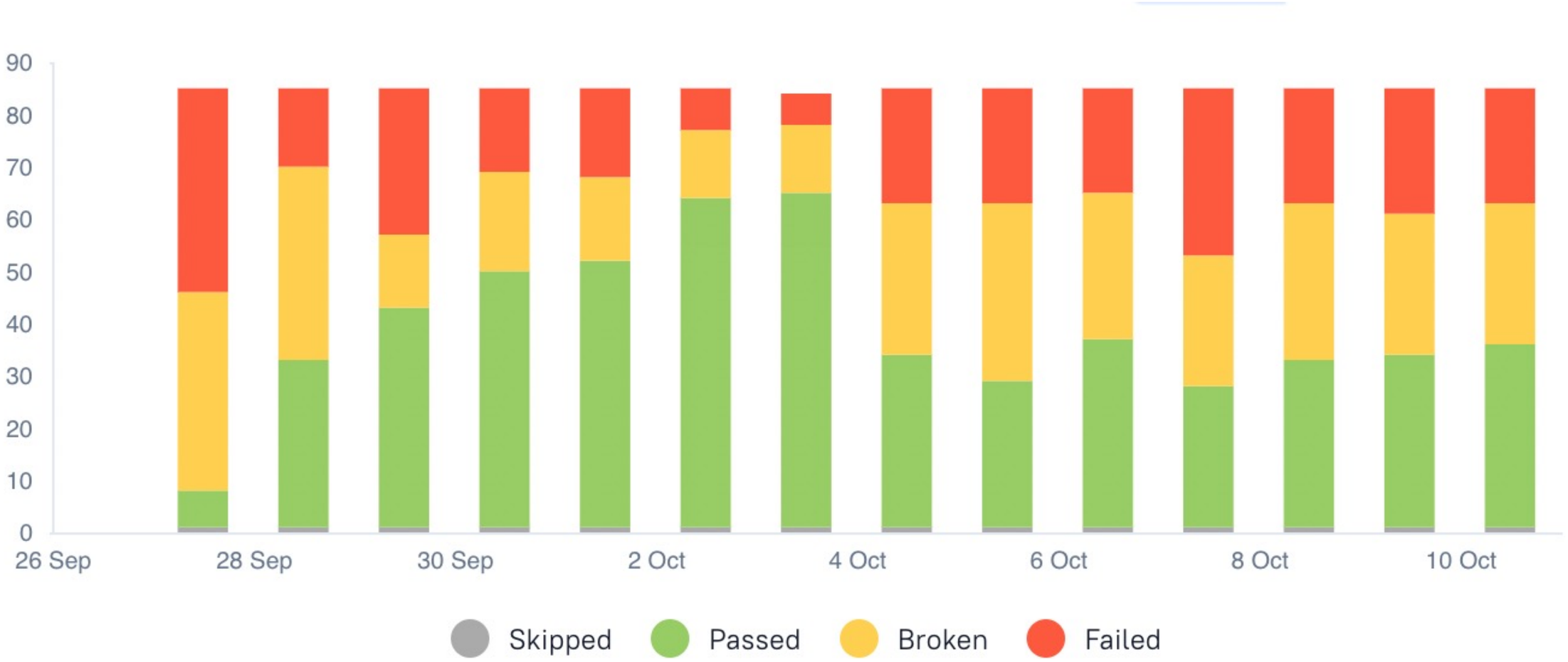
Жизнь за Нерзула

Как сгенерировать 1000 моков и отказаться от тестовых стендов

Белоглазов Алексей


RGX

Allure


90
80
70
60
50
40
30
20
10
0
26 Sep
28 Sep
30 Sep
2 Oct
4 Oct
6 Oct
8 Oct
10 Oct
Skipped
Passed
Broken
Failed

Спикер расскажет про «Нерзул» — инструмент,
который генерирует моки
на основе поведения реальных сервисов.
Это один из инструментов в подходе shift left,
где нужно тестировать требования к бизнес-задачам.

Предисловие

- Quality Assurance
- Agile Testing
- Dev Ops

Предисловие

- Quality Assurance
- Agile Testing
- Dev Ops
- Suffer

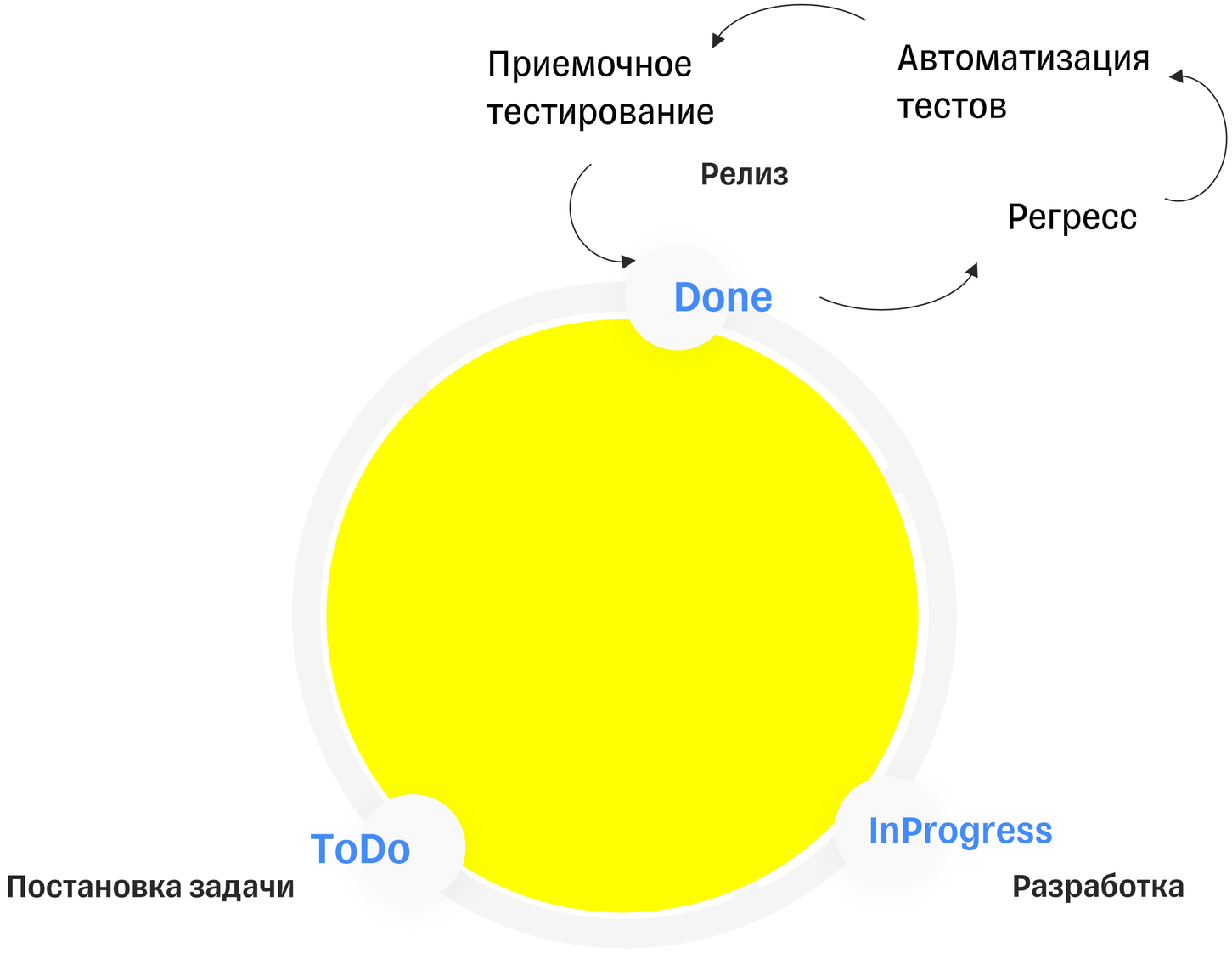
Цикл Жизни Задачи

Последовательность выполнения работ над задачей

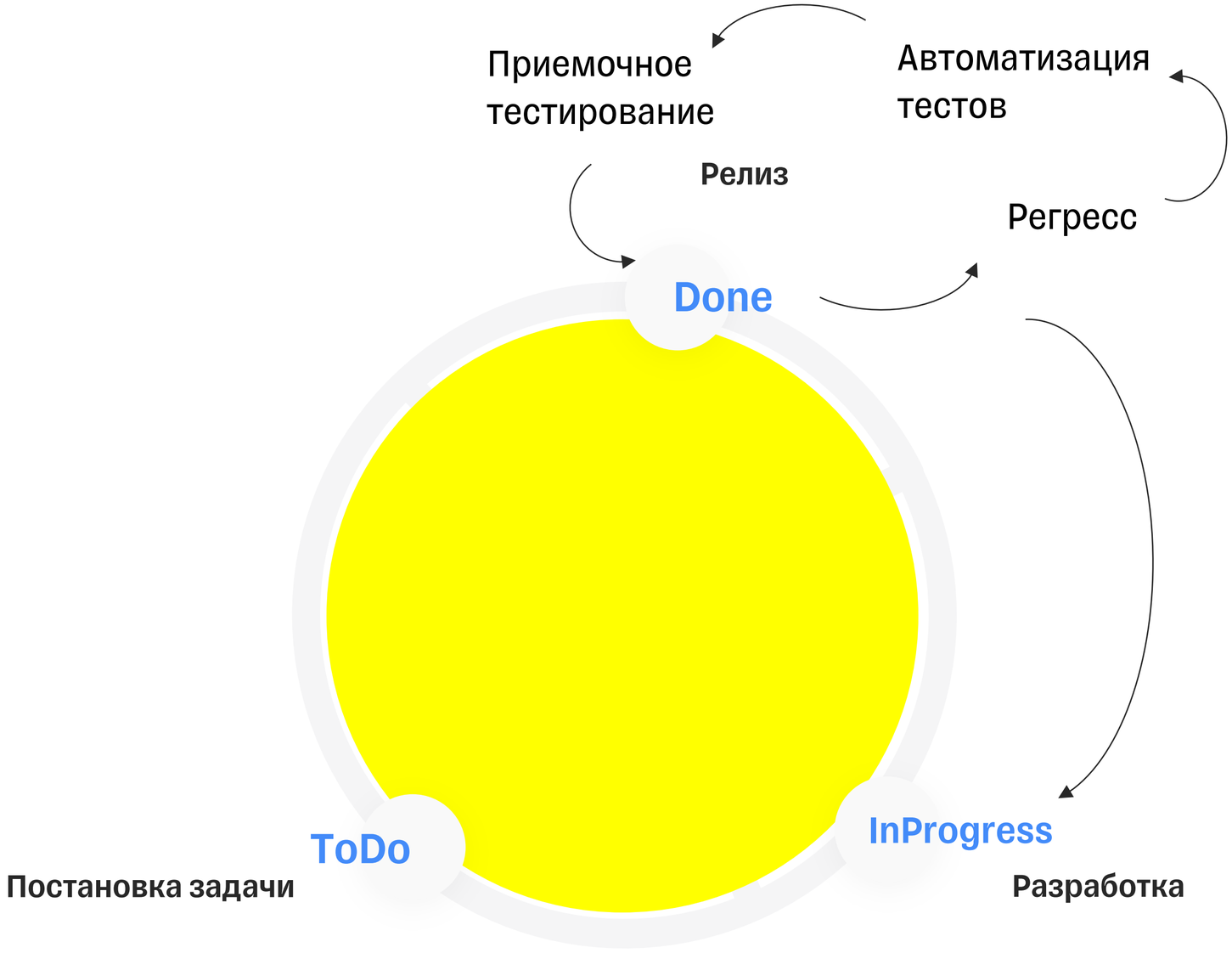
Постановка задачи | Разработка | Регресс | Автоматизация Тестов | Релиз | Возврат Багов

Последовательность выполнения работ над задачей

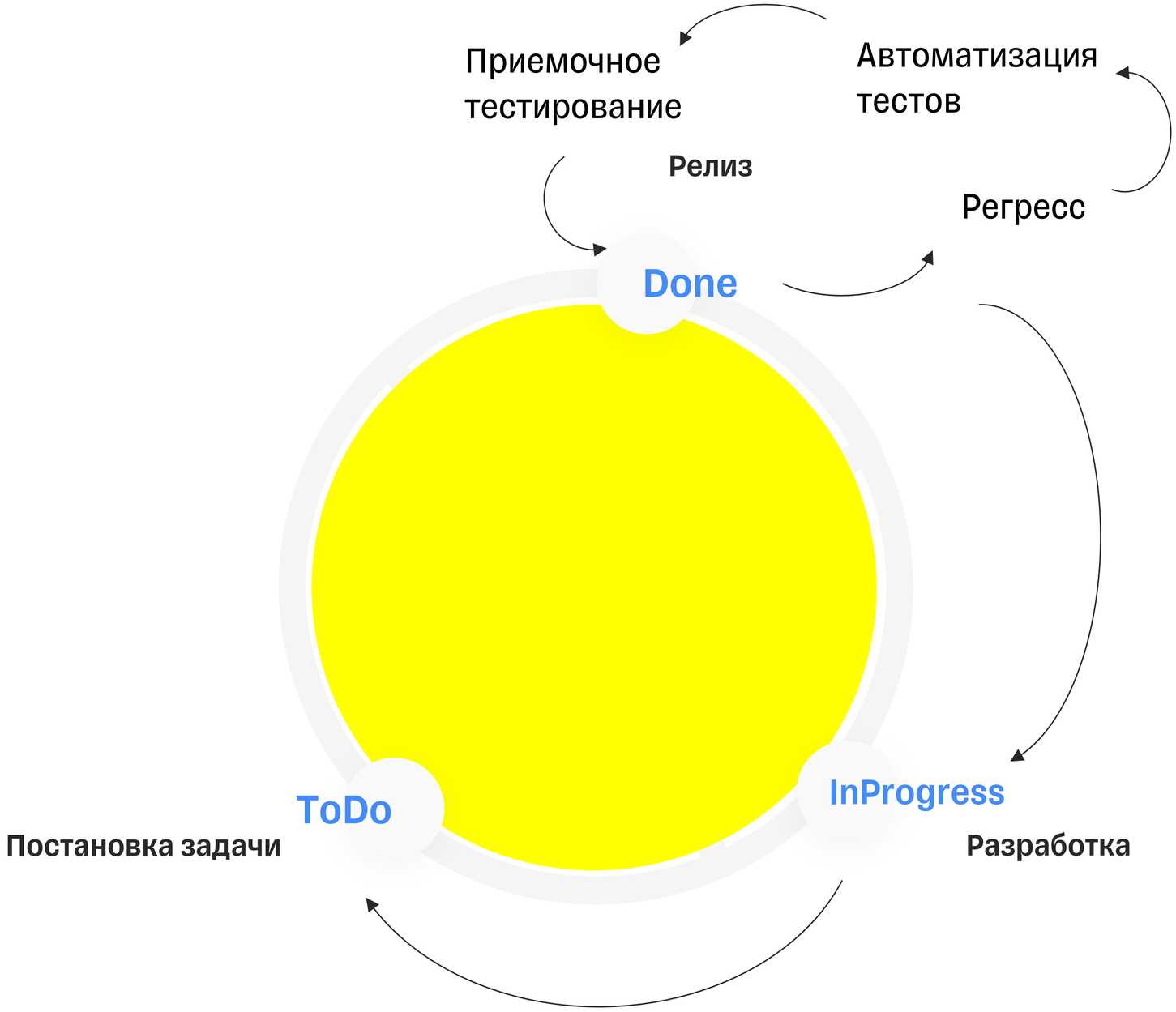
Цикл Жизни Задачи



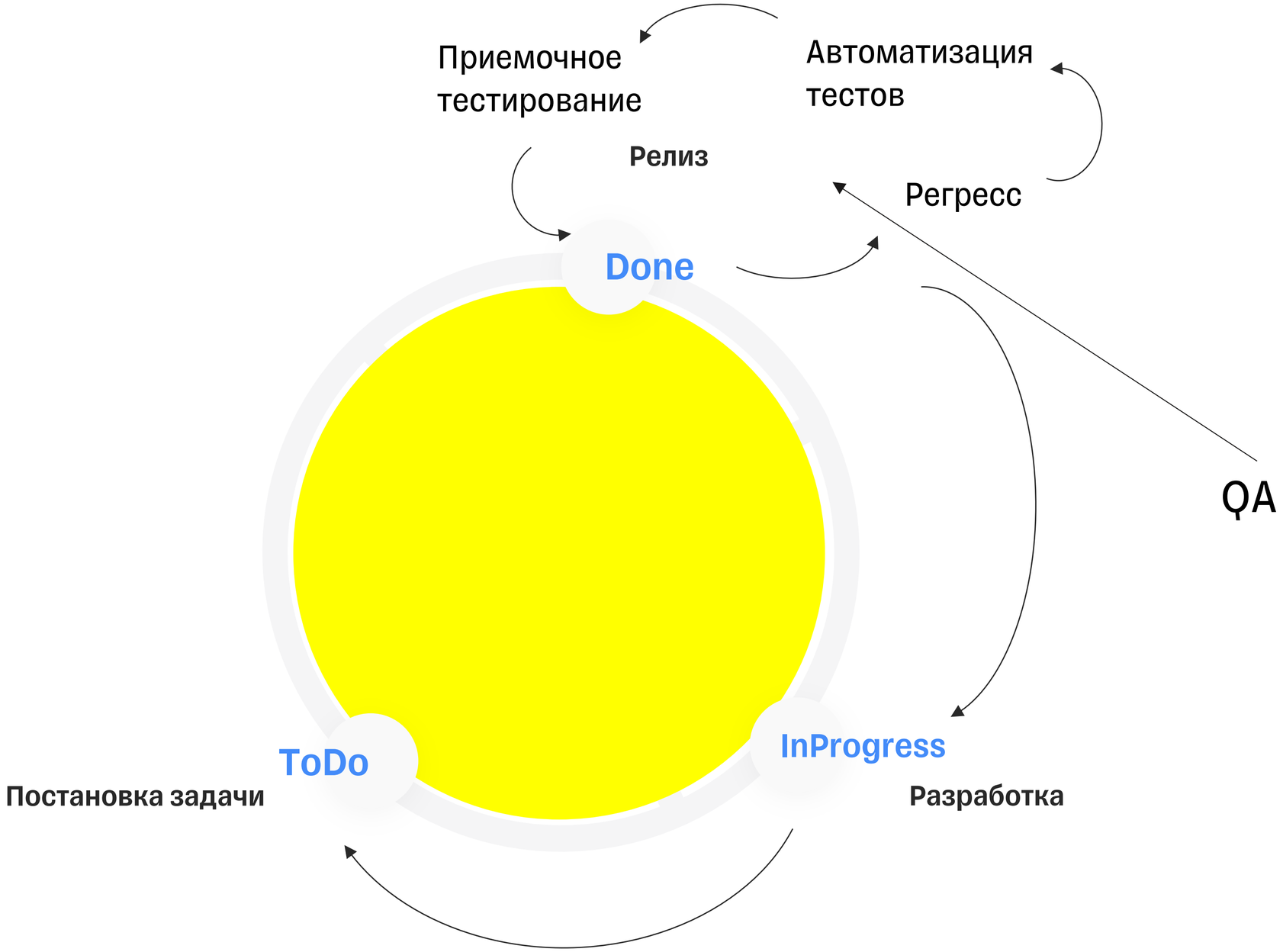
Цикл Жизни Задачи



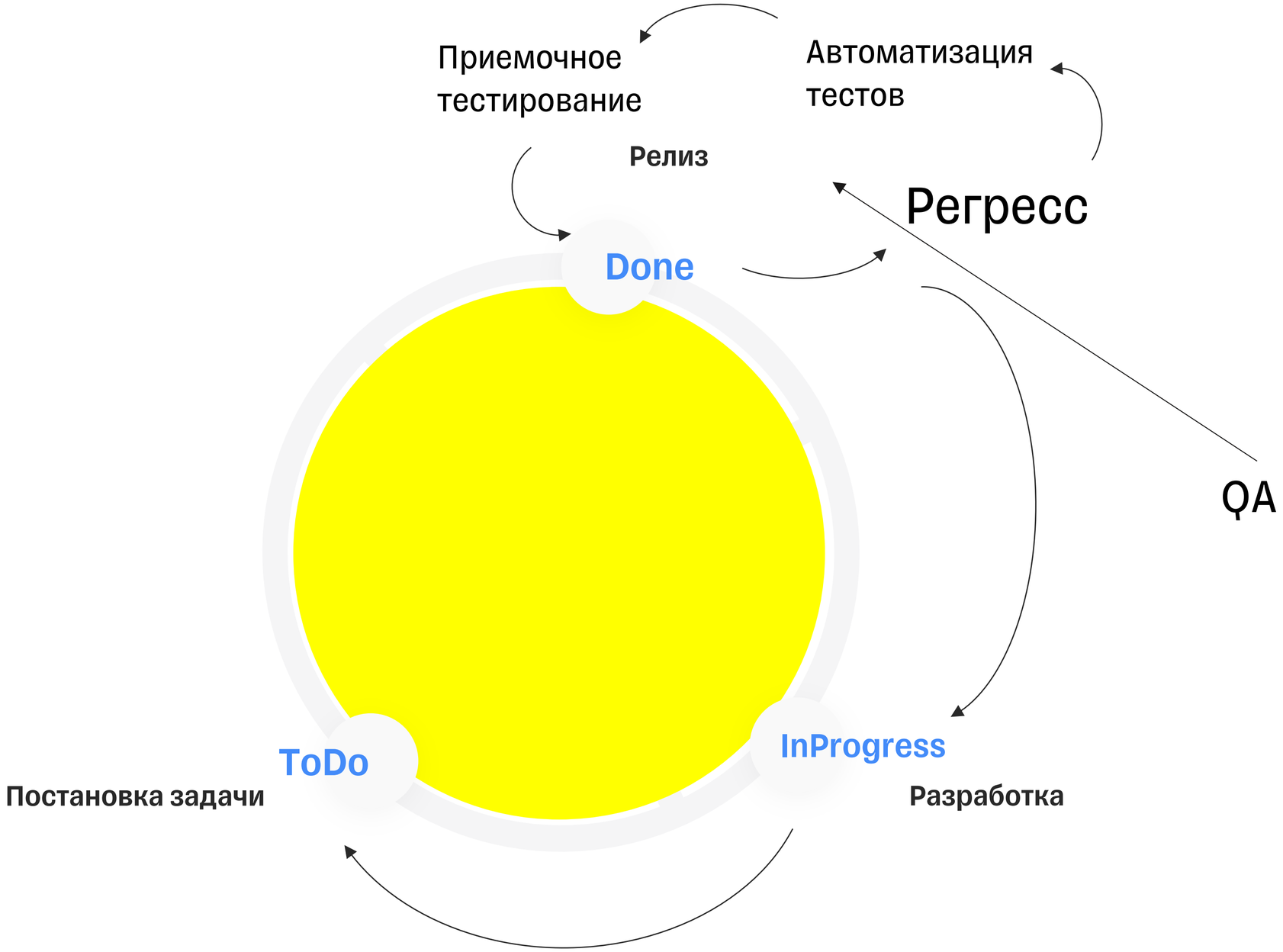
Цикл Жизни Задачи



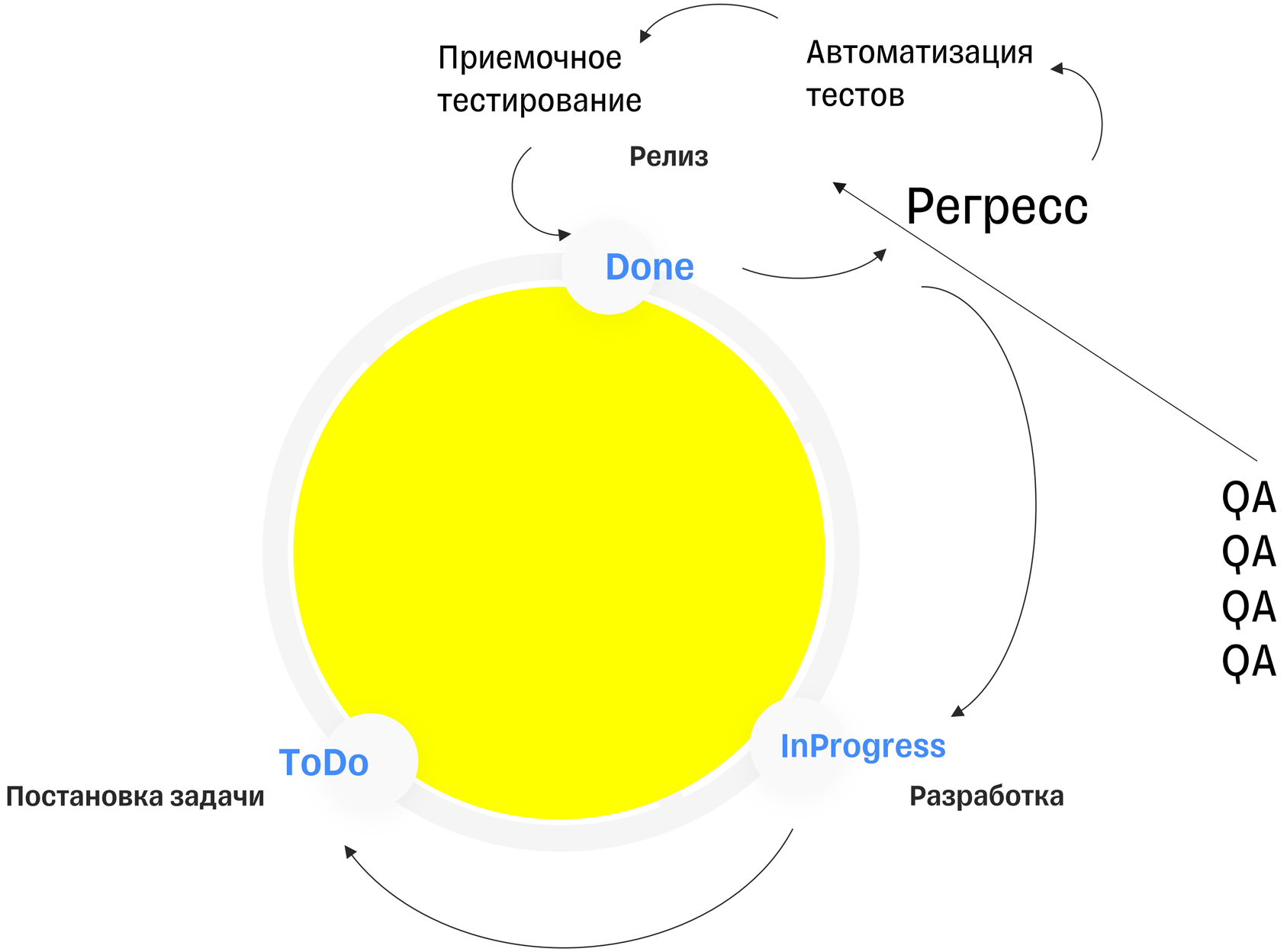
Цикл Жизни Задачи



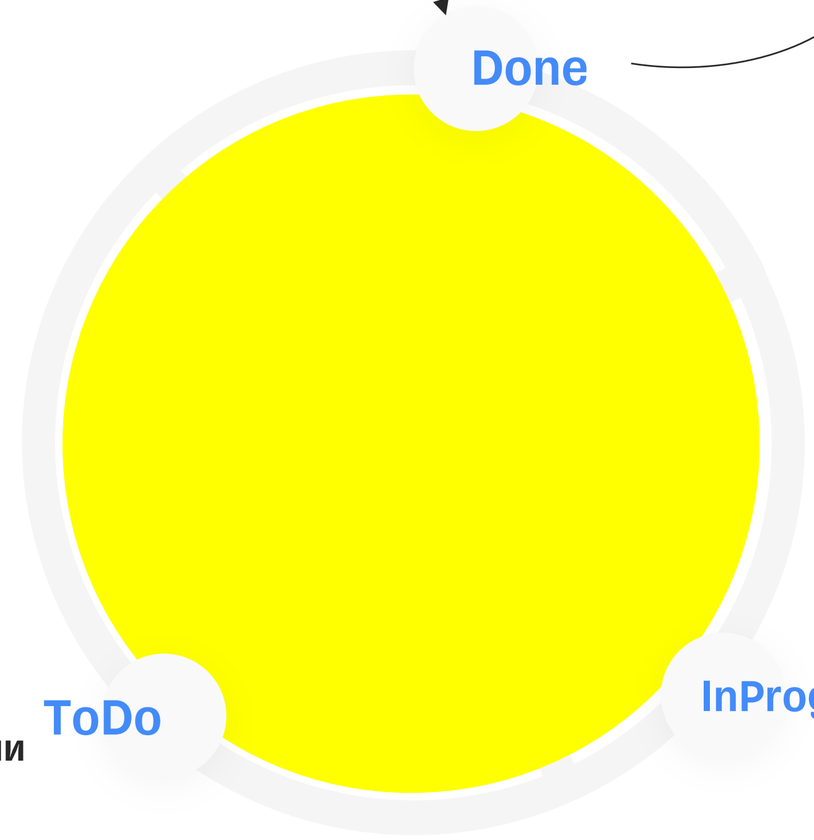
Цикл Жизни Задачи



Цикл Жизни Задачи



Цикл Жизни Задачи



Приемочное Тестирование

Релиз

Разработка

Автоматизация тестов

Постановка задачи

Тест спецификация

- Требования к постановке задач
- Три Амиго
- Пирамида тестирования
- Аллюр
- Интеграционные тесты без контуров
- Канареечное тестирование
- Метрики прода

- Требования к постановке задач
- Три Амиго
- Пирамида тестирования
- Аллюр
- Интеграционные тесты без контуров
- Канареечное тестирование
- Метрики прода

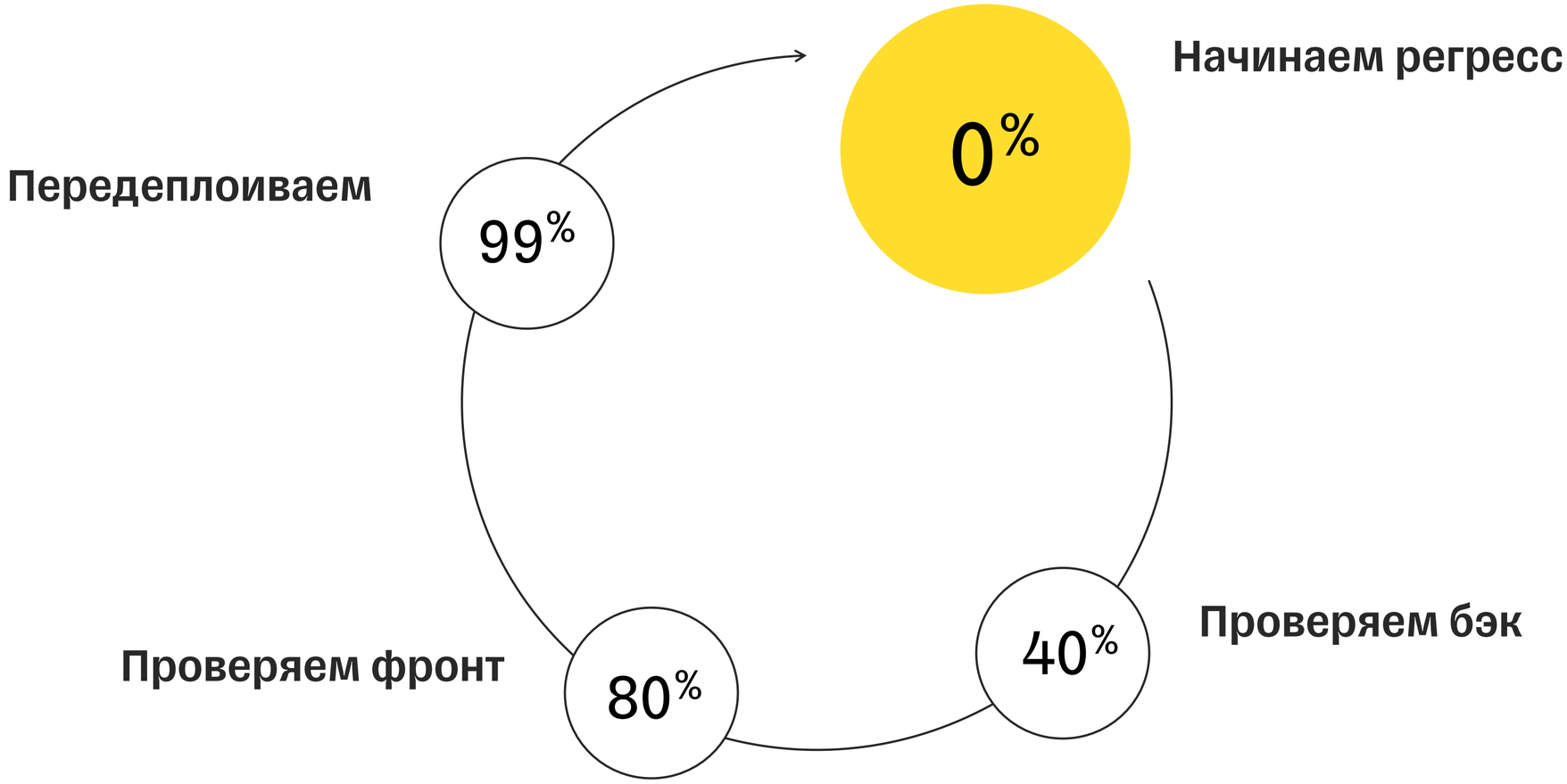
Проблематика

Тестовый контур

01

Поддержка компонентов

Поддержка компонентов



Тестовый контур

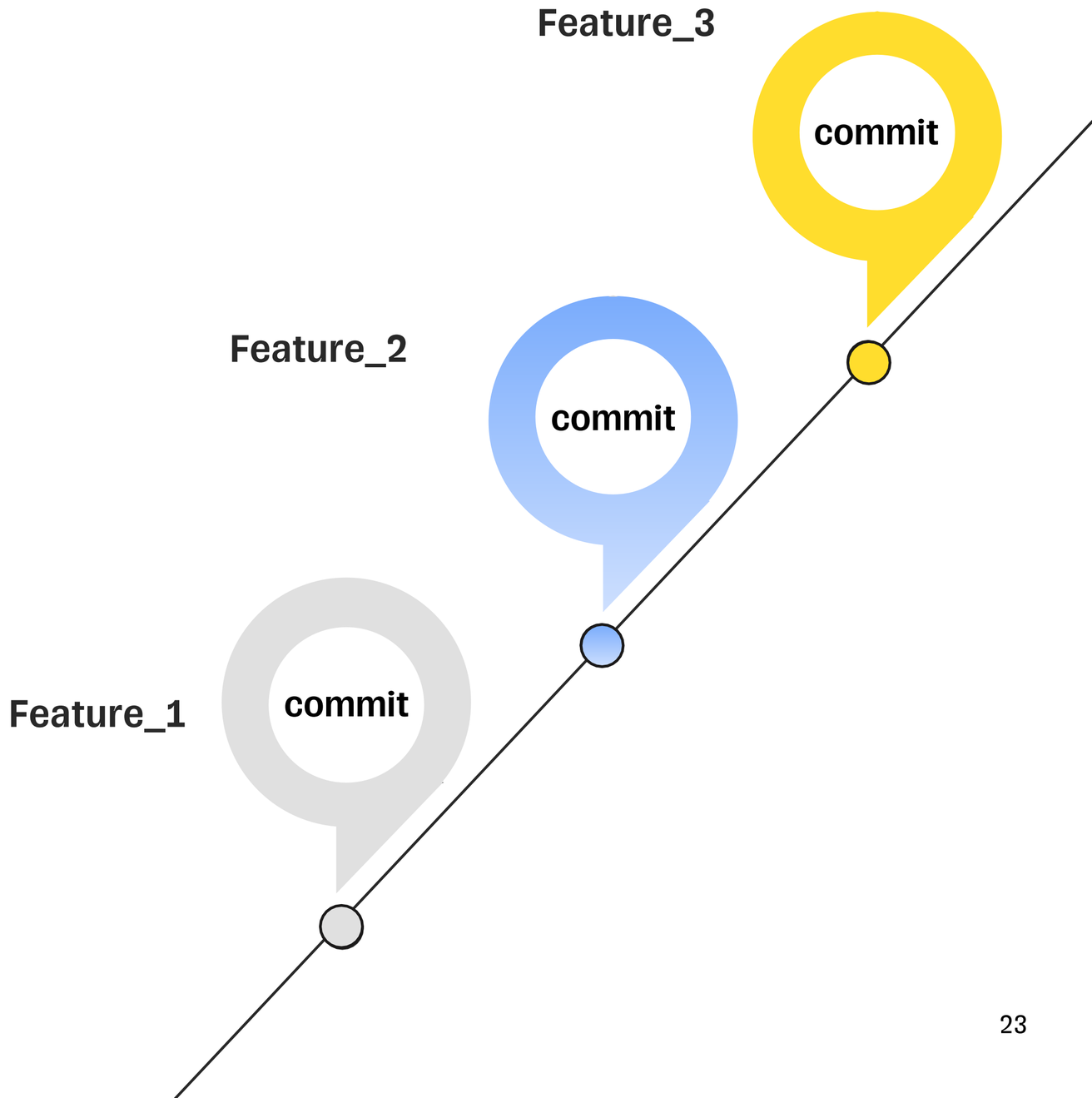
01

Поддержка компонентов

02

Тестирование разных веток встает в очередь

REGRESS



Тестовый контур

01

Поддержка компонентов

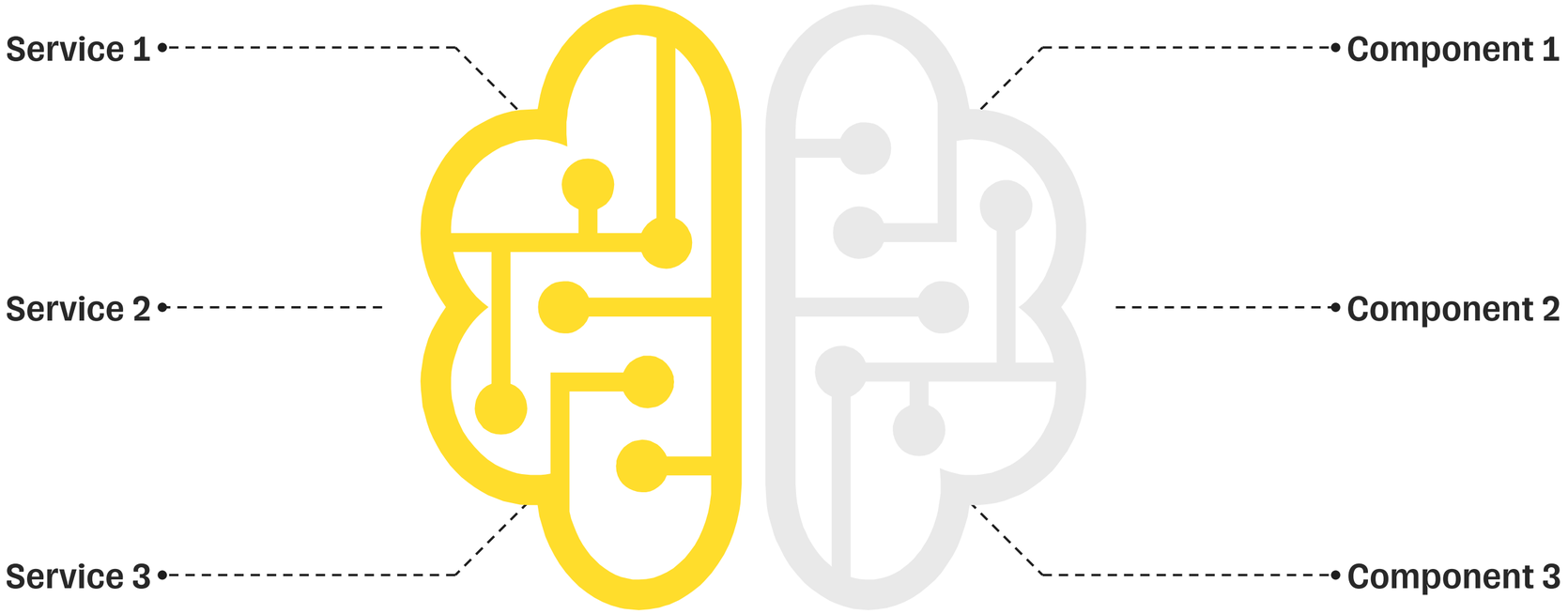
02

Тестирование разных веток
встает в очередь
можно решить несколькими
контурами

03

SLA разных микросервисов
умножаются
(0,99)^n

Тестовый контур



Тестовый контур

01

Поддержка компонентов

02

Тестирование разных веток
встает в очередь
можно решить несколькими
контурами

03

SLA разных микросервисов
умножаются
(0,99)^n

04

Отдельная команда на поддержку
тестового контура

Тестовый контур

01

Поддержка компонентов

02

Тестирование разных веток
встает в очередь
можно решить несколькими
контурами

03

SLA разных микросервисов
умножаются
(0,99)^n

04

Отдельная команда на поддержку
тестового контура

05

Большие ресурсы

Тестовый контур

01

Поддержка компонентов

02

Тестирование разных веток
встает в очередь
можно решить несколькими
контурами

03

SLA разных микросервисов
умножаются
(0,99)^n

04

Отдельная команда на поддержку
тестового контура

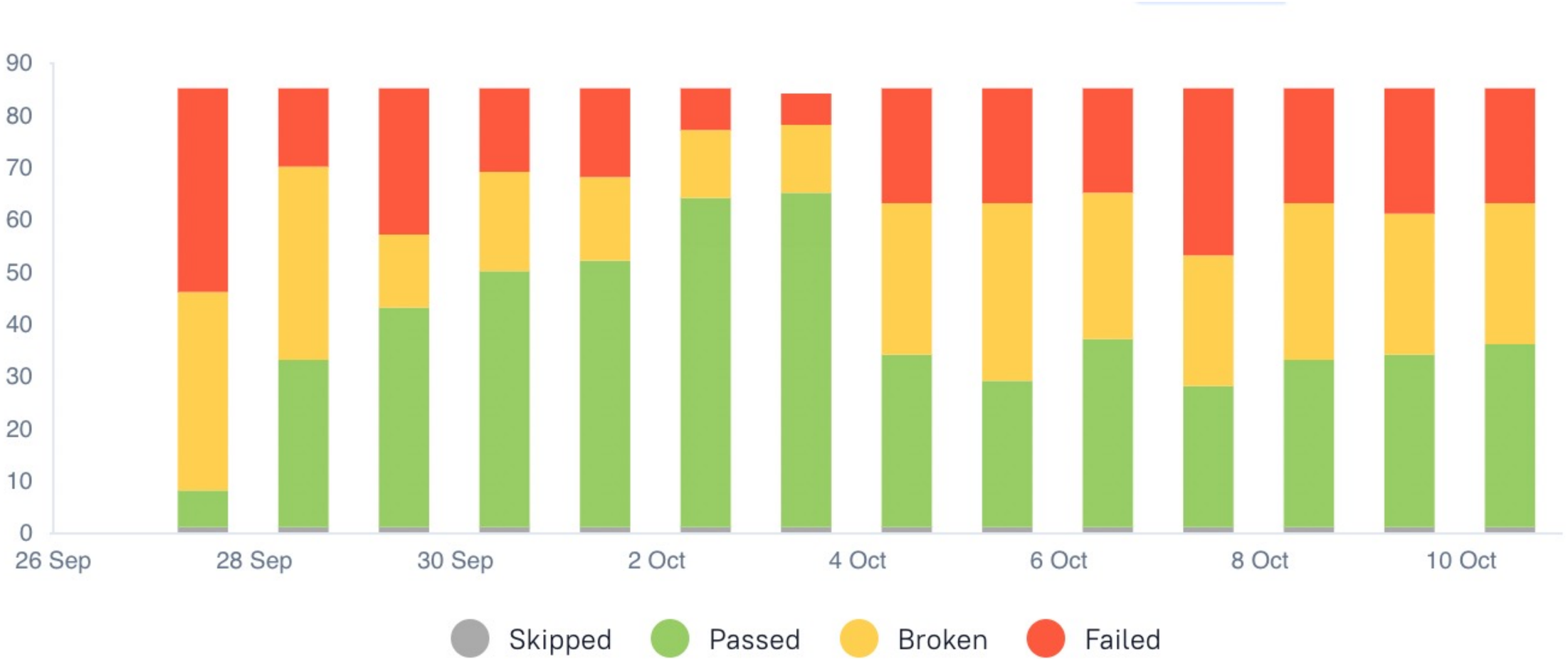
05

Большие ресурсы

06

Рандомные тестовые данные

Allure


90
80
70
60
50
40
30
20
10
0
26 Sep
28 Sep
30 Sep
2 Oct
4 Oct
6 Oct
8 Oct
10 Oct
Skipped
Passed
Broken
Failed

Тестовый контур

01

Поддержка компонентов

фичаконтур

Тестирование разных веток
встает в очередь
можно решить несколькими
контурами

03

SLA разных микросервисов
умножаются
(0,99)^n

04

Отдельная команда на поддержку
тестового контура

05

Большие ресурсы

06

Рандомные тестовые данные

Тестовый контур

docker-compose

Поддержка компонентов

фичаконтур

Тестирование разных веток
встает в очередь
можно решить несколькими
контурами

docker-compose

SLA разных микросервисов
умножаются
(0,99)^n

docker-compose

Отдельная команда на поддержку
тестового контура

05

Большие ресурсы

docker-compose

Рандомные тестовые данные

Тестовый контур

docker-compose

Поддержка компонентов

фичаконтур

Тестирование разных веток
встает в очередь
можно решить несколькими
контурами

docker-compose

SLA разных микросервисов
умножаются
(0,99)^n

docker-compose

Отдельная команда на поддержку
тестового контура

mocks

Большие ресурсы

docker-compose

Рандомные тестовые данные

Тестовый контур

docker-compose

Поддержка компонентов

фичаконтур

Тестирование разных веток
встает в очередь
можно решить несколькими
контурами

docker-compose

SLA разных микросервисов
умножаются
(0,99)^n

docker-compose

Отдельная команда на поддержку
тестового контура

mocks

Большие ресурсы

docker-compose

Рандомные тестовые данные

07

Соответствие моков

Популярные мокеры

Mockito

Типы библиотек для создания моков для зависимостей из классов своего приложения

Популярные мокеры

Mock server

Mockito

Типы библиотек для создания моков для зависимостей из классов своего приложения

Mock-service

Это инструмент для мокирования запросов от вашего сервиса, поведение моков из классов своего приложения

Популярные мокеры

Mockito

Типы библиотек для создания моков для зависимостей из классов своего приложения

Mock-service

Это инструмент для мокирования запросов от вашего сервиса, поведение моков из классов своего приложения

Mountebank

Инструмент для управления поведением моков, к которым обращается тестируемое приложение

Популярные мокеры

Mockito

Типы библиотек для создания моков для зависимостей из классов своего приложения

Mock server

Mock-service

Это инструмент для мокирования запросов от вашего сервиса, поведение моков из классов своего приложения

Mountebank

Инструмент для управления поведением моков, к которым обращается тестируемое приложение

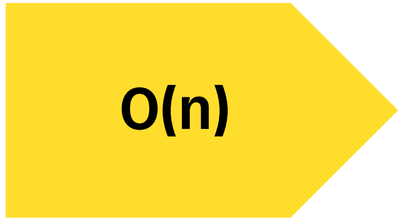
contracts

PACTS

Pact

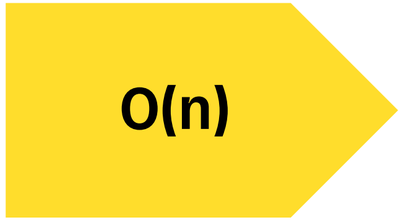
Позволяет проводить контрактное тестирование, обеспечивающее соответствие контрактов между вашими моками и реальными сервисами

Асимптотическая сложность ручного труда для написания теста

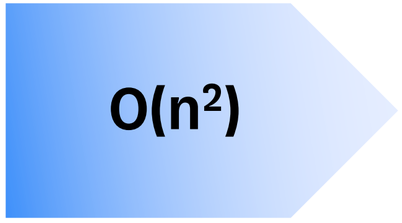


Изучить сваггеры сторонних приложений

Асимптотическая сложность ручного труда для написания теста

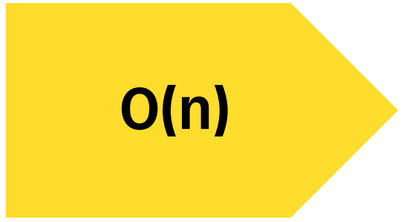


Изучить сваггеры сторонних приложений

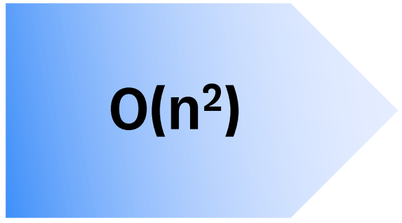


Написать мок, сверив его с поведением стороннего приложения

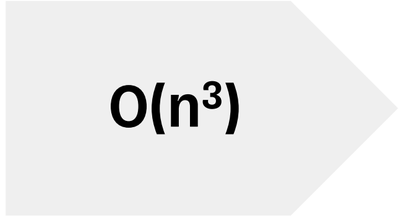
Асимптотическая сложность ручного труда для написания теста



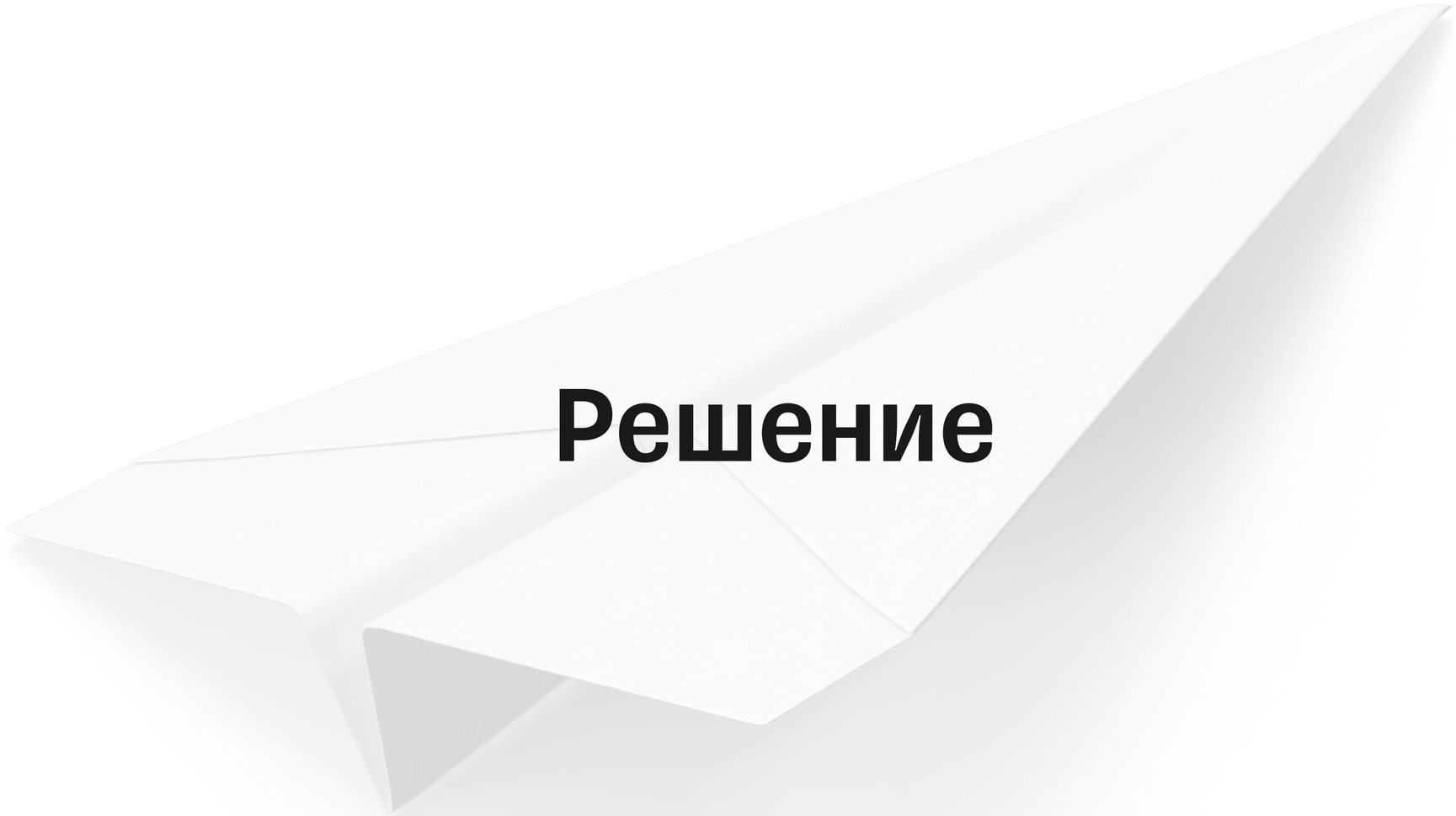
Изучить сваггеры сторонних приложений



Написать мок, сверив его с поведением стороннего приложения



Написать контрактный тест



Решение

Что такое API test?

```
Test () {
        method = “POST”
        baseUrl = “http://som.site.ru”
        endpoint = “/user”
        httpClient = new HttpClient(baseUrl)
        model = new Model.Builder().Build()
        request = httpClient.method(method, endpoint).body(model)
        response = httpClient.asJson()
        assertEquals(response.status, 200)
}
```

Что такое API test?

```
Test () {
        method = “POST”
        baseUrl = “http://som.site.ru”
        endpoint = “/user”
        httpClient = new HttpClient(baseUrl)
        model = new Model.Builder().Build()
        request = httpClient.method(method, endpoint).body(model)
        response = httpClient.asJson()
        assertEquals(response.status, 200)
}
```

Что такое API test?

```
Test () {
        method = “POST”
        baseUrl = “http://som.site.ru”
        endpoint = “/user”
        httpClient = new HttpClient(baseUrl)
        model = new Model.Builder().Build()
        request = httpClient.method(method, endpoint).body(model)
        response = httpClient.asJson()
        assertEquals(response.status, 200)
        export()
}
```

Итоговая json заглушка

```
{
        request: {
                url: “http://some.site.ru/user”,
                method : “POST”,
                body: {}
        }
        response: {
                status: 200,
                body: { message: “OK” }
        }
}
```

Итоговая json заглушка

```
{
        request: {
                url: “http://some.site.ru/user”,
                method : “POST”,
                body: {}
        }
        response: {
                status: 200,
                body: { message: “OK” }
        }
}
```

API test

```
Test () {
        method = “POST”
        baseUrl = “http://som.site.ru”
        endpoint = “/user”
        httpClient = new HttpClient(baseUrl)
        model = new Model.Builder().Build()
        request = httpClient.method(method, endpoint).body(model)
        response = httpClient.asJson()
        assertEquals(response.status, 200)
}
```

Итоговая json заглушка

```
{
        request: {
                url: “http://some.site.ru/user”,
                method : “POST”,
                body: {}
        }
        response: {
                status: 200,
                body: { message: “OK” }
        }
}
```

API test

```
Test () {
        method = “POST”
        baseUrl = “http://som.site.ru”
        endpoint = “/user”
        httpClient = new HttpClient(baseUrl)
        model = new Model.Builder().Build()
        request = httpClient.method(method, endpoint).body(model)
        response = httpClient.asJson()
        assertEquals(response.status, 200)
}
```

Итоговая json заглушка

```
{
        request: {
                url: “http://some.site.ru/user”,
                method : “POST”,
                body: {}
        }
        response: {
                status: 200,
                body: { message: “OK” }
        }
}
```

Итоговая json заглушка

```
{
        request: {
                url: “http://some.site.ru/user”,
                method : “POST”,
                body: {}
        }
        response: {
                status: 200,
                body: { message: “OK” }
        }
}
```

Реализация

Nerzul

ApiTest framework

Абстракция для апи тестов

Надстройка на refit

HTTP-Server

Слушает все запросы

На aspNet

Проведение тестирования микросервиса провайдера

Docker-compose

Среда с единой сетью

Доступностью компонентов 0,(9)

app

Гостевой контейнер, к которому идут запросы от тестов

dependencies

Гостевые контейнеры с Postgres, Redis, Kafka

К которым обращается app

tests

Подготавливают данные перед тестами

Обращаются к app во время тестов

Автоматически экспортируют моки

Проведение тестирования микросервиса провайдера

Docker-compose

Среда с единой сетью

Доступностью компонентов 0,(9)

app

Гостевой контейнер, к которому идут запросы от тестов

dependencies

Гостевые контейнеры с Postgres, Redis, Kafka

К которым обращается app

tests

Подготавливают данные перед тестами

Обращаются к app во время тестов

Автоматически экспортируют моки

Проведение тестирования микросервиса провайдера

Docker-compose

Среда с единой сетью

Доступностью компонентов 0,(9)

app

Гостевой контейнер, к которому идут запросы от тестов

dependencies

Гостевые контейнеры с Postgres, Redis, Kafka

К которым обращается app

tests

Подготавливают данные перед тестами

Обращаются к app во время тестов

Автоматически экспортируют моки

Проведение тестирования микросервиса провайдера

Docker-compose

Среда с единой сетью

Доступностью компонентов 0,(9)

app

Гостевой контейнер, к которому идут запросы от тестов

dependencies

Гостевые контейнеры с Postgres, Redis, Kafka

К которым обращается app

tests

Подготавливают данные перед тестами

Обращаются к app во время тестов

Автоматически экспортируют моки

Проведение тестирования микросервиса провайдера

Docker-compose

Среда с единой сетью

Доступностью компонентов 0,(9)

app

Гостевой контейнер, к которому идут запросы от тестов

dependencies

Гостевые контейнеры с Postgres, Redis, Kafka

К которым обращается app

tests

Подготавливают данные перед тестами

Обращаются к app во время тестов

Автоматически экспортируют моки

Итоговая json заглушка

```
{
        request: {
                url: “http://some.site.ru/user”,
                method : “POST”,
                body: {}
        }
        response: {
                status: 200,
                body: { message: “OK” }
        }
}
```

Проведение интеграционного тестирования сервиса клиента

Docker-compose

Среда с единой сетью

Доступностью компонентов 0,(9)

app

Гостевой контейнер, к которому идут запросы от тестов

dependencies

Гостевые контейнеры с Postgres, Redis, Kafka,

Фейки протестированных приложений

К которым обращается app

tests

Подготавливают данные перед тестами

Обращаются к app во время тестов

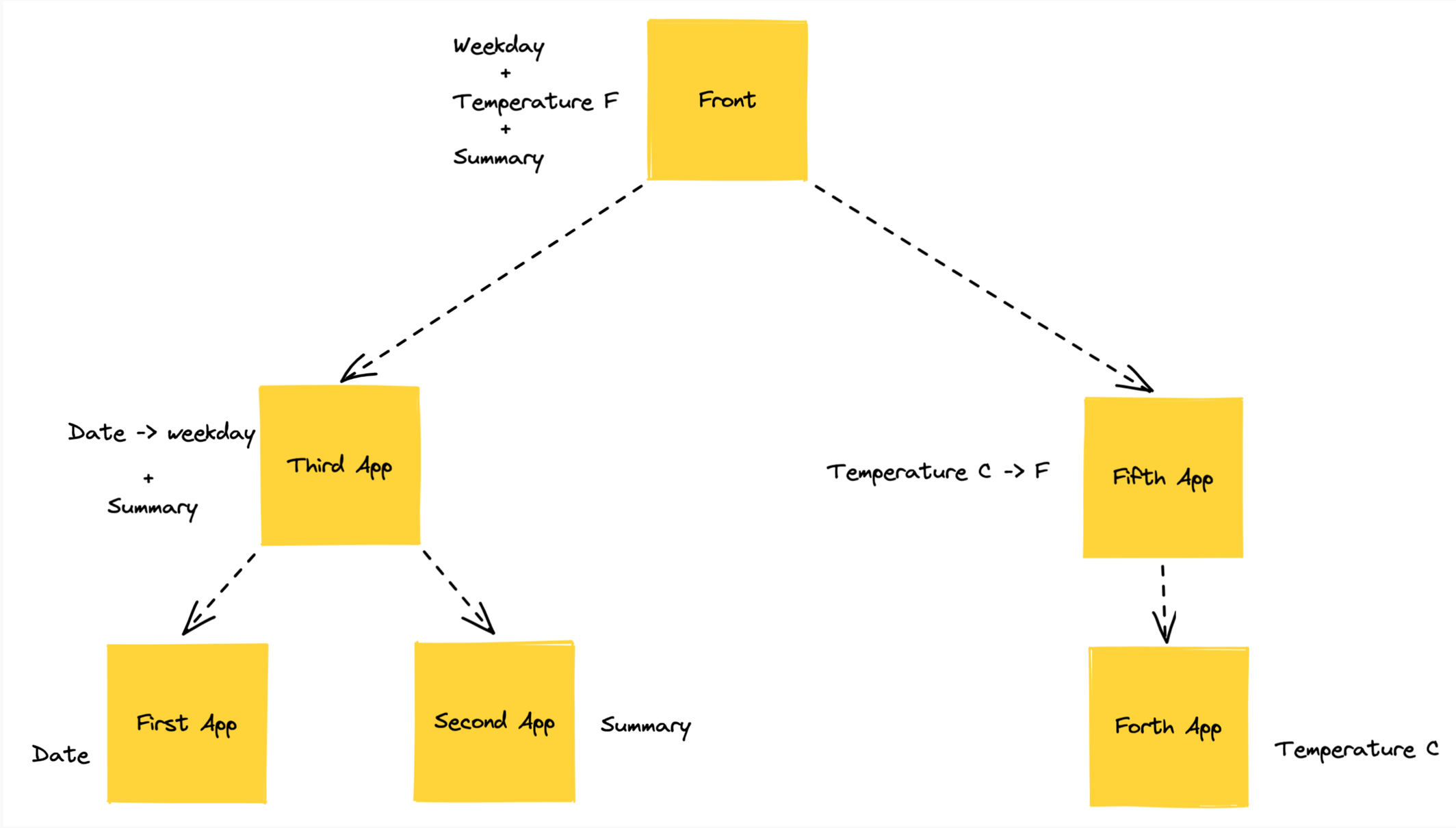
Автоматически экспортируют моки

Итоговая json заглушка

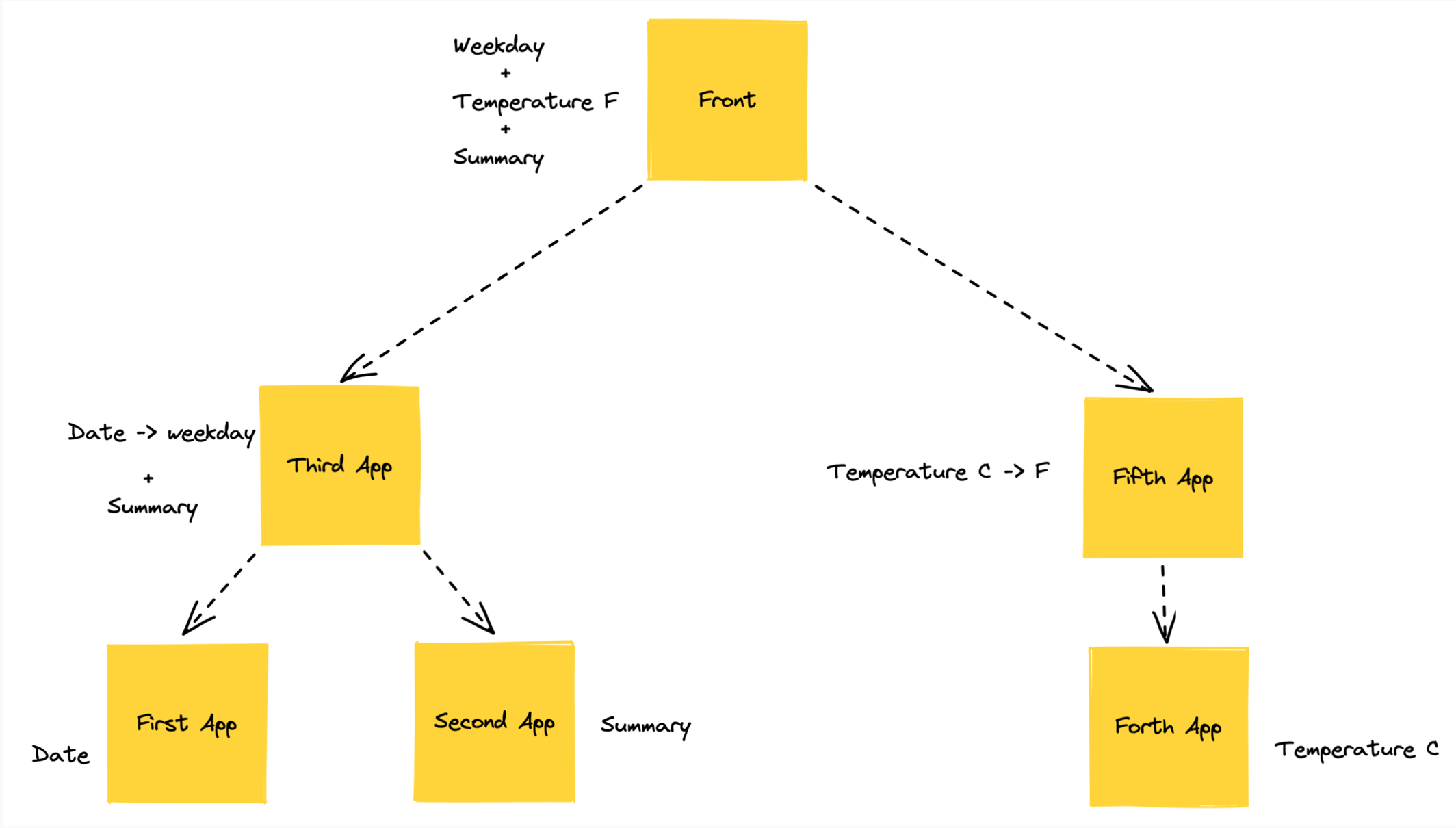
```
{
        request: {
                url: “http://some.site.ru/user”,
                method : “POST”,
                body: {}
        }
        response: {
                status: 200,
                body: { message: “OK” }
        }
}
```

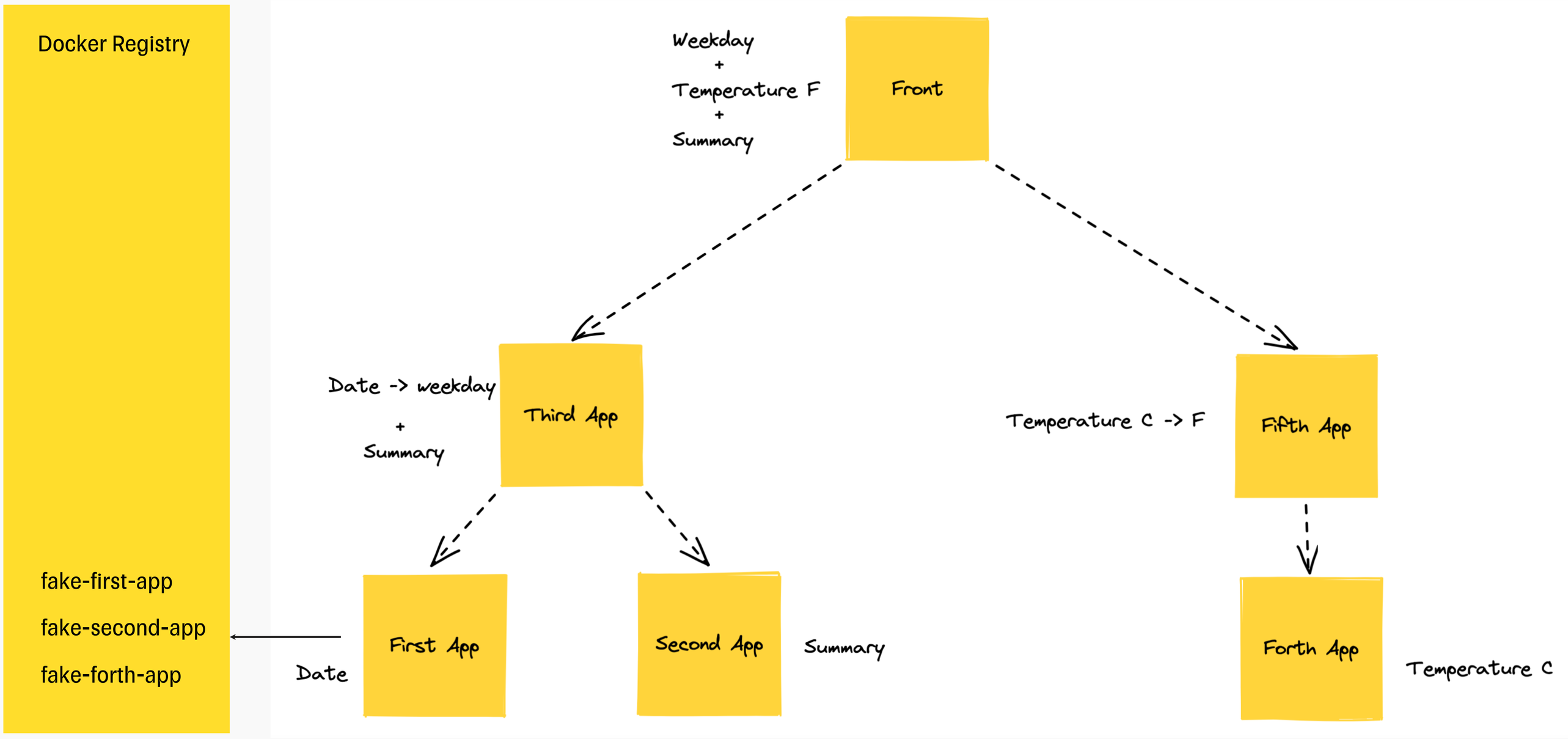
Итоговая json заглушка

```
{
        request: {
                url: “http://some.site.ru/user”,
                method : “POST”,
                body: {}
        }
        response: {
                status: 200,
                body: { message: “OK” }
        }
}
```


Weekday
+
Temperature F
+
Summary
Front
Date -> weekday
+
Summary
Third App
Temperature C -> F
Fifth App
First App
Second App
Summary
Forth App
Temperature C
Date

Docker Registry



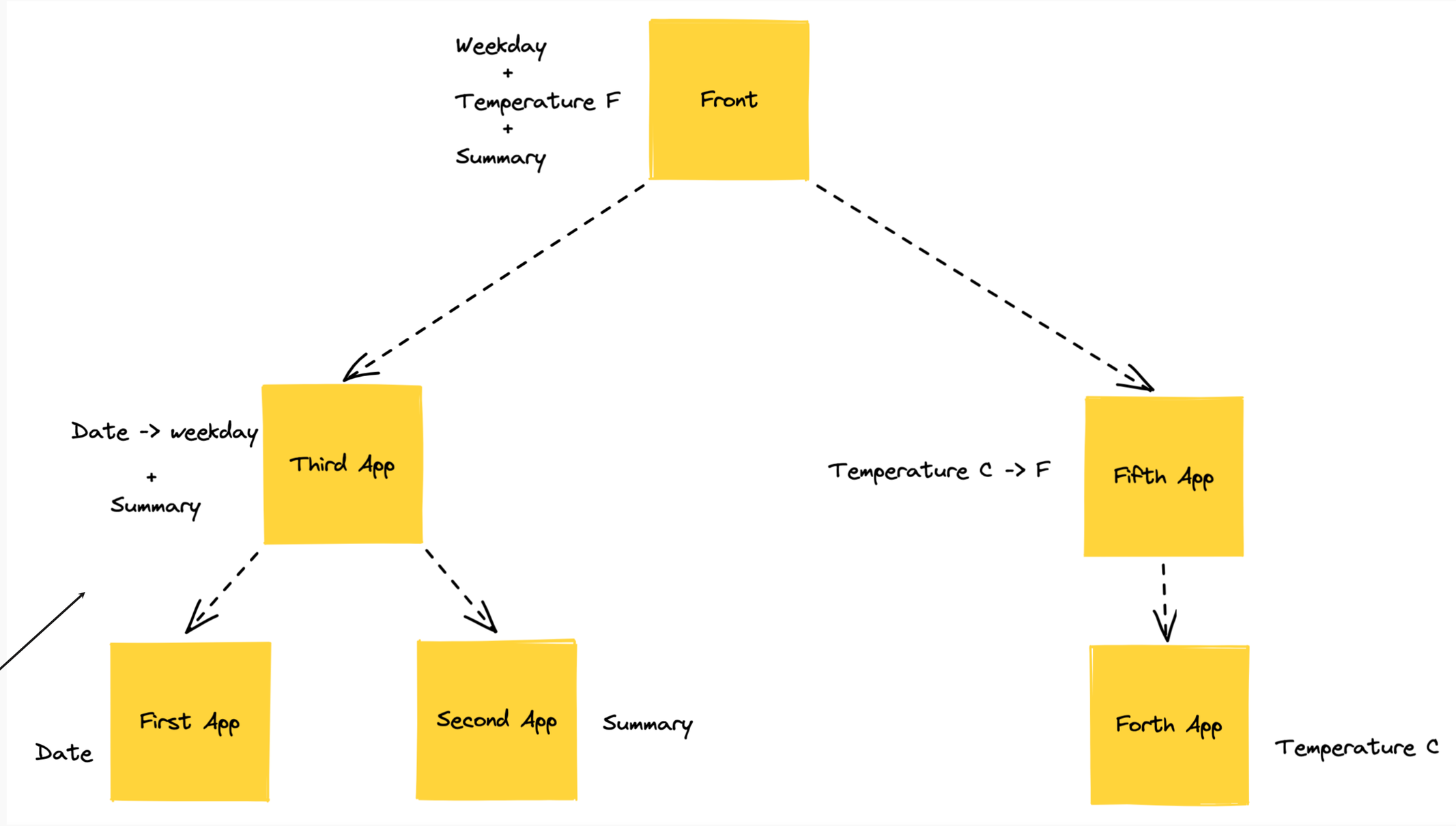

Docker Registry
fake-first-app
fake-second-app
fake-forth-app
Weekday
+
Temperature F
+
Summary
Front
Date -> weekday
+
Summary
Third App
Temperature C -> F
Fifth App
Date
First App
Second App
Summary
Forth App
Temperature C

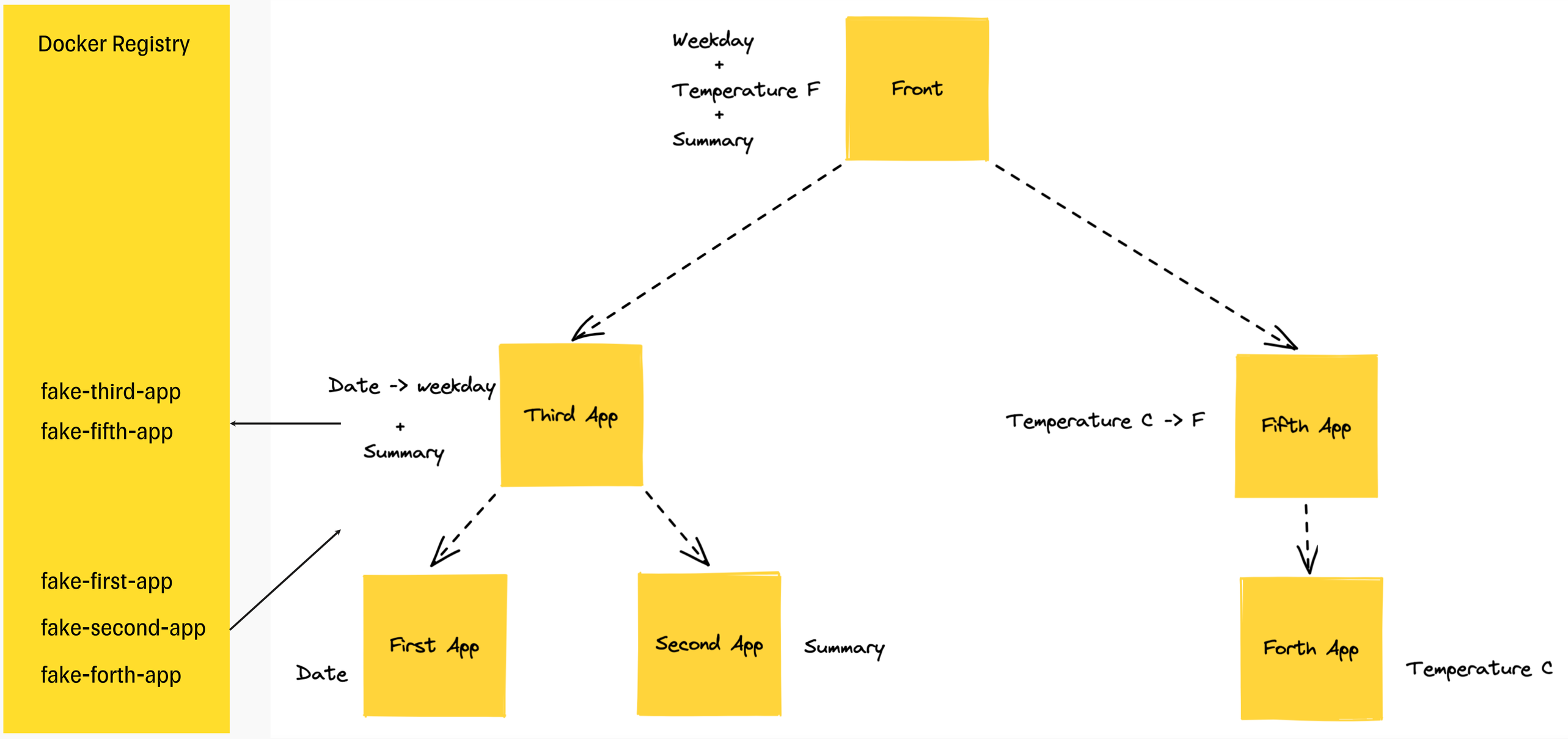
Docker Registry

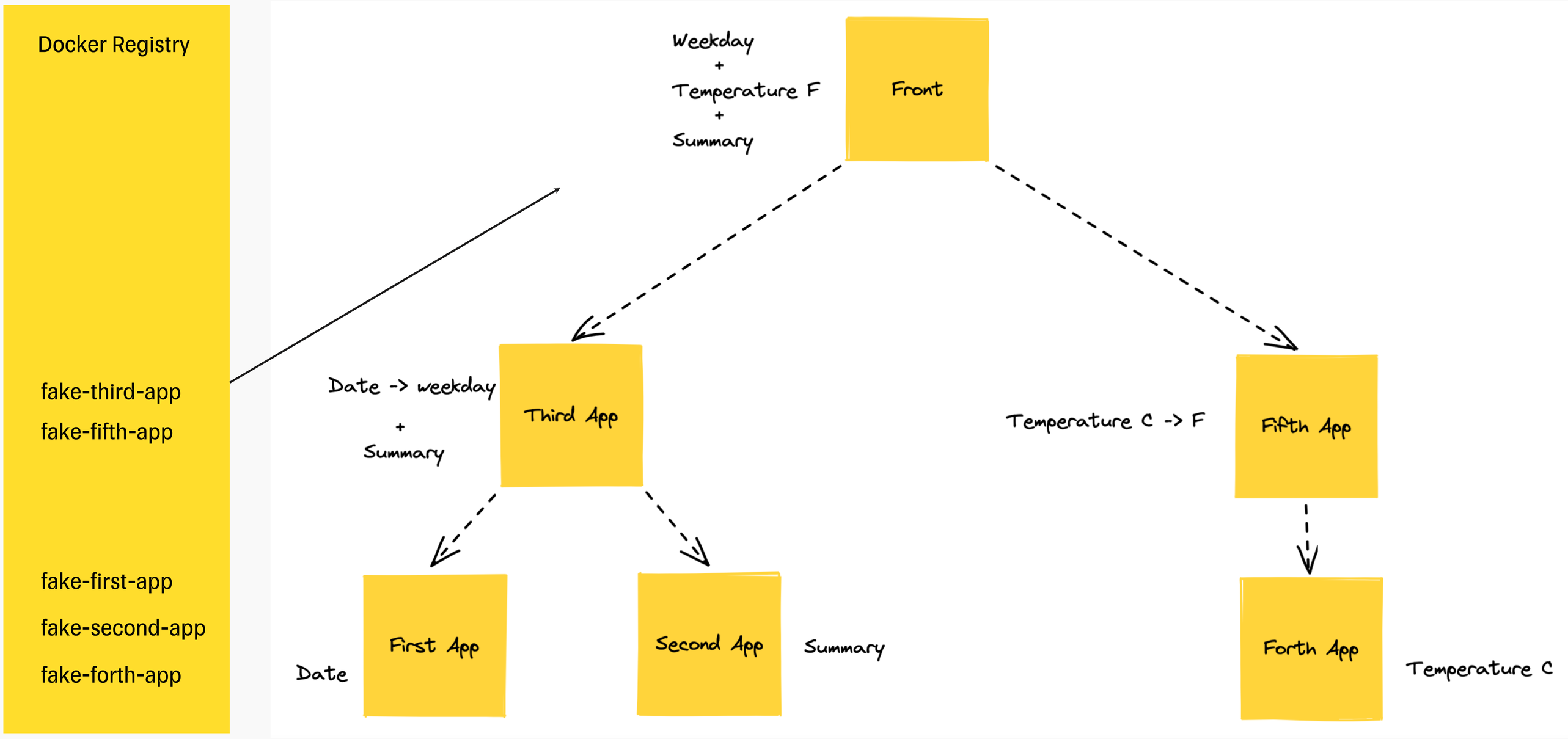
fake-first-app

fake-second-app

fake-forth-app




Docker Registry
fake-third-app
fake-fifth-app
fake-first-app
fake-second-app
fake-forth-app
Weekday
+
Temperature F
+
Summary
Front
Date -> weekday
+
Summary
Third App
Temperature C -> F
Fifth App
Date
First App
Second App
Summary
Forth App
Temperature C

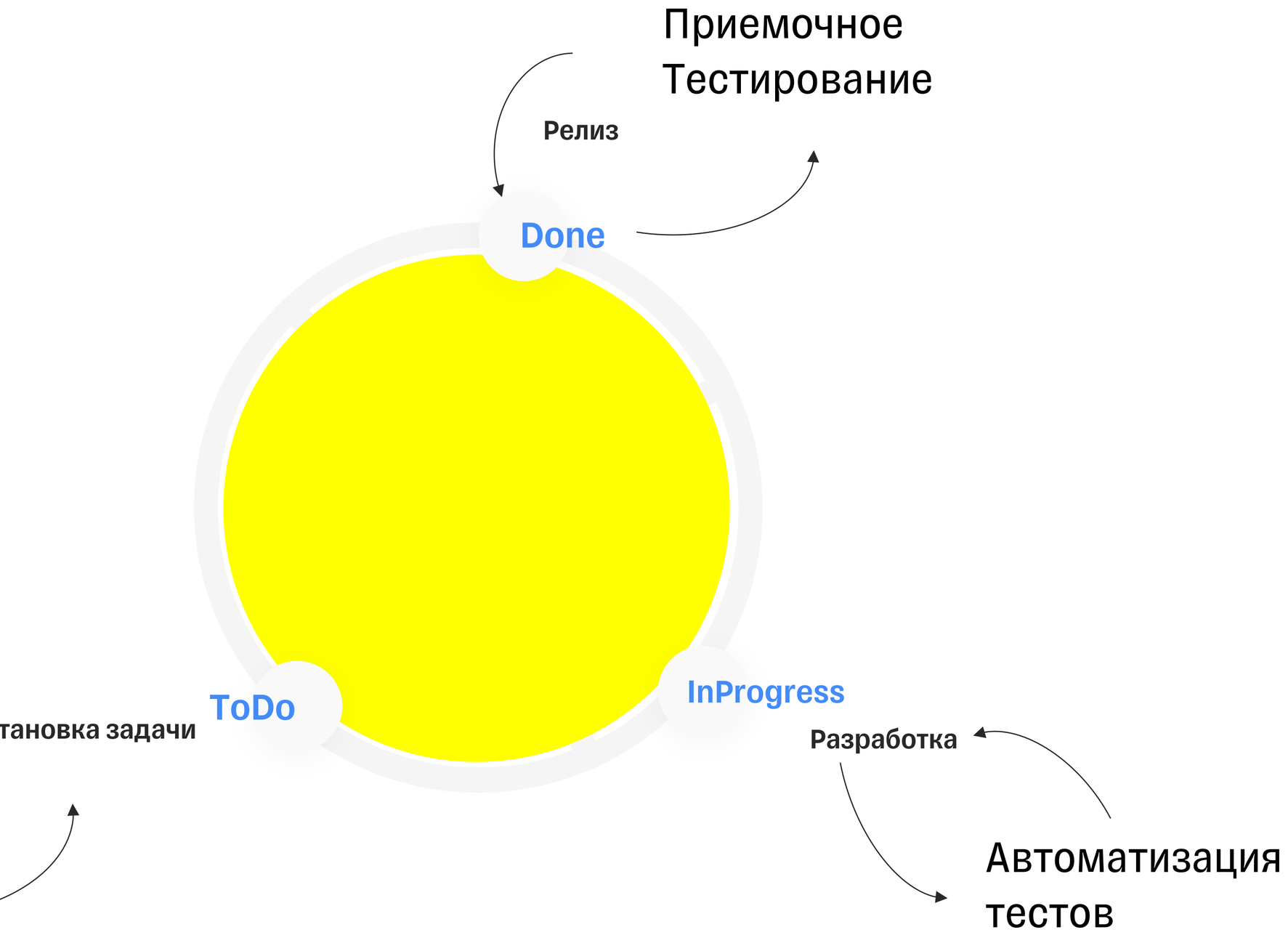

Docker Registry
fake-third-app
fake-fifth-app
fake-first-app
fake-second-app
fake-forth-app
Weekday
+
Temperature F
+
Summary
Front
Date -> weekday
+
Summary
Third App
Temperature C -> F
Fifth App
Date
First App
Second App
Summary
Forth App
Temperature C

Демонстрация

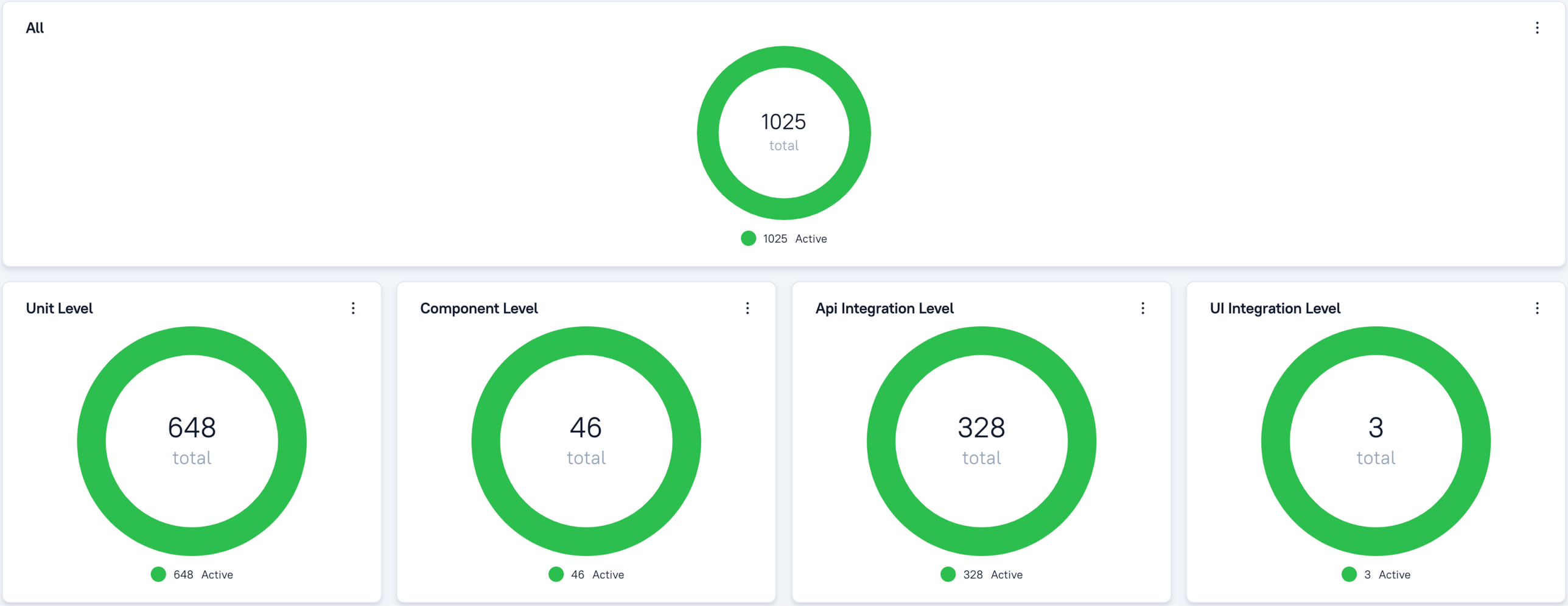
Итоги

Спикер расскажет про «Нерзул» — инструмент,
который генерирует моки
на основе поведения реальных сервисов.
Это один из инструментов в подходе shift left,
где нужно тестировать требования к бизнес-задачам.

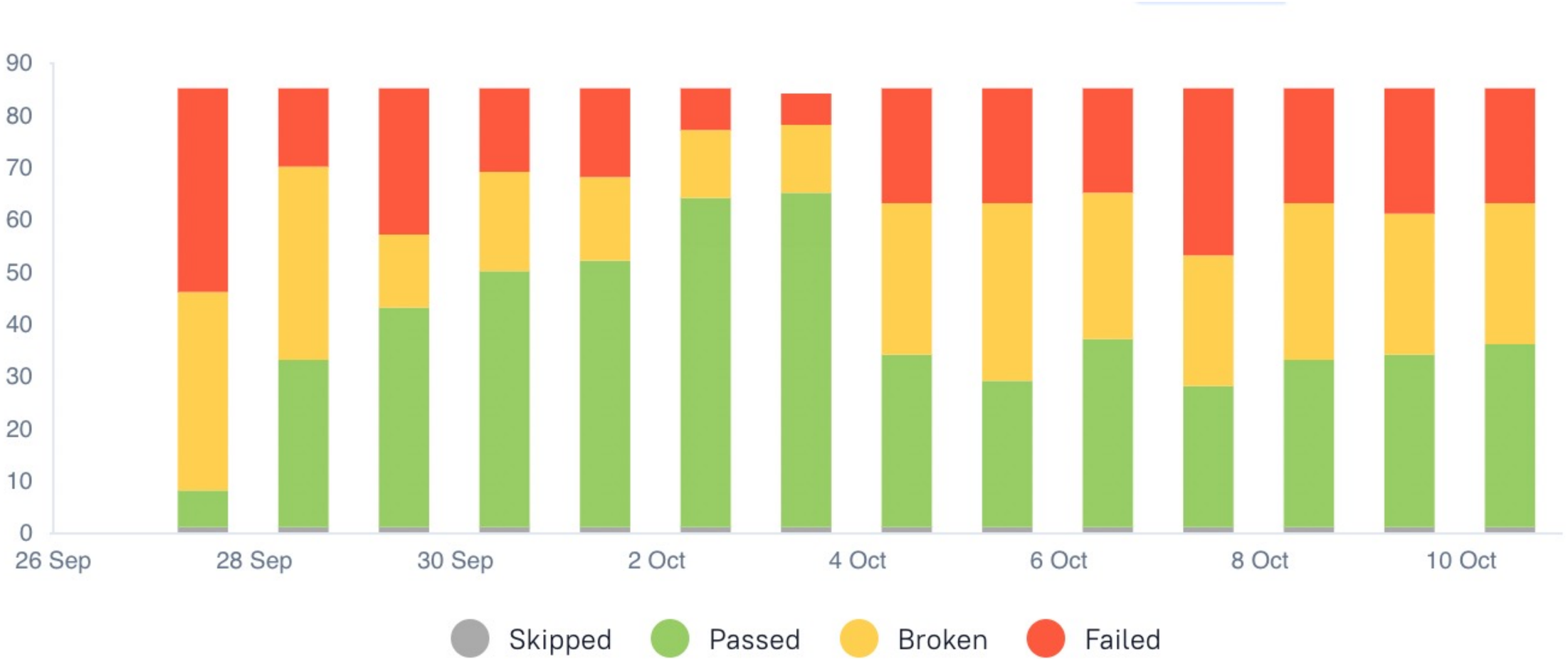
Цикл Жизни Задачи



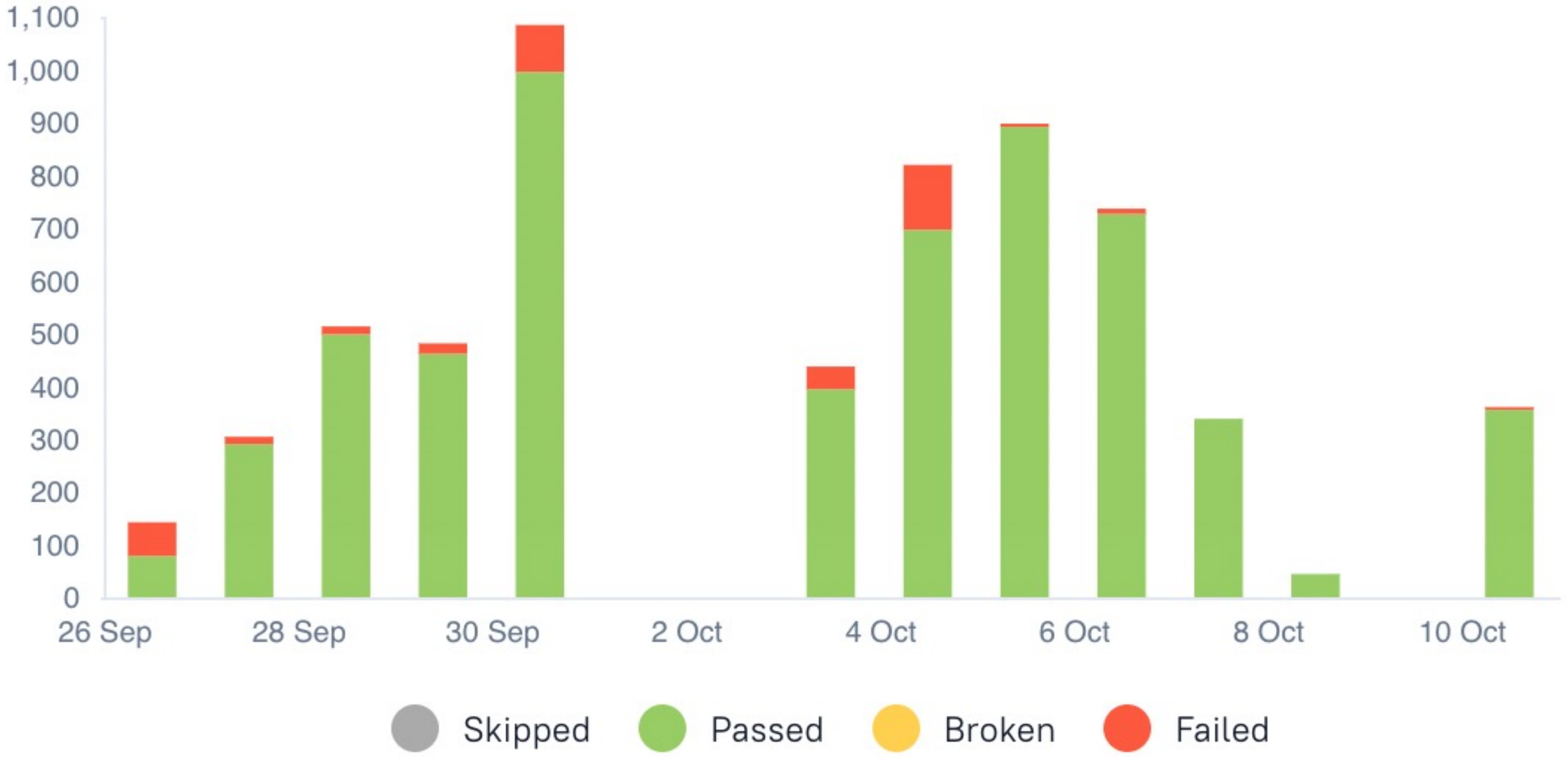
Allure


All
1025
total
1025 Active
Unit Level
648
total
648 Active
Component Level
46
total
46 Active
Api Integration Level
328
total
328 Active
UI Integration Level
3
total
3 Active

Allure

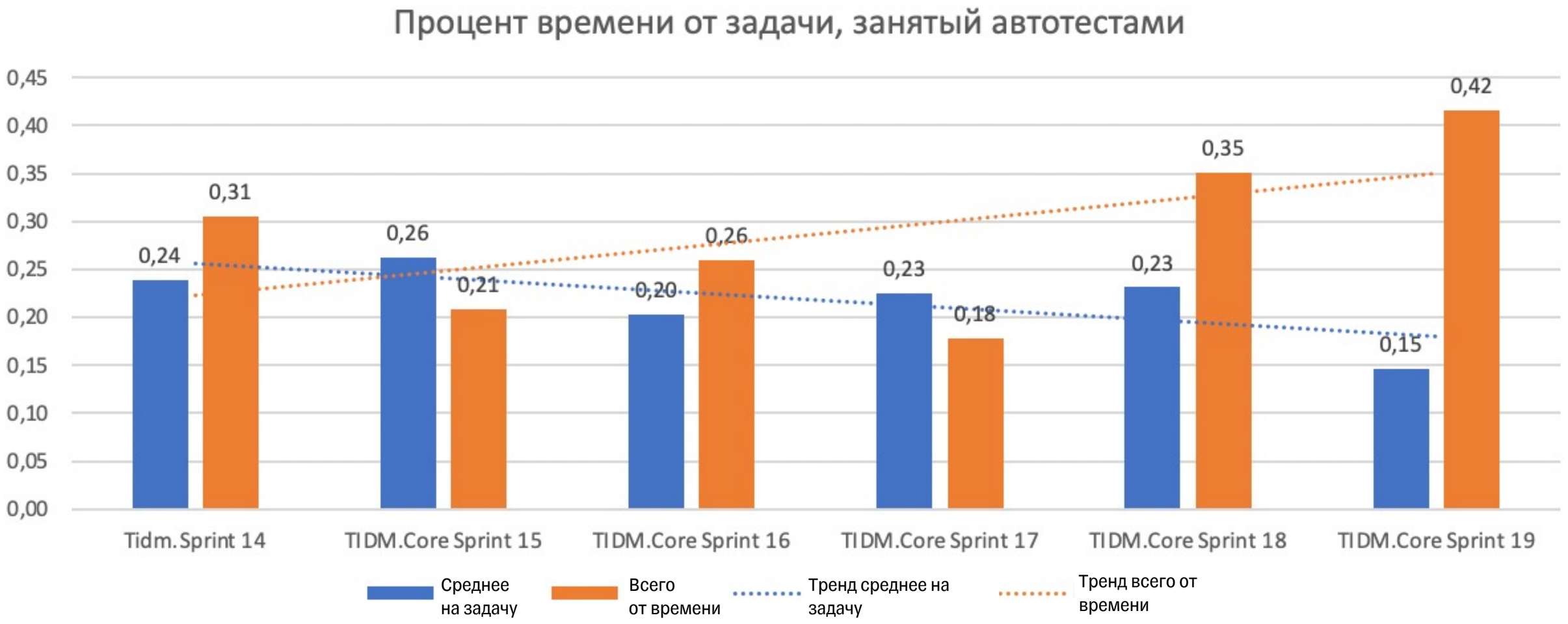

90
80
70
60
50
40
30
20
10
0
26 Sep
28 Sep
30 Sep
2 Oct
4 Oct
6 Oct
8 Oct
10 Oct
Skipped
Passed
Broken
Failed

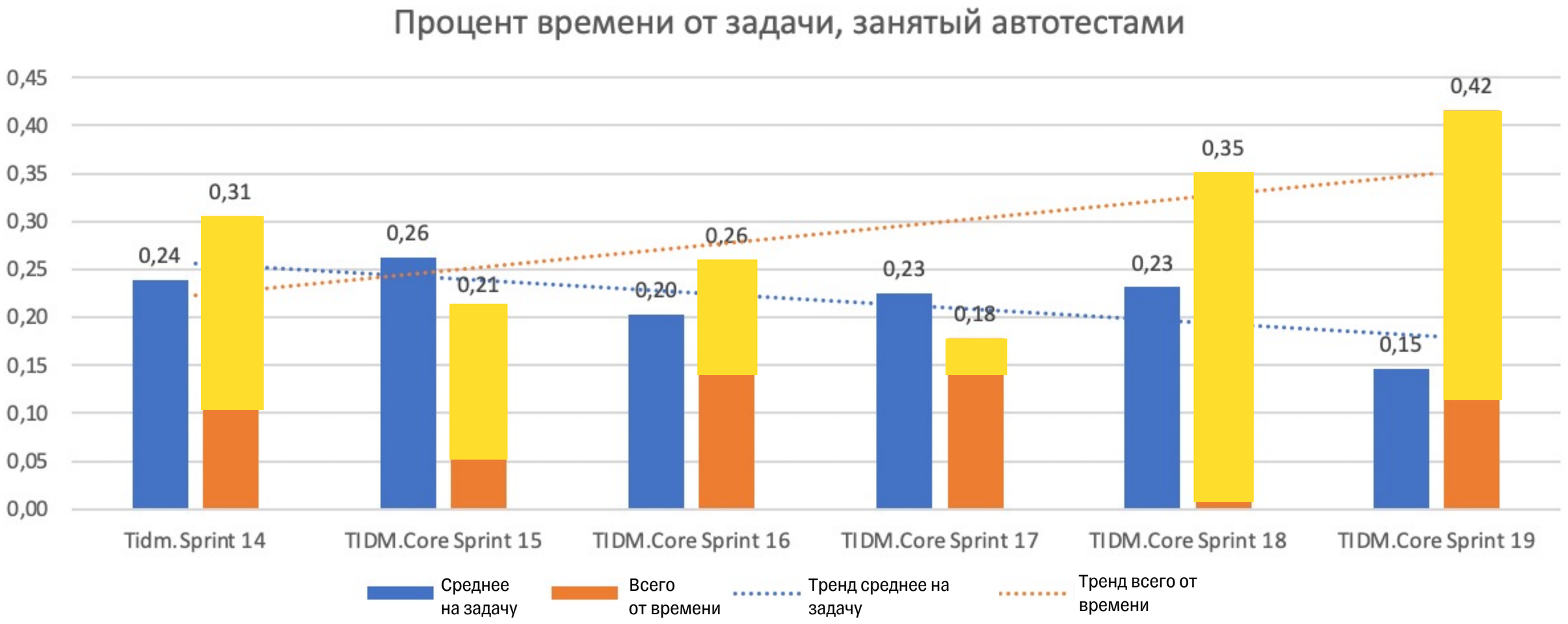
Allure


1,100
1,000
900
800
700
600
500
400
300
200
100
0
26 Sep
28 Sep
30 Sep
2 Oct
4 Oct
6 Oct
8 Oct
10 Oct
Skipped
Passed
Broken
Failed

Команда

- ~~QA~~
- ~~OPS~~
- DEV


Процент времени от задачи, занятый автотестами
0,45
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
0,00
0,24
0,31
0,26
0,21
0,20
0,26
0,23
0,18
0,23
0,35
0,15
0,42
Tidm.Sprint 14
TIDM.Core Sprint 15
TIDM.Core Sprint 16
TIDM.Core Sprint 17
TIDM.Core Sprint 18
TIDM.Core Sprint 19
Среднее на задачу
Всего от времени
Тренд среднее на задачу
Тренд всего от времени


Процент времени от задачи, занятый автотестами
0,45
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
0,00
0,24
0,31
0,26
0,21
0,20
0,26
0,23
0,18
0,23
0,35
0,15
0,42
Tidm.Sprint 14
TIDM.Core Sprint 15
TIDM.Core Sprint 16
TIDM.Core Sprint 17
TIDM.Core Sprint 18
TIDM.Core Sprint 19
Среднее на задачу
Всего от времени
Тренд среднее на задачу
Тренд всего от времени

Фидбэк

- HTTP-фейки равны сервисам и разработчик не может допустить ошибку при написании моков
- Легко поднять локально
- Не надо пропиливать доступы к сторонним системам, потому что, используя второй флоу, так же можно делать заглушки для сторонних сервисов

Планы

- Масштабироваться
- Выложить код
- Анализ тестовой модели

ТИНЬКОФФ

Спасибо

Telegram: @DiakonRai