пОле.рф

Управляй бизнес-процессами. Введение в Temporal для Java-разработчика

Пётр Сальников

руководитель разработки архитектуры

- Больше **20 лет** опыта в IT
- **14+** лет на позиции **Team Lead / Software Architect**
- **6+** лет на позиции **Head of DevOps**
- Проходил путь **монолит – микросервисы - модульный монолит**
- Изучал **психоанализ** 6 лет

Мои цели и о чём буду говорить

- Рассказать про Temporal
- Показать нюансы работы из JVM
- Рассказать о самых частых способах использования

Как реализовать бизнес-процесс

Как реализовать бизнес-процесс

Декларативно

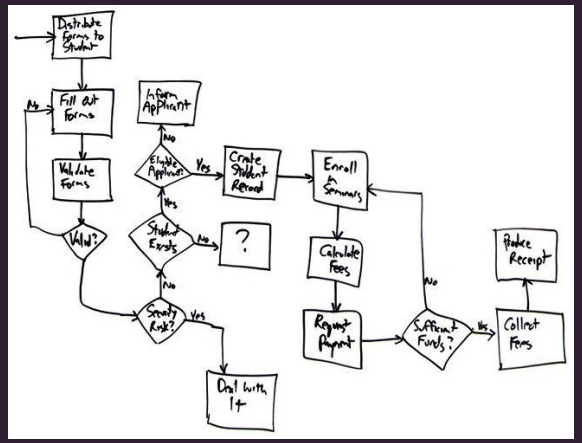
```
order.common: {
  archive-document-type: "ACCREDITATION_DOCUMENTS_ARCHIVE",
  statuses = [
    {
      status: "NOT_STARTED",
      name: "Не начата",
      group: "Не начата",
      actions: {
        "AGRARIAN": {
          confirm: { name: "Пройти", nextStatus: { "default": { status:
        }
      }
    },
    {
      status: "DOCUMENTS_SUBMISSION",
      name: "Подача документов",
      group: "Подача документов",
      actions {
        "AGRARIAN": {
          confirm: { name: "Отправить на проверку", nextStatus: { "defau
          addDocument: { name: "Загрузить", nextStatus: { "default": { s
          updateDocument: { name: "Обновить", nextStatus: { "default": {
          deleteDocument: { name: "Удалить", nextStatus: { "default": {
        },
        "SUPPLIER": {
        },
        "ADMIN": {
          reject: { name: "Отклонить", nextStatus: { "default": { status
          rejectDocument: { name: "Отклонить", nextStatus: { "default":
          confirm: { name: "Отправить на проверку", nextStatus: { "defau
          addDocument: { name: "Загрузить", nextStatus: { "default": { s
          updateDocument: { name: "Обновить", nextStatus: { "default": {
          deleteDocument: { name: "Удалить", nextStatus: { "default": {
        }
      }
    },
```

поле.рф

Как реализовать бизнес-процесс

Декларативно

BPM



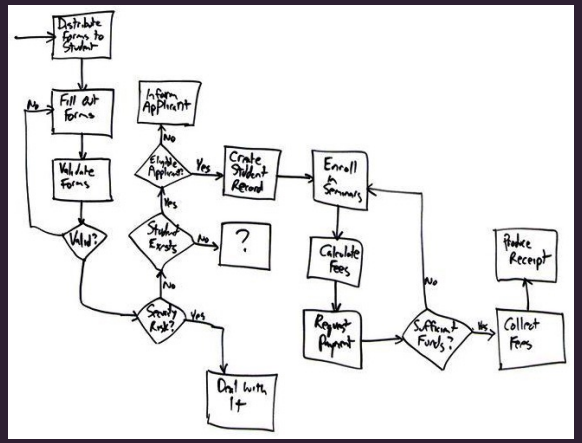

```
order.common: {
  archive-document-type: "ACCREDITATION_DOCUMENTS_ARCHIVE",
  statuses = [
    {
      status: "NOT_STARTED",
      name: "Не начата",
      group: "Не начата",
      actions: {
        "AGRARIAN": {
          confirm: { name: "Пройти", nextStatus: { "default": { status:
        }
      }
    },
    {
      status: "DOCUMENTS_SUBMISSION",
      name: "Подача документов",
      group: "Подача документов",
      actions {
        "AGRARIAN": {
          confirm: { name: "Отправить на проверку", nextStatus: { "defau
          addDocument: { name: "Загрузить", nextStatus: { "default": { s
          updateDocument: { name: "Обновить", nextStatus: { "default": {
          deleteDocument: { name: "Удалить", nextStatus: { "default": {
        },
        "SUPPLIER": {
        },
        "ADMIN": {
          reject: { name: "Отклонить", nextStatus: { "default": { status
          rejectDocument: { name: "Отклонить", nextStatus: { "default":
          confirm: { name: "Отправить на проверку", nextStatus: { "defau
          addDocument: { name: "Загрузить", nextStatus: { "default": { s
          updateDocument: { name: "Обновить", nextStatus: { "default": {
          deleteDocument: { name: "Удалить", nextStatus: { "default": {
        }
      }
    },
```

поле.рф

Как реализовать бизнес-процесс

Декларативно

BPM




```
order.common: {
  archive-document-type: "ACCREDITATION_DOCUMENTS_ARCHIVE",
  statuses = [
    {
      status: "NOT_STARTED",
      name: "Не начата",
      group: "Не начата",
      actions: {
        "AGRARIAN": {
          confirm: { name: "Пройти", nextStatus: { "default": { status:
        }
      }
    },
    {
      status: "DOCUMENTS_SUBMISSION",
      name: "Подача документов",
      group: "Подача документов",
      actions {
        "AGRARIAN": {
          confirm: { name: "Отправить на проверку", nextStatus: { "defau
          addDocument: { name: "Загрузить", nextStatus: { "default": { s
          updateDocument: { name: "Обновить", nextStatus: { "default": {
          deleteDocument: { name: "Удалить", nextStatus: { "default": {
        },
        "SUPPLIER": {
        },
        "ADMIN": {
          reject: { name: "Отклонить", nextStatus: { "default": { status
          rejectDocument: { name: "Отклонить", nextStatus: { "default":
          confirm: { name: "Отправить на проверку", nextStatus: { "defau
          addDocument: { name: "Загрузить", nextStatus: { "default": { s
          updateDocument: { name: "Обновить", nextStatus: { "default": {
          deleteDocument: { name: "Удалить", nextStatus: { "default": {
        }
      }
    },
```

В коде Kotlin

```kotlin
fun requestProcess(requestDto: RequestDto) {
    val saga = Saga(options)
    try {
        val requestId = request.create(requestDto)
        saga.addCompensation(request::cancelCreate, requestId)

        val requestParty = request.createParty(requestId, requestDto)
        saga.addCompensation(request::cancelCreateParty, requestParty)
    }catch (e: ActivityFailure){
        saga.compensate()
    }
}
```

поле.рф

Temporal – что это такое?

Temporal – что это такое?

- Оркестратор кода

Temporal – что это такое?

- Оркестратор кода
- Бизнес-процесс описывает кодом – Workflow

Temporal – что это такое?

- Оркестратор кода
- Бизнес-процесс описывает кодом – Workflow
- Работает на многих языках (Java, GO, TS, PHP, Python, .NET)

Temporal – что это такое?

- Оркестратор кода
- Бизнес-процесс описывает кодом – Workflow
- Работает на многих языках (Java, GO, TS, PHP, Python, .NET)
- Отказоустойчивость через Retry, Save Point, Rate Limit

Temporal – что это такое?

- Оркестратор кода
- Бизнес процесс описывает кодом – Workflow
- Работает на многих языках (Java, GO, TS, PHP, Python, .NET)
- Отказоустойчивость через Retry, Save Point, Rate Limit
- Фоновые процессы с Cron запуском

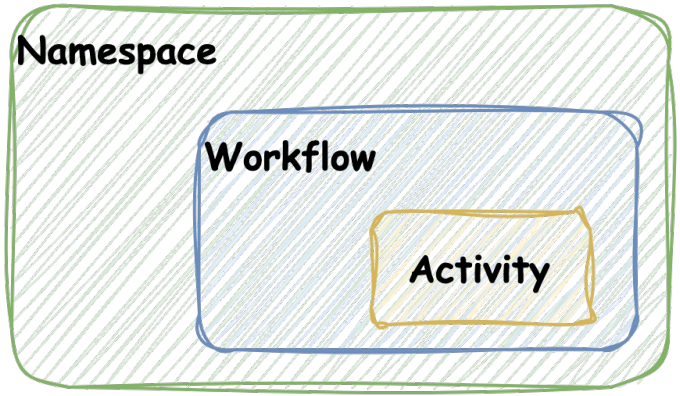
Temporal – что это такое?

- Оркестратор кода
- Бизнес процесс описывает кодом – Workflow
- Работает на многих языках (Java, GO, TS, PHP, Python, .NET)
- Отказоустойчивость через Retry, Save Point, Rate Limit
- Фоновые процессы с Cron запуском
- Паттерн SAGA

Ключевые понятия

Workflow - гибкая программа, которая выполняет задачи, реагирует на внешние события, включая таймеры и таймауты

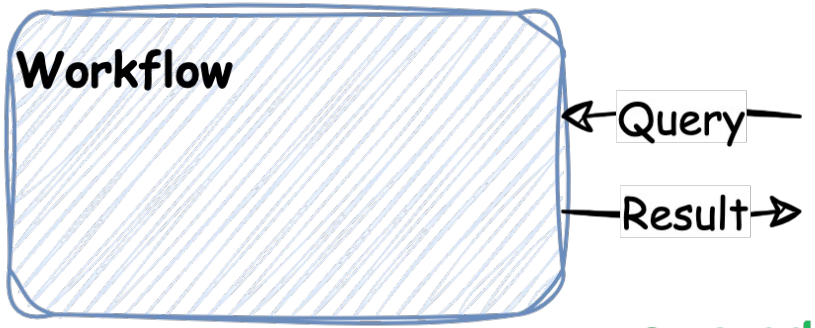
- **Namespace**
- **Workflow**
- **Activity**



Ключевые понятия

Workflow - гибкая программа, которая выполняет задачи, реагирует на внешние события, включая таймеры и таймауты

- **Namespace**
- **Workflow**
- **Activity**
- **Signal**
- **Query**

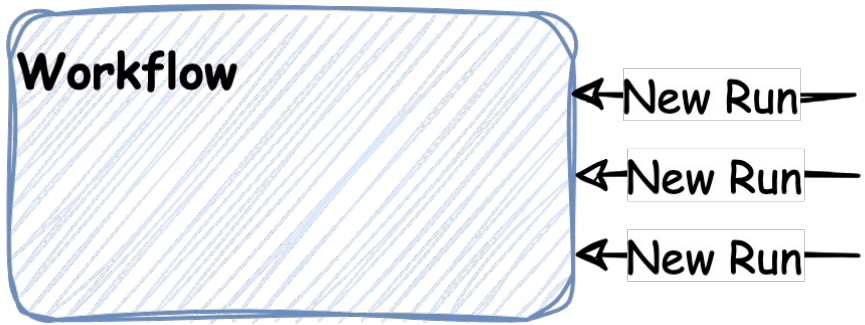


поле.рф

Ключевые понятия

Workflow - гибкая программа, которая выполняет задачи, реагирует на внешние события, включая таймеры и таймауты

- **Namespace**
- **Workflow**
- **Activity**
- **Signal**
- **Query**
- **Timer**



Ключевые понятия

Workflow / Workflow Run

- Метод в Java объекте
- Синхронизация вызова методов снаружи
- Время жизни не контролируется

Ключевые понятия

Workflow / Workflow Run

- Метод в Java объекте
- Синхронизация вызова методов снаружи
- Время жизни не контролируется

Activity

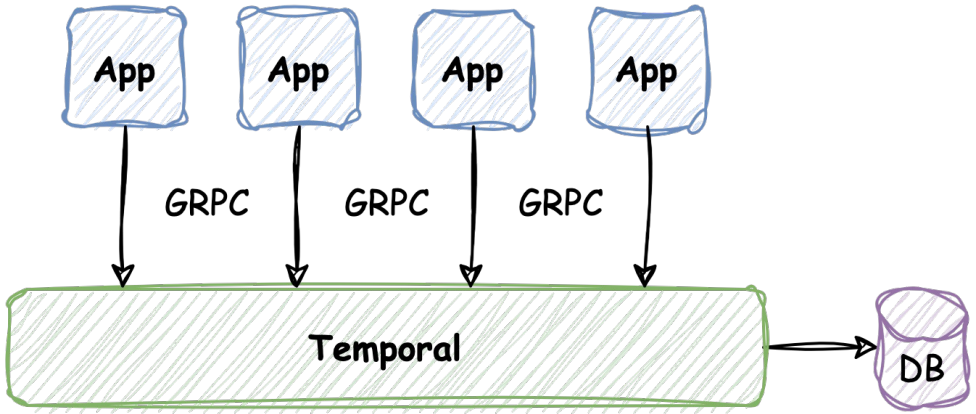
- Метод в IoC компоненте
- Идемпотентность
- Используются для вызова внешних систем/БД
- Можно запускать параллельно

Устойчивость к сбоям

- Retry механика
- Save Point - через историю выполнения
- Temporal - это не база данных
- Передавать большие данные – вне Input/Output значений
- Soft limit 256KB / Hard Limit 2MB

Развёртывание

- Helm для Kubernetes
- Docker Compose как пример для Standalone
- На уровне сети: App –> Temporal <– App



Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

```
14: override fun requestProcess(requestDto: RequestDto) {
15:     val saga = Saga(options)
16:     try {
17:         log.info("Creating request")
18:         val requestId = requestActivity.create(requestDto)
19:         saga.addCompensation(requestActivity::cancelCreate, requestId)
20:
21:         log.info("Creating request($requestId) party")
22:         val requestPartyId = partyActivity.createParty(requestId, requestDto)
23:         saga.addCompensation(partyActivity::cancelCreateParty, requestPartyId)
24:
25:         log.info("Linking party($requestPartyId) to request($requestId)")
26:         val requestPartyLinkRef = linkActivity.linkParty(requestId, requestPartyId)
27:         saga.addCompensation(linkActivity::unlinkParty, requestPartyLinkRef)
28:     }catch (e: ActivityFailure){
29:         saga.compensate()
20:     }
31: }
```

Механика на уровне JVM

Нельзя

- println()
- synchronized
- Работа с другими системами вне Activity

Можно

- Workflow.logger()
- Синхронное/асинхронное выполнение Activity
- Использовать поля объекта Workflow

Механика на уровне JVM – Query, Signal

Вызов Query/Signal ->

Создание объекта и выполнение всех строк кода до Save Point

Вывод =>

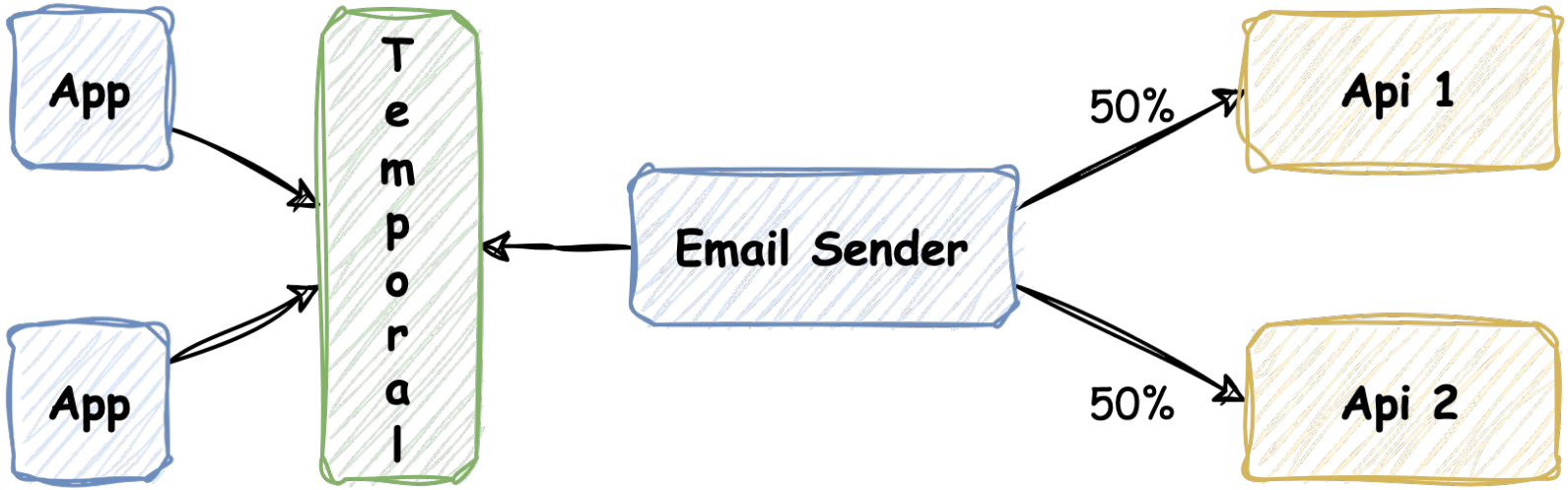
Используйте Query/Signal аккуратно и с пониманием работы кода

Состояние бизнес сущности хранить в БД и проверять там

Где мы применяем

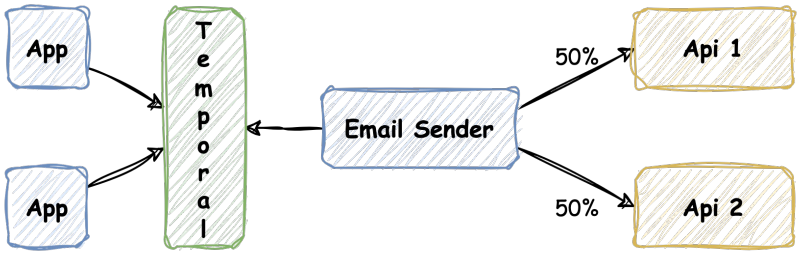
- Интеграция с внешними системами
- Запуск фоновых Job-ов
- Паттерн SAGA
- Событийная модель – publisher/subscriber

Интеграция с внешними системами



Интеграция с внешними системами

- Retry делает за нас
- Можно настроить Backoff Limit
- Есть Rate Limiting на реплику и на очередь
- Легко перезапустить отправку при долгой недоступности силами Support команды



Регулярные процессы

Бесконечный Workflow

- Удобно в сопровождении
- Лимит Temporal на историю запусков

Регулярные процессы

Бесконечный Workflow

- Удобно в сопровождении
- Лимит Temporal на историю запусков

Workflow Schedule

- Интервальный или по расписанию

Регулярные процессы

Бесконечный Workflow

- Удобно в сопровождении
- Лимит Temporal на историю запусков

Workflow Schedule

- Интервальный или по расписанию

Выносить в свой Namespace

Регулярные процессы

Бесконечный Workflow

- Удобно в сопровождении
- Лимит Temporal на историю запусков

Workflow Schedule

- Интервальный или по расписанию

Выносить в свой Namespace

Одновременный запуск многих Job-ов

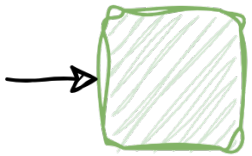
- Защита от лавины Activity через настройки Capacity на Worker

Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity

Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity



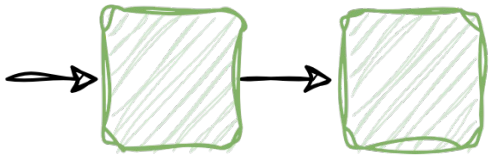
поле.рф

Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity

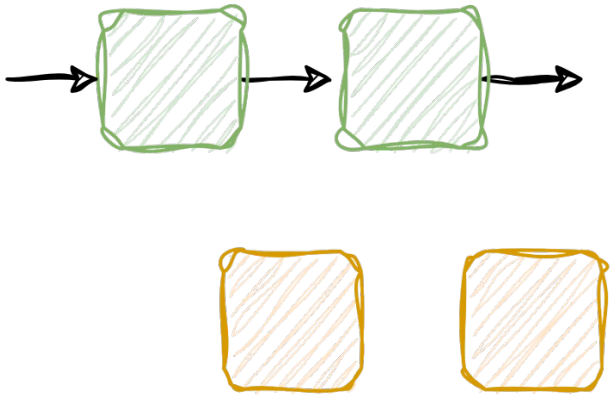
Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity



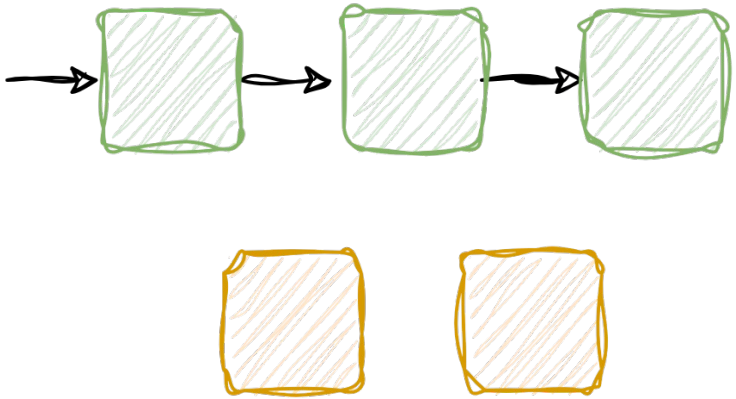
Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity



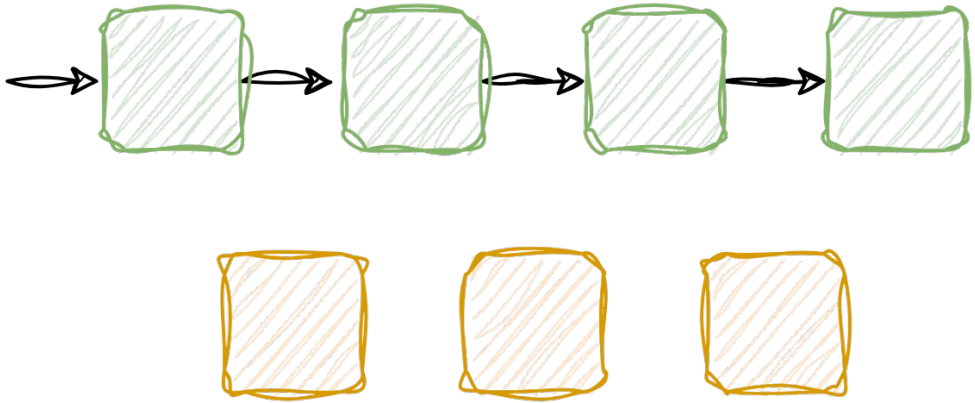
Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity



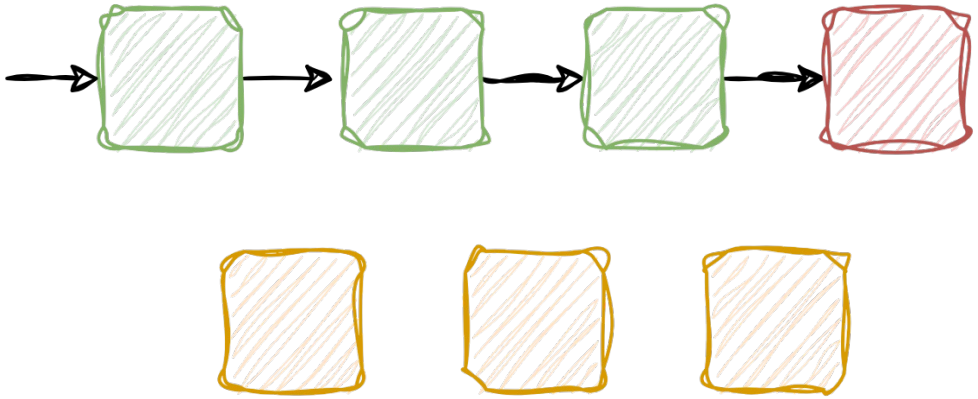
Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity



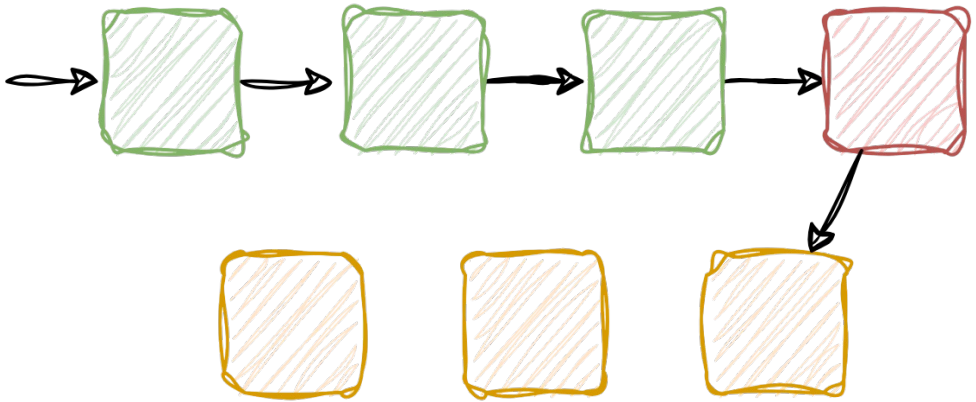
Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity



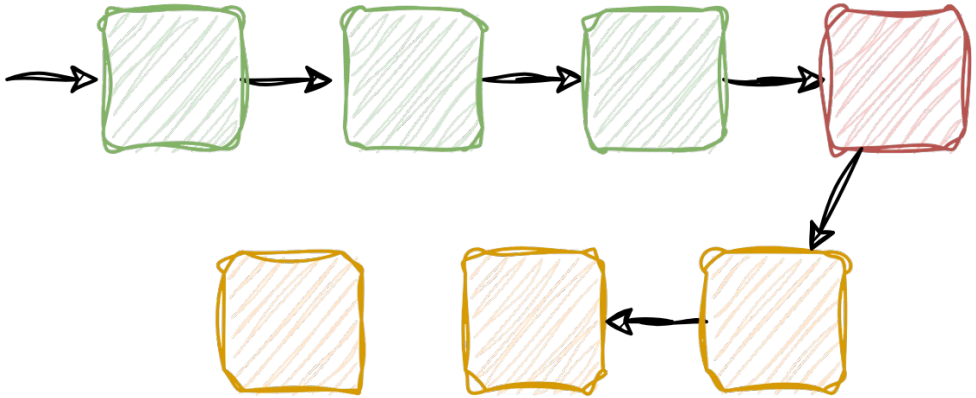
Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity



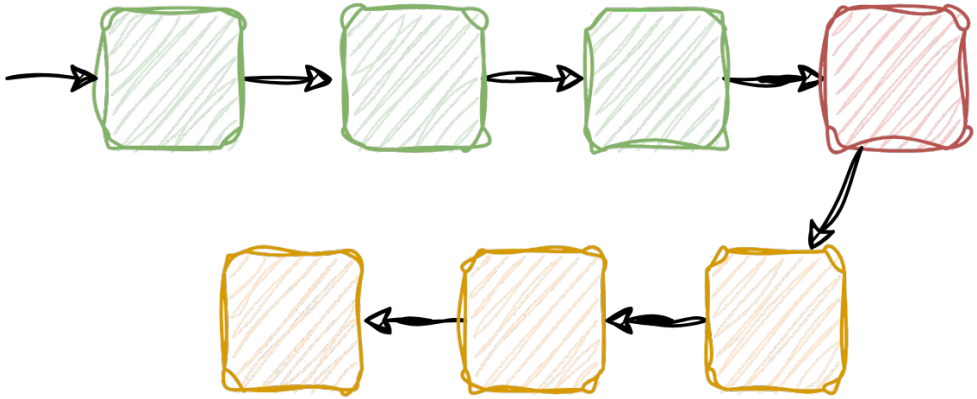
Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity



Паттерн SAGA

- Temporal реализует распределённую SAGA через оркестрацию
- Операция целиком - Workflow
- Прямое действие – через Activity
- Компенсирующий коммит – тоже через Activity

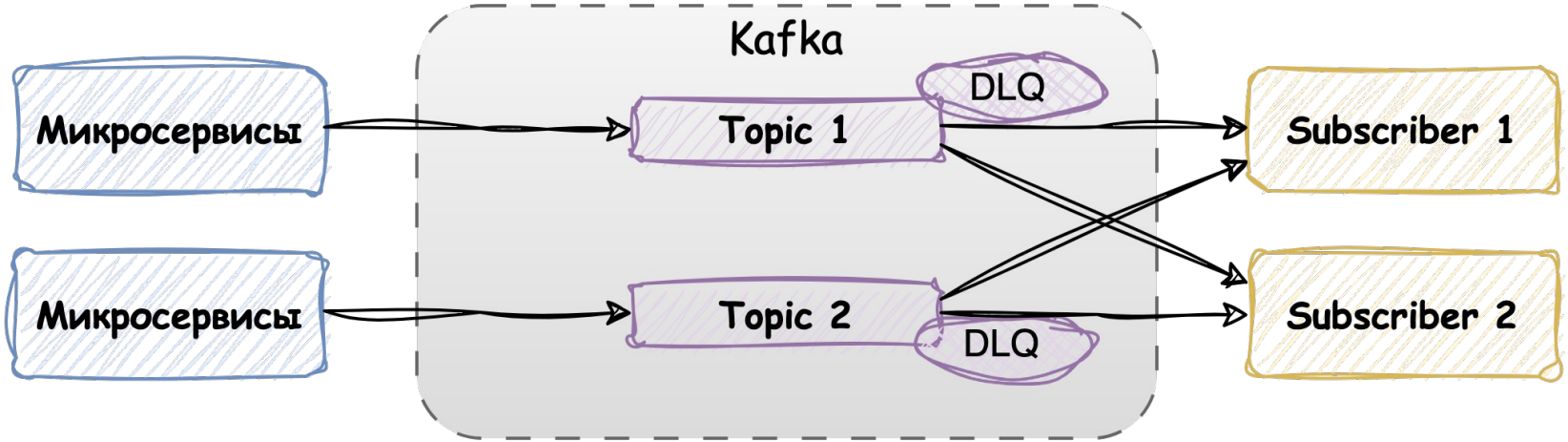


поле.рф

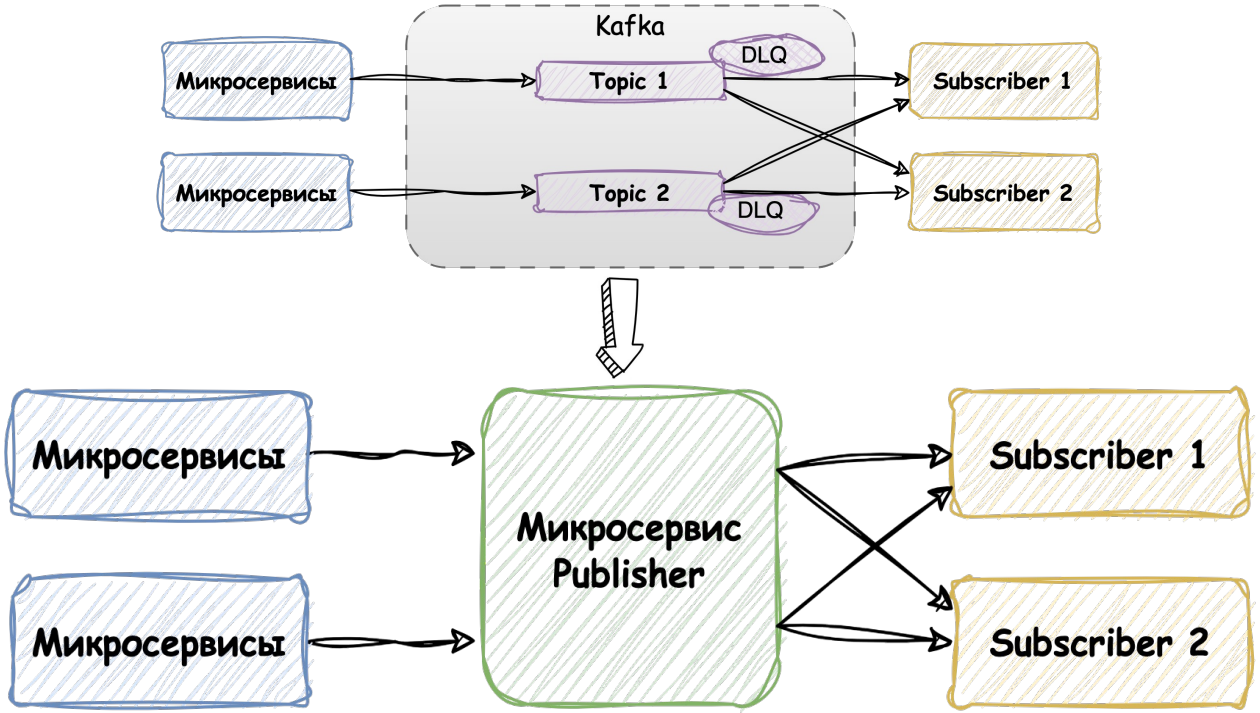
Паттерн publisher/subscriber

- Kafka это хорошо, но что если она избыточна в проекте?
- Если у нас уже есть Temporal, можем ли мы на нём реализовать паттерны из Kafka?
- Мы решили попробовать

Паттерн publisher/subscriber



Паттерн publisher/subscriber



Паттерн publisher/subscriber

1. Workflow с бизнес-процессом

Паттерн publisher/subscriber

1. Workflow с бизнес-процессом
2. На каждое событие, которое нужно отправить - выполняем Activity из Publisher-а, которое работает в паттерне Outbox

Паттерн publisher/subscriber

1. Workflow с бизнес-процессом
2. На каждое событие, которое нужно отправить - выполняем Activity из Publisher-а, которое работает в паттерне Outbox
3. Activity запускает новый Workflow в Publisher-е

Паттерн publisher/subscriber

2. На каждое событие, которое нужно отправить - выполняем Activity из Publisher-а, которое работает в паттерне Outbox
3. Activity запускает новый Workflow в Publisher-е
4. В Workflow используются Activity для каждого Subscriber-а, который запускает новый Workflow из Subscriber-а

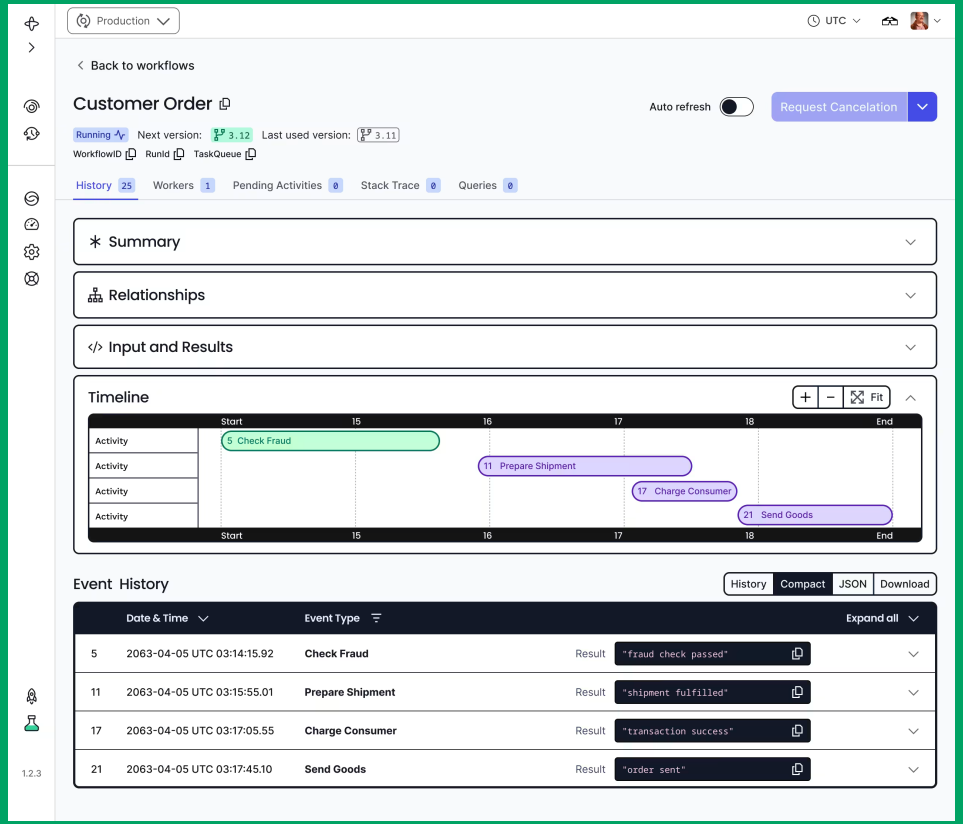
Паттерн publisher/subscriber

3. Activity запускает новый Workflow в Publisher-е
4. В Workflow используются Activity для каждого Subscriber-а, который запускает новый Workflow из Subscriber-а
5. Каждый Subscriber реализует свой Workflow, свой Rate Limiting и свой Retry Policy

Паттерн publisher/subscriber

4. В Workflow используются Activity для каждого Subscriber-а, который запускает новый Workflow из Subscriber-а
5. Каждый Subscriber реализует свой Workflow, свой Rate Limiting и свой Retry Policy
6. Добавляем Signal + внешнюю БД – получаем FIFO

Как делать Troubleshooting



- Temporal UI
- Свободное именование Workflow/Activity
- Метрики, много метрик
- Server
- Client SDK
- Поиск по Custom Search Attribute

Happy End

- Познакомились с Temporal и его основными понятиями
- Увидели нюансы работы в JVM и как работает Retry
- Посмотрели на 4 варианта использования:
- Интеграция / Фоновые процессы / Паттерны Saga и PubSub
- Ждём в гости на стенде поле.рф

поле.рф

Примеры

Задать вопросы