Яндекс 360

Стейт-машины
(The Good, The Bad and The Ugly)

Как мы в Биллинге 360 стейт-машины причесывали

Дарья Андреева
14 Октября 2023

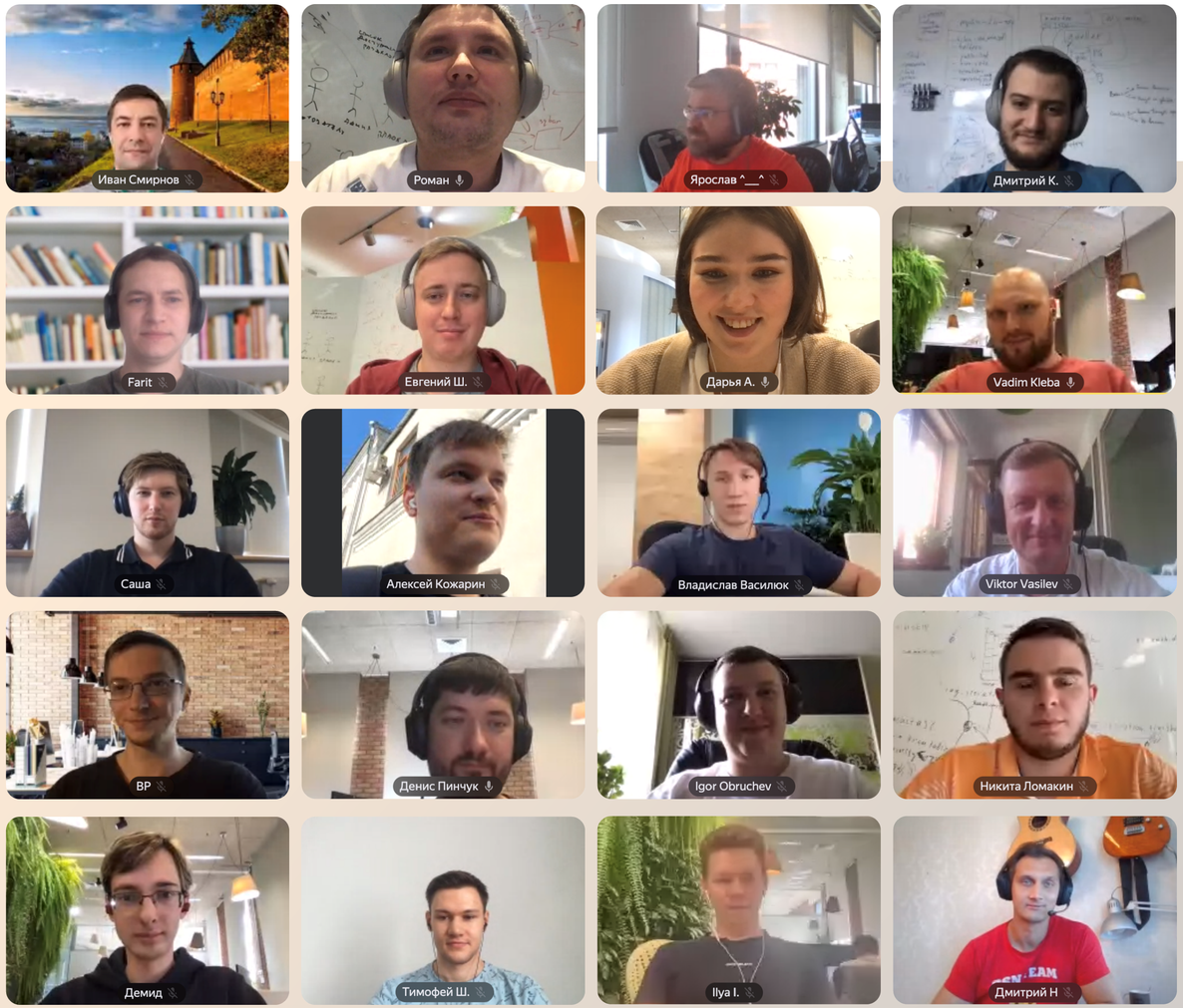
Дарья Андреева

Разработчик в Яндекс 360

Люблю:

- Хорошо работающие процессы
- Качественно спроектированные продукты
- Интересные и важные задачи

Команда 360


Иван Смирнов
Роман
Ярослав ^__^
Дмитрий К.
Farit
Евгений Ш.
Дарья А.
Vadim Kleba
Саша
Алексей Кожарин
Владислав Василюк
Viktor Vasilev
BP
Денис Пинчук
Igor Obruchev
Никита Ломакин
Демид
Тимофей Ш.
Ilya I.
Дмитрий Н

Что такое Биллинг 360

- Яндекс 360 - это виртуальный офис для пользователей и организаций, который помогает решать задачи с помощью сервисов Яндекса.
- Биллинг 360 — сервис отвечающий за предварительную обработку платежных данных, формирование тарифов, расчёт стоимости оказанных услуг, подписочную модель, управлением доступными услугами. Предложить тариф, довести до оплаты, включить оплаченные услуги.

Мы обслуживаем

> 100.000.000 запросов

и обрабатываем десятки миллионов асинхронных задач в сутки

billing-360.notion.site

Действующие лица

Команда

Бизнес

Про что
я буду рассказывать

Команда

Бизнес

01

Как мы подходим к проработке архитектуры в Яндекс.360

02

Про «Промокоды для студентов»: как мы научились проектировать и писать стейт-машины

03

Не Спрингом единым: как мы реализовали свой кастомный движок в Телемосте

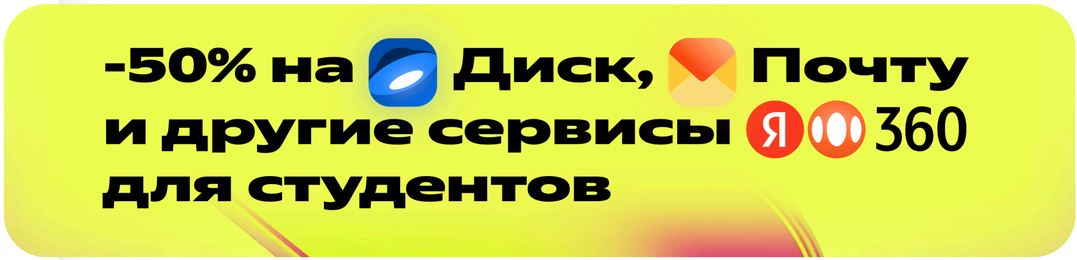
04

Не стейт-машиной единой: почему важно использовать подходящие модели

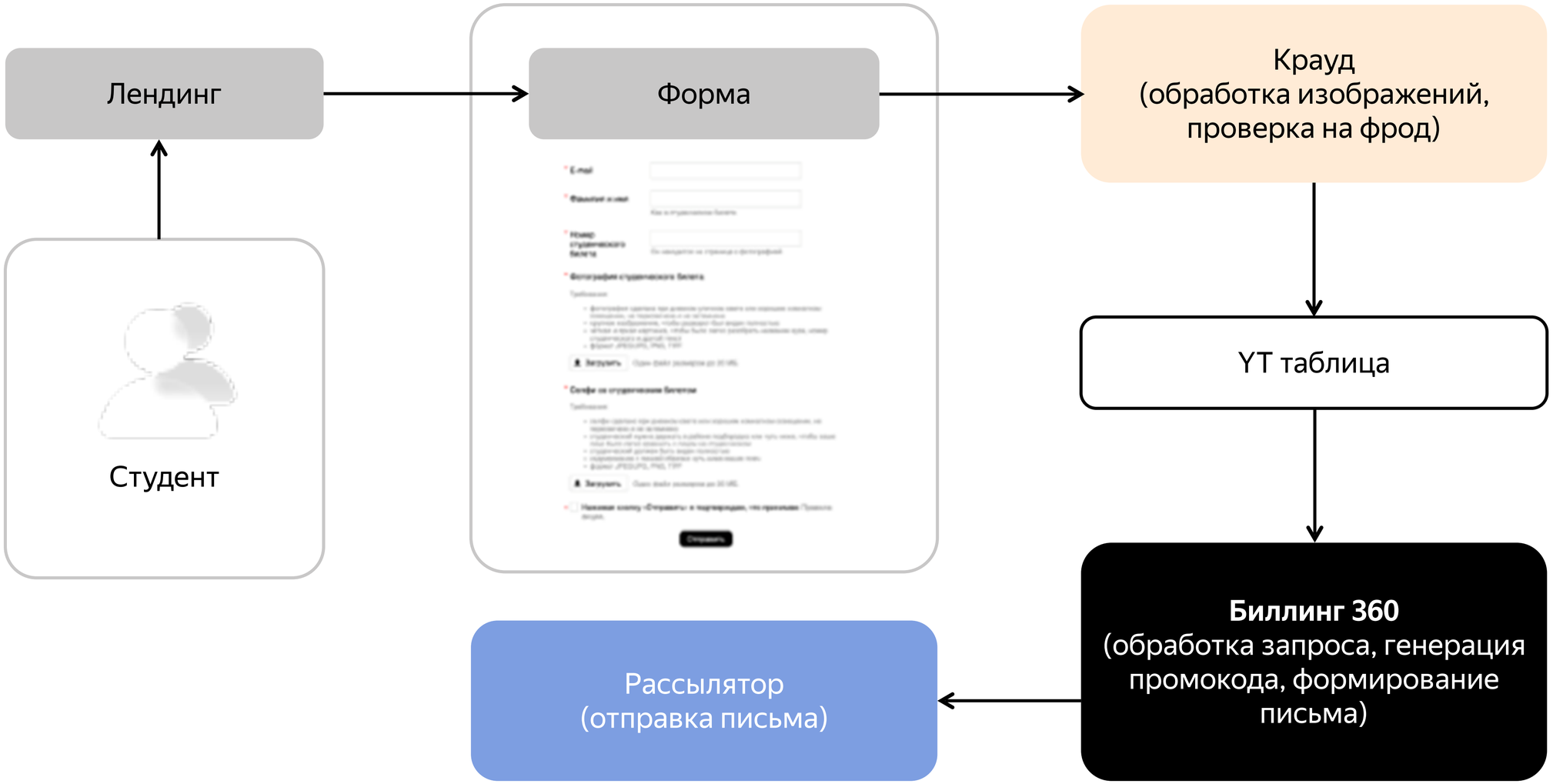
Как мы решили использовать стейт машину

Бизнес

Сервис анализа изображений анализирует селфи студента и в зависимости от результата проверки и истории предыдущих проверок выслать сообщение с промокодом через сервис рассылки сообщений.

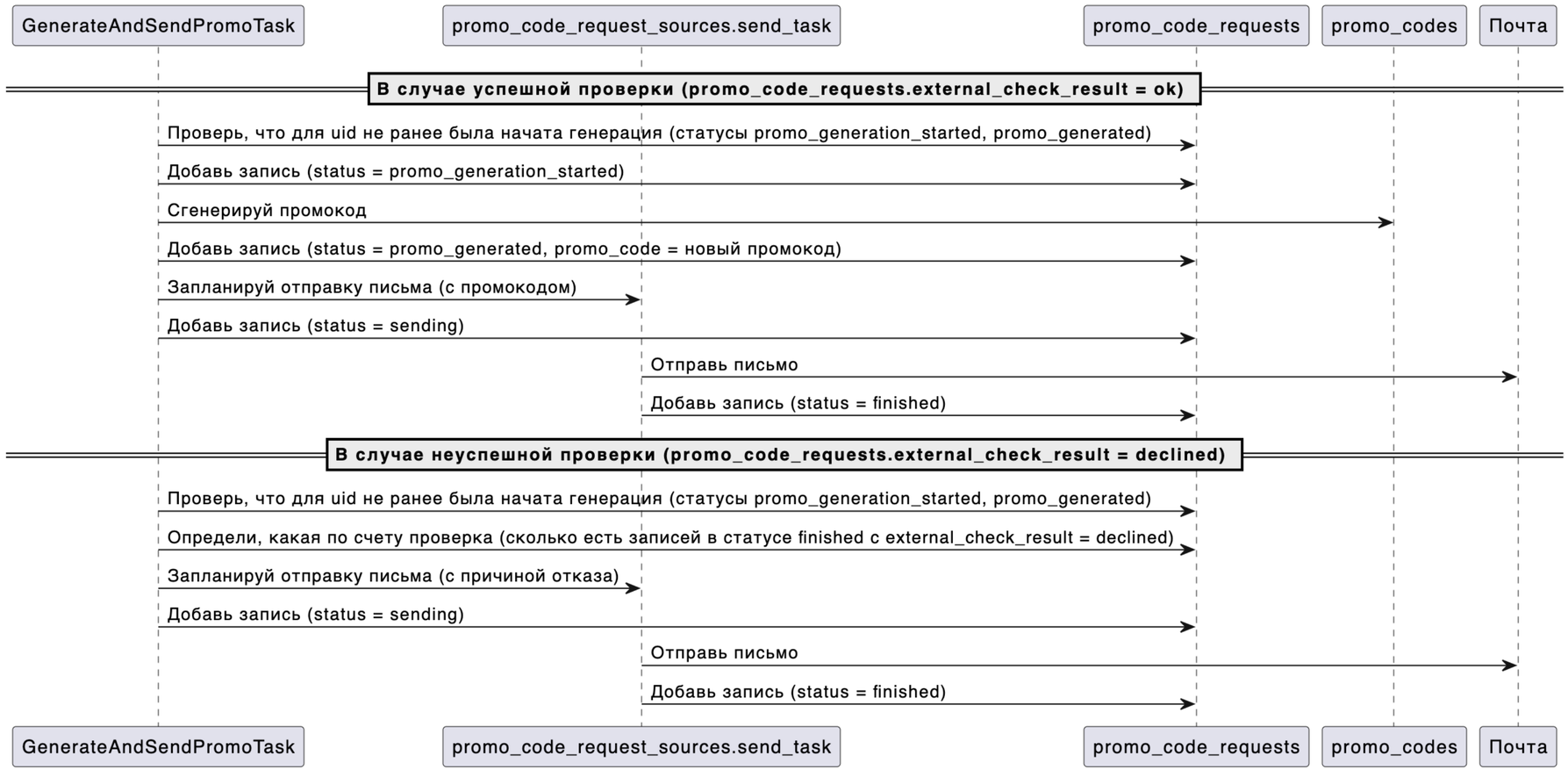


https://360.yandex.ru/student


Лендинг
Форма
Крауд
(обработка изображений,
проверка на фрод)
Студент
YT таблица
Биллинг 360
(обработка запроса, генерация
промокода, формирование
письма)
Рассылятор
(отправка письма)

Проект «Промокоды для студентов»

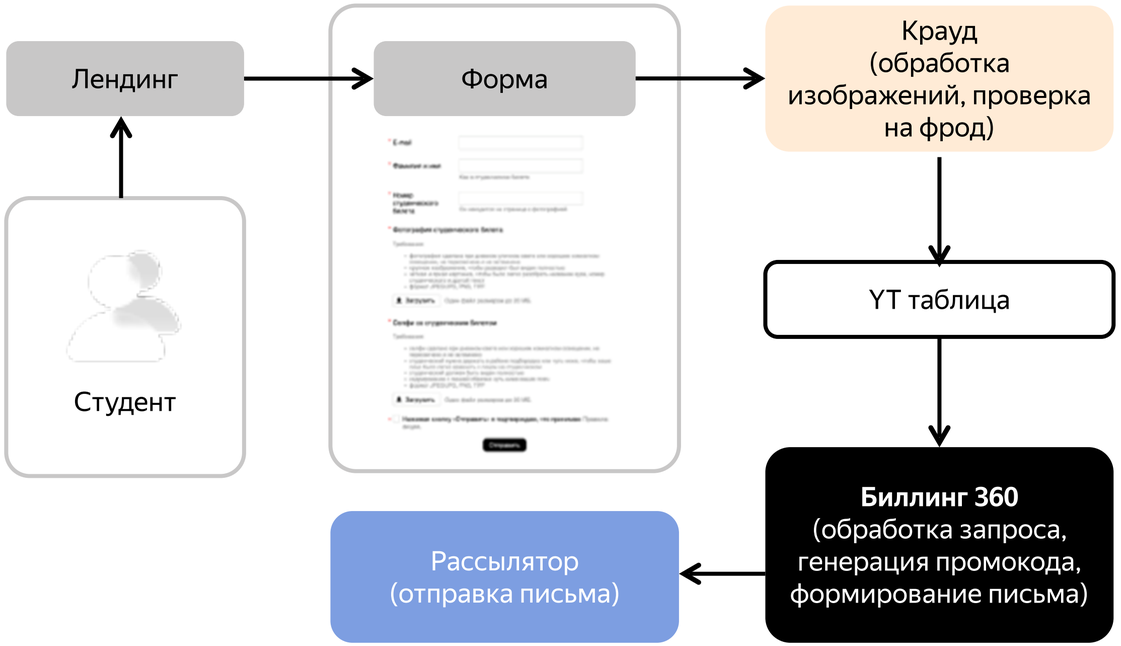
Проработка без стейт-машины



Как мы решили использовать стейт машину

Команда

Проект «Промокоды для студентов»



Что если обработаем один и тот же запрос несколько раз?

Что делать, если на середине случится отказ? Как обеспечить гарантированную доставку сообщения?

Что делать, если рассылка писем упадет? Как узнать об этом и переслать сообщение?

Как переиспользовать это решение в дальнейшем? Не исключено, что нам понадобится процесс генерации промокодов по какому-то запросу

Как мы решили использовать стейт машину

Команда

Проект «Промокоды для студентов»

Что если обработаем один и тот же запрос несколько раз?

Что делать, если на середине случится отказ? Как обеспечить гарантированную доставку сообщения?

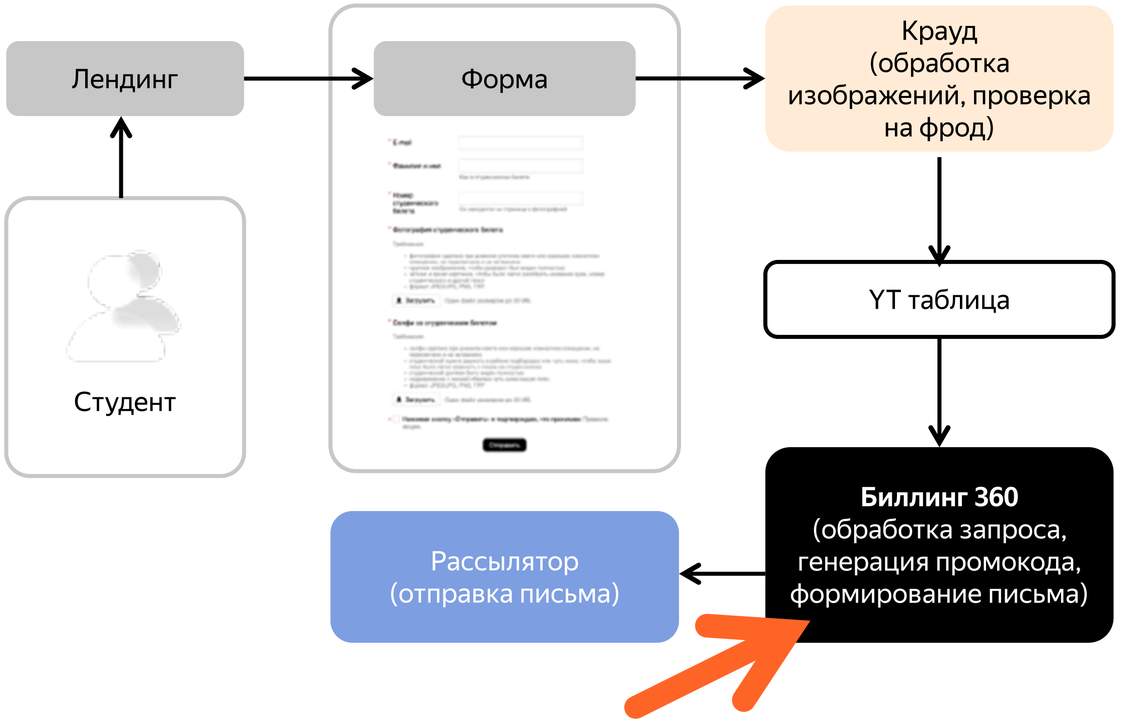
Что делать, если рассылка писем упадет? Как узнать об этом и переслать сообщение?

Как переиспользовать это решение в дальнейшем? Не исключено, что нам понадобится процесс генерации промокодов по какому-то запросу

Как мы решили использовать стейт машину

Команда

Проект «Промокоды для студентов»



Что если обработаем один и тот же запрос несколько раз?

Что делать, если на середине случится отказ? Как обеспечить гарантированную доставку сообщения?

Что делать, если рассылка писем упадет? Как узнать об этом и переслать сообщение?

Как переиспользовать это решение в дальнейшем? Не исключено, что нам понадобится процесс генерации промокодов по какому-то запросу

Как мы решили использовать стейт машину

Команда

Проект «Промокоды для студентов»

Что если обработаем один и тот же запрос несколько раз?

Что делать, если на середине случится отказ? Как обеспечить гарантированную доставку сообщения?

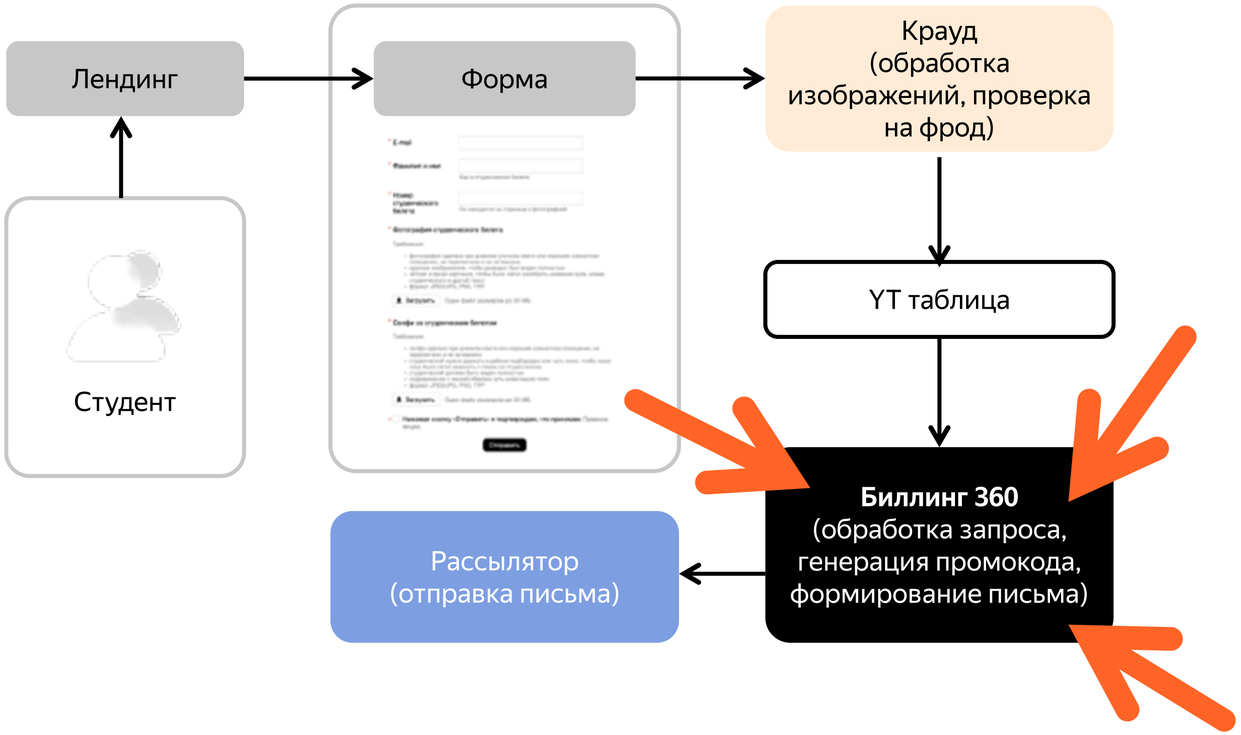
Что делать, если рассылка писем упадет? Как узнать об этом и переслать сообщение?

Как переиспользовать это решение в дальнейшем? Не исключено, что нам понадобится процесс генерации промокодов по какому-то запросу

Как мы решили использовать стейт машину

Команда

Проект «Промокоды для студентов»



Что если обработаем один и тот же запрос несколько раз?

Что делать, если на середине случится отказ? Как обеспечить гарантированную доставку сообщения?

Что делать, если рассылка писем упадет? Как узнать об этом и переслать сообщение?

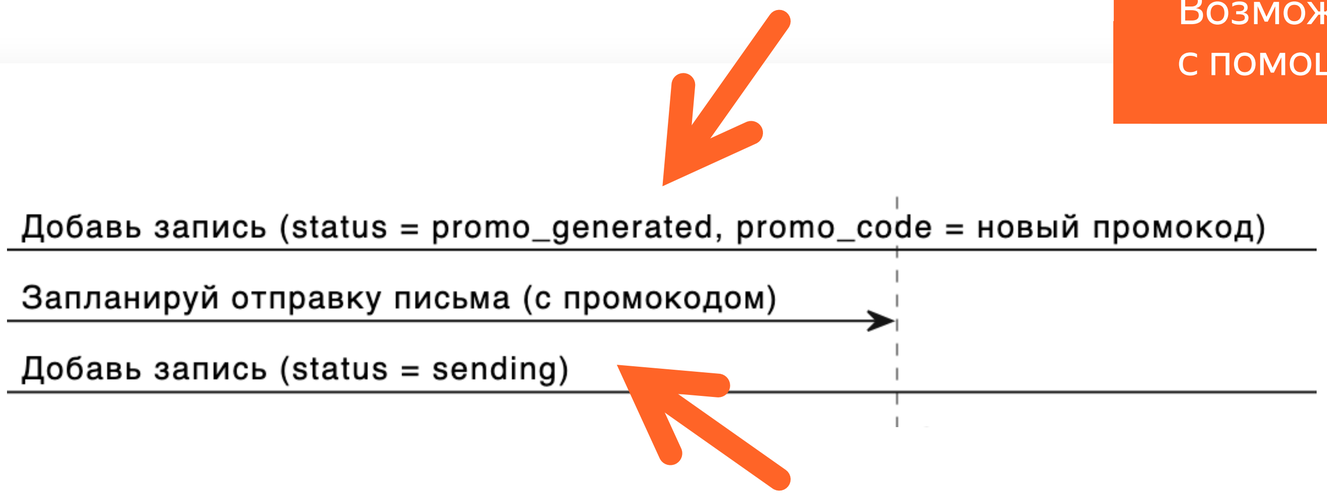
Как переиспользовать это решение в дальнейшем? Не исключено, что нам понадобится процесс генерации промокодов по какому-то запросу

Проект «Промокоды для студентов»

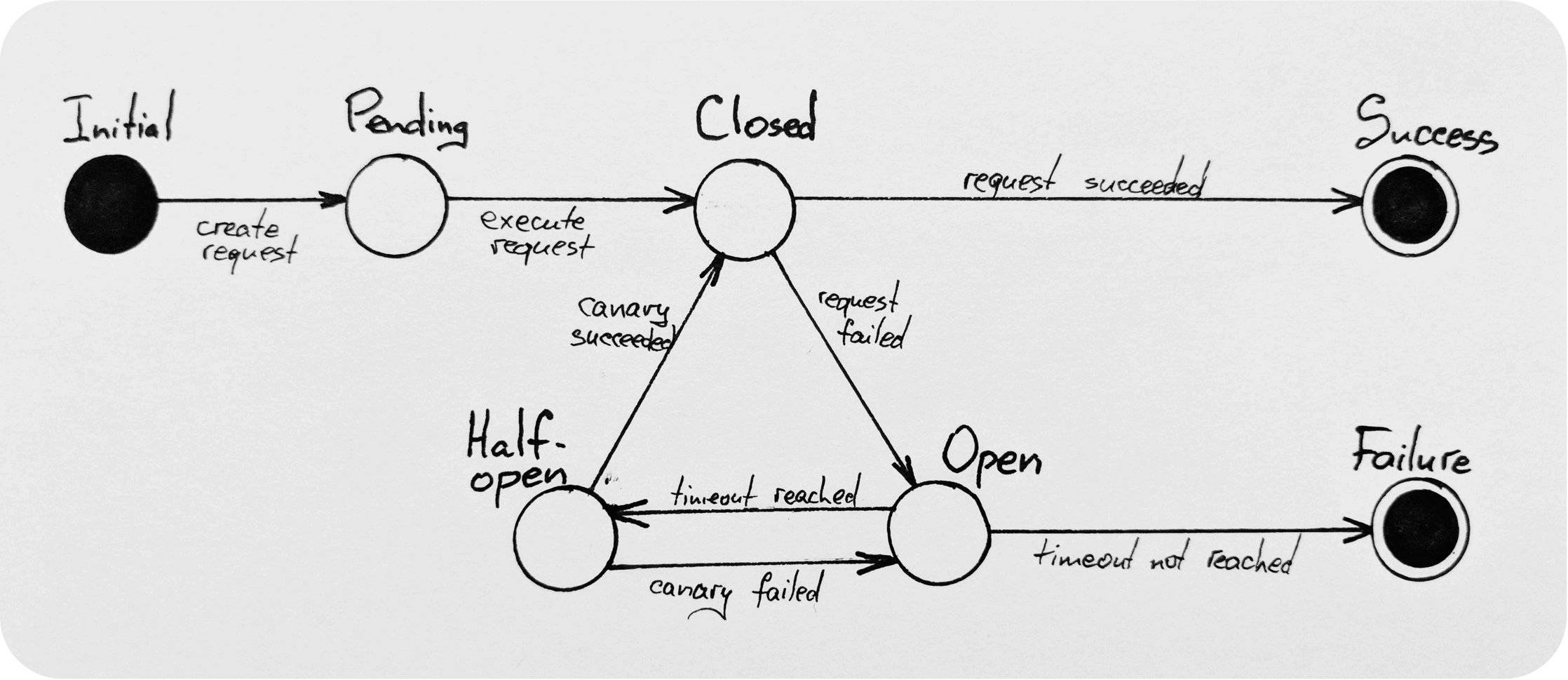
Проработка без стейт-машины

При каждом действии мы меняем статус запроса

Возможно, лучше решать эту задачу с помощью стейт машины?



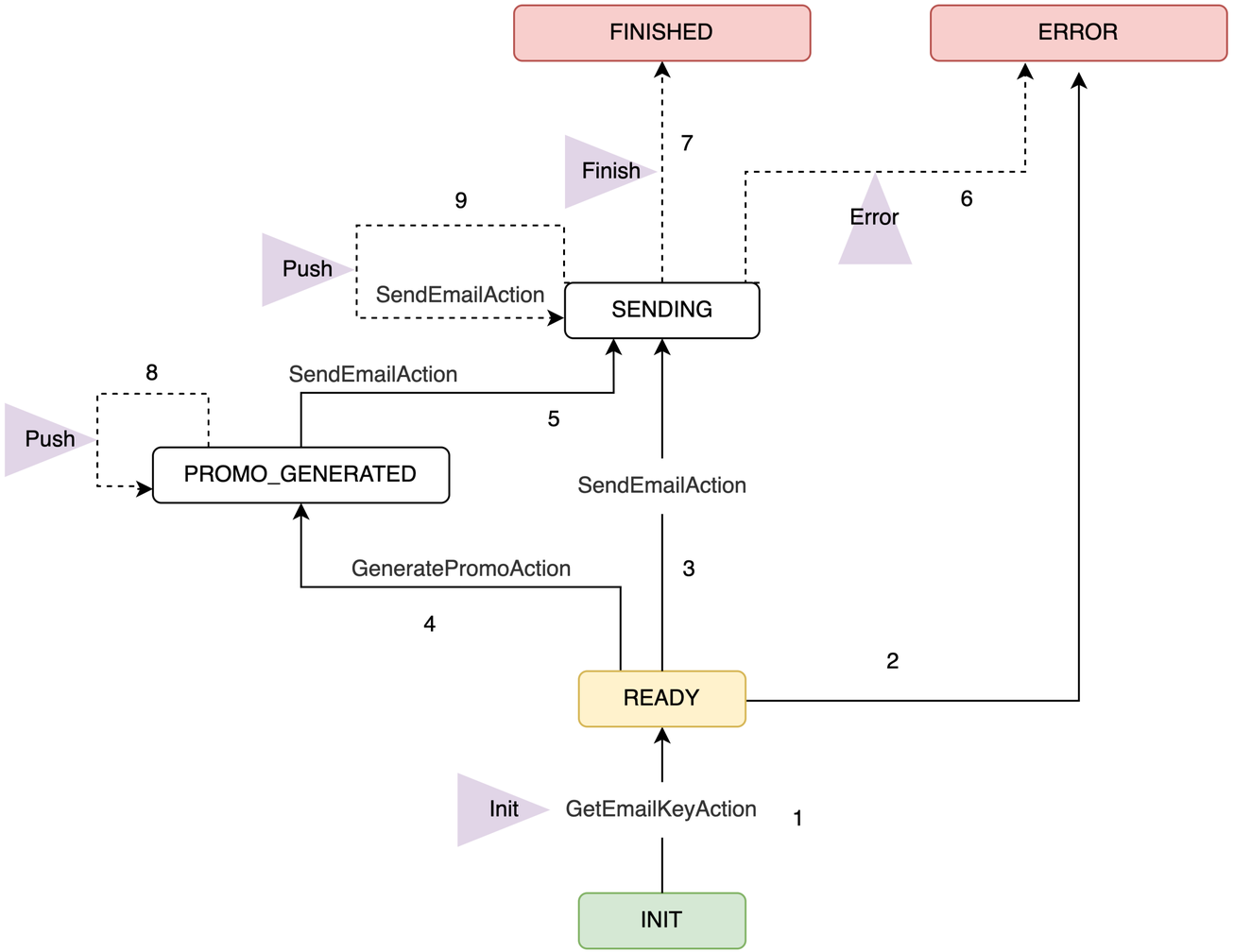
Что такое стейт-машина



https://ru.answacode.com/

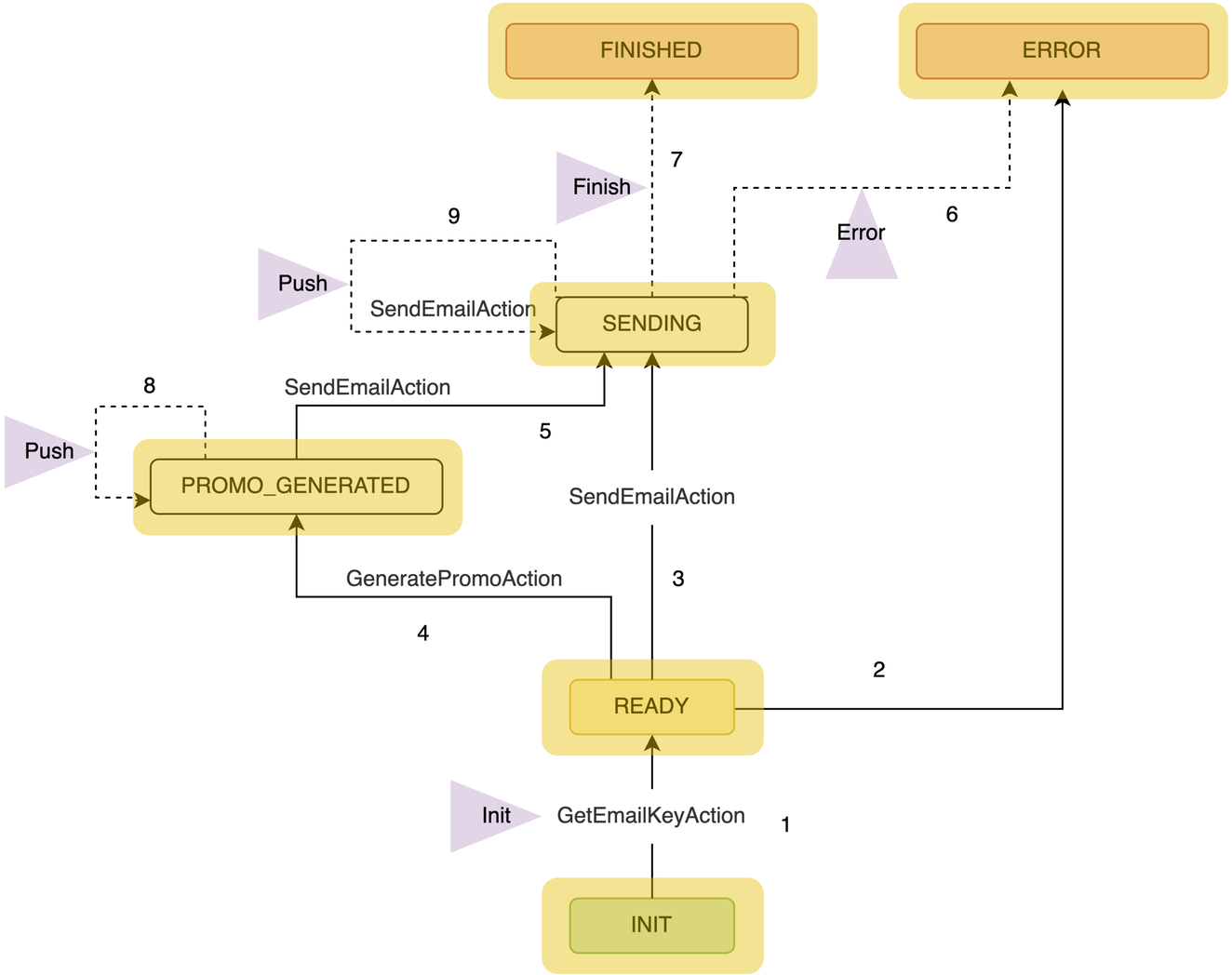
Стейт-машина

«Промокоды для студентов»



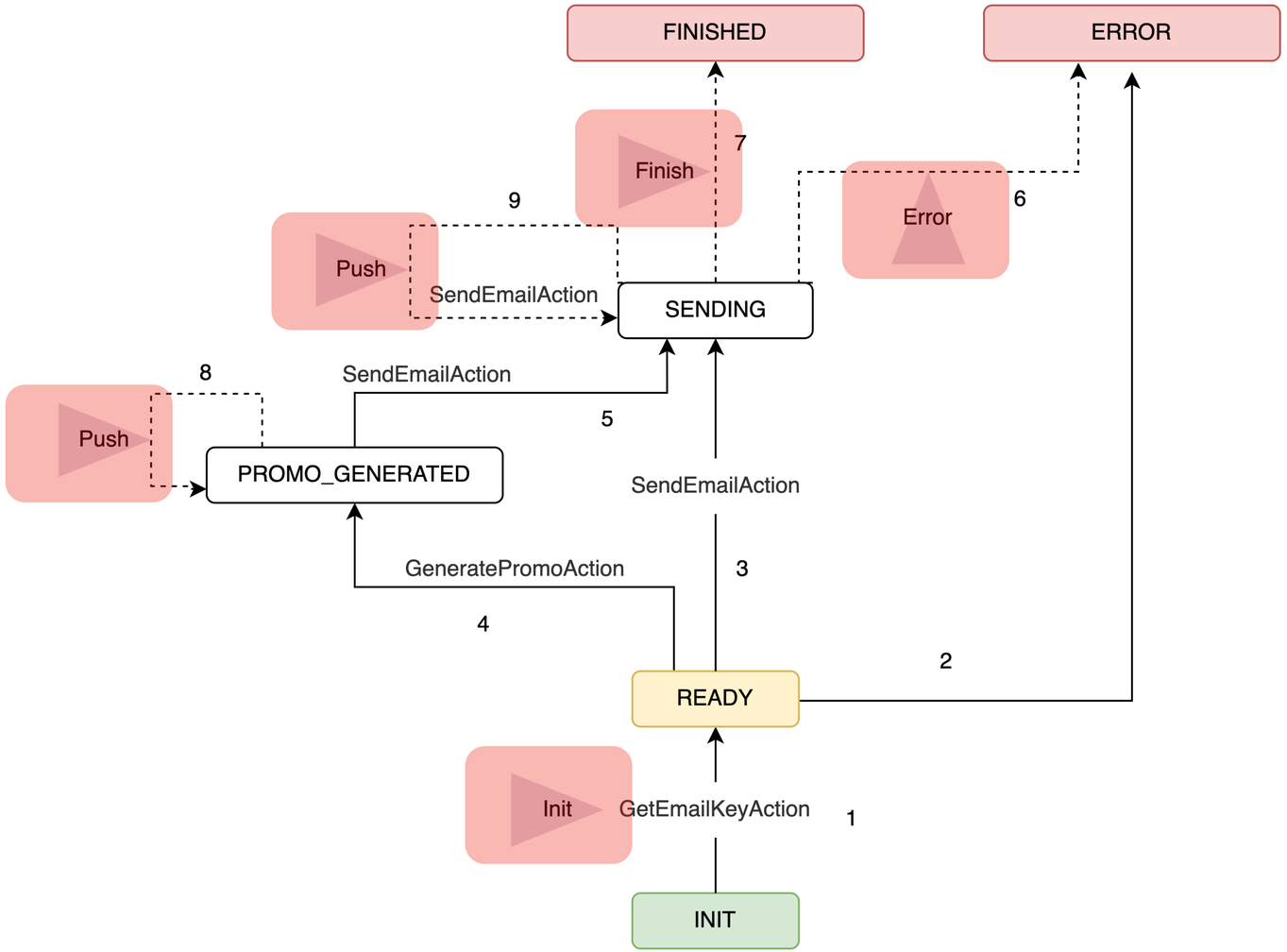
Стейт-машина

«Промокоды для студентов»



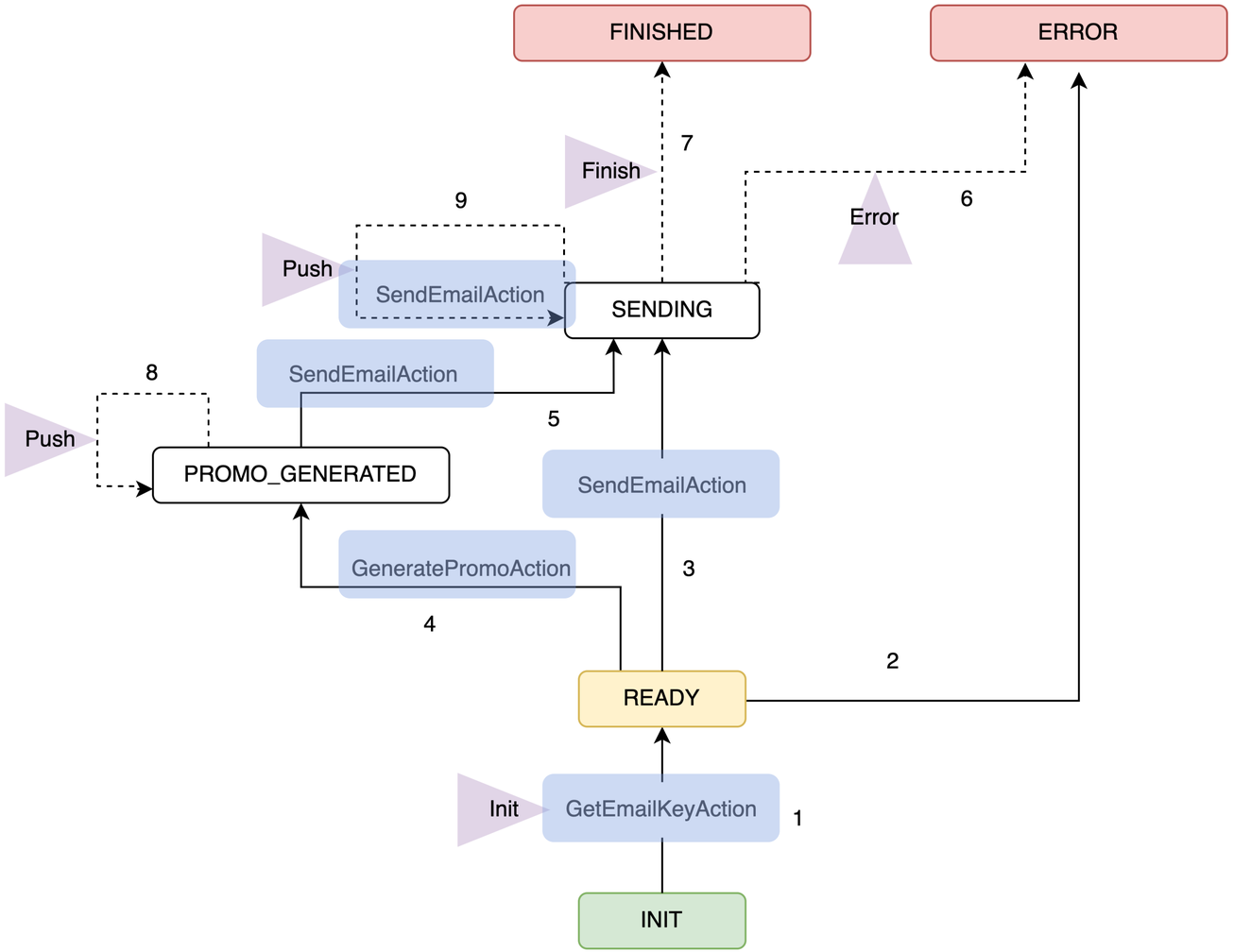
Стейт-машина

«Промокоды для студентов»



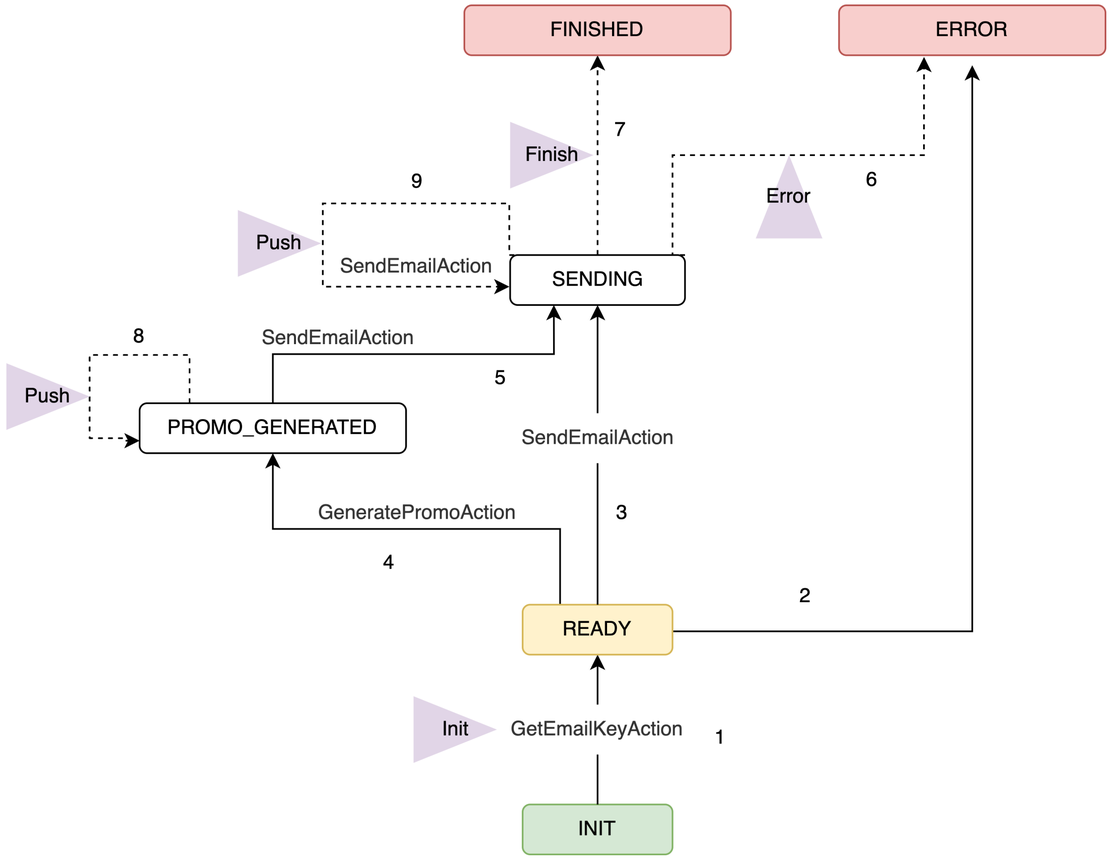
Стейт-машина

«Промокоды для студентов»



Стейт-машина

«Промокоды для студентов»



При этом можно сходу решить обозначенные проблемы:

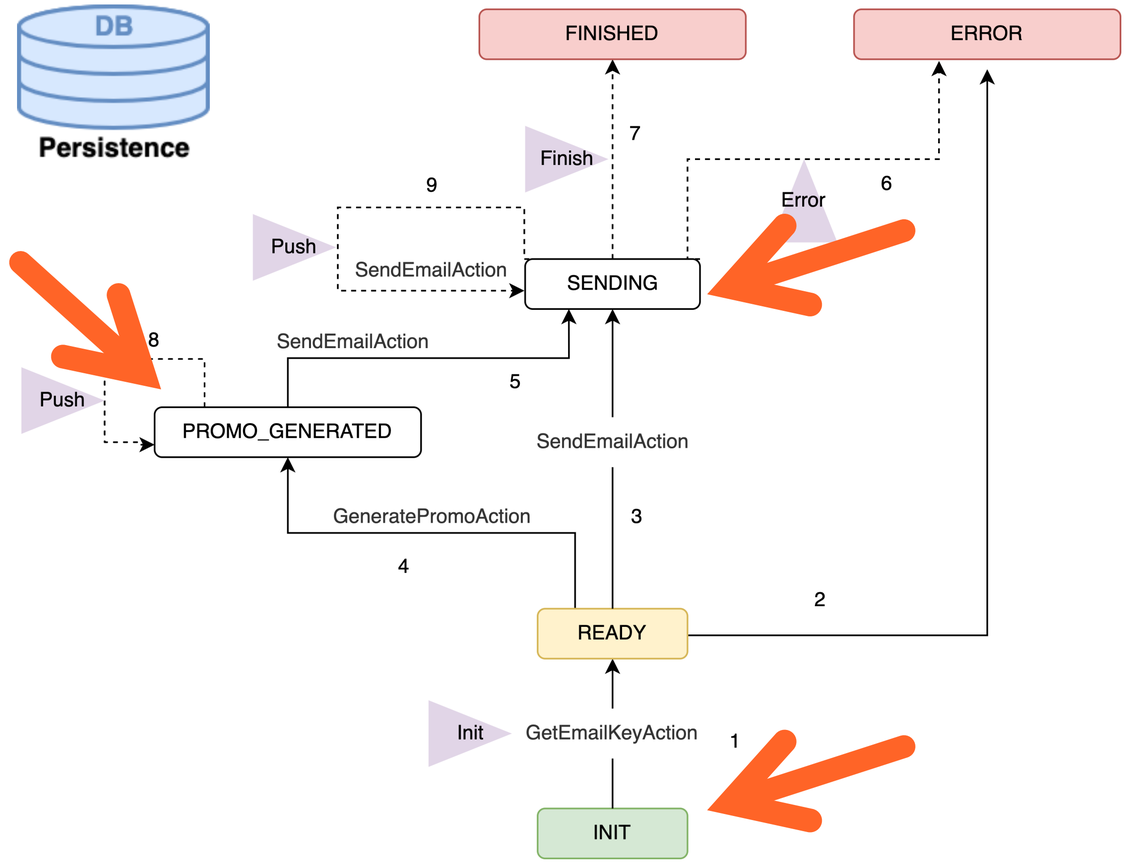
При корректном сохранении состояния стейт машины не составит труда восстановить ее выполнение после падения с консистентного состояния

Есть возможность выполнить Push, если стейт машина зависла в нефинальном состоянии

При условии дедупликации решается проблема с повторной обработкой запросов

Стейт-машина

«Промокоды для студентов»



При этом можно сходу решить обозначенные проблемы:

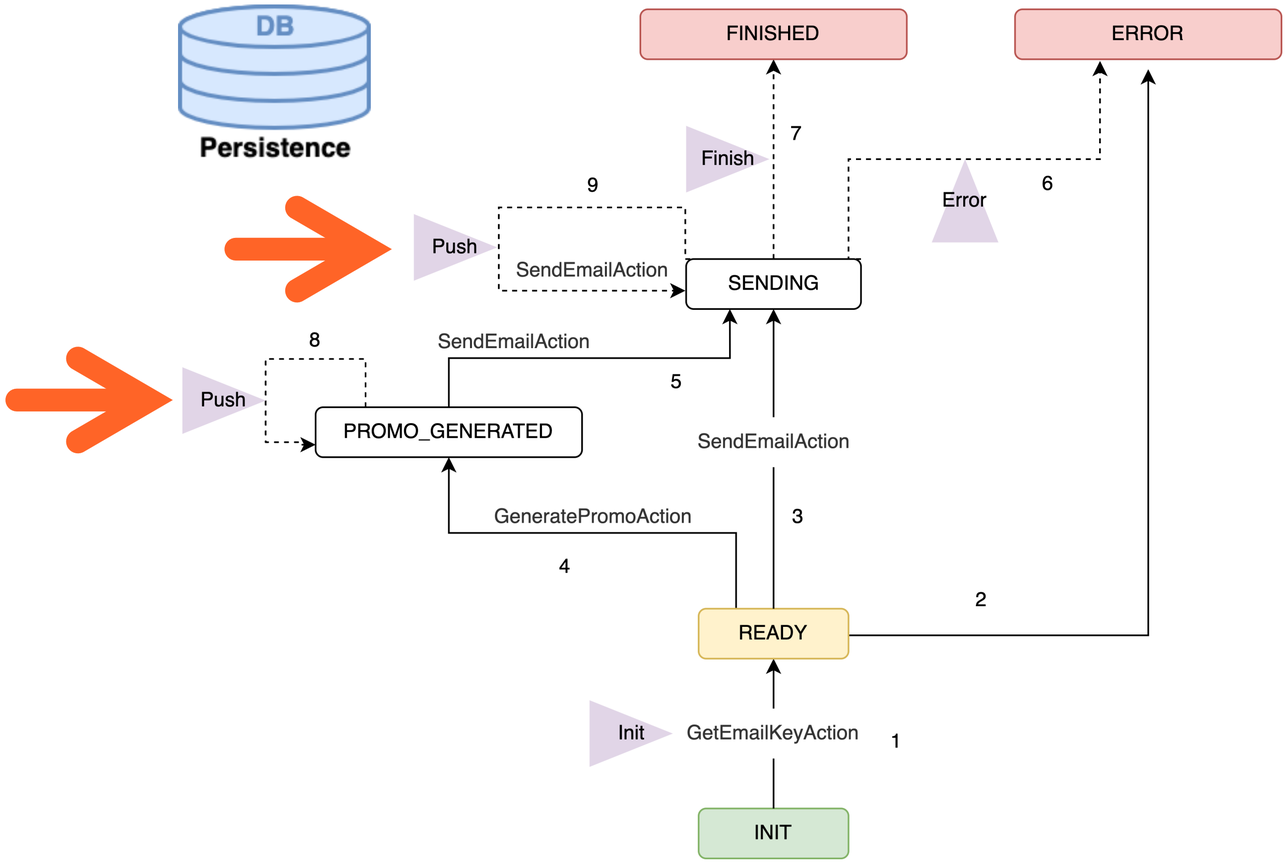
При корректном сохранении состояния стейт машины не составит труда восстановить ее выполнение после падения с консистентного состояния

Есть возможность выполнить Push, если стейт машина зависла в нефинальном состоянии

При условии дедупликации решается проблема с повторной обработкой запросов

Стейт-машина

«Промокоды для студентов»



При этом можно сходу решить обозначенные проблемы:

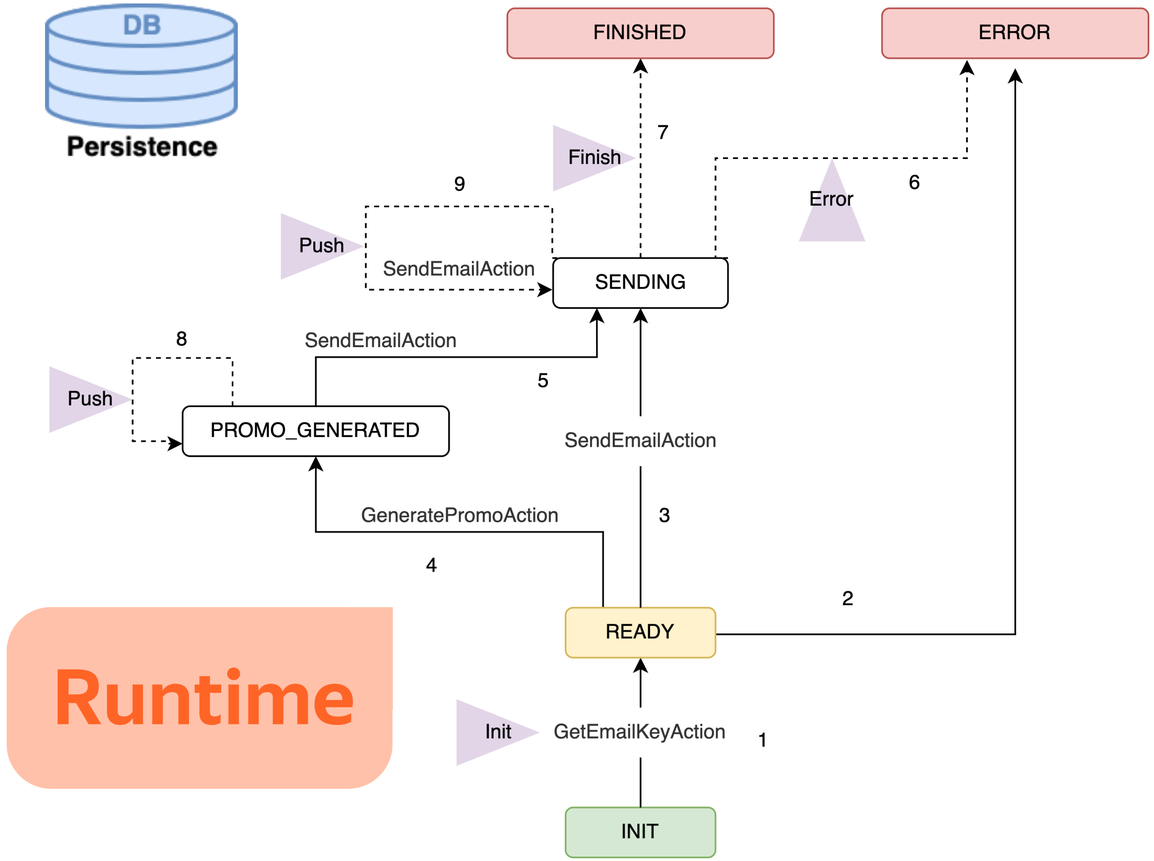
При корректном сохранении состояния стейт машины не составит труда восстановить ее выполнение после падения с консистентного состояния

Есть возможность выполнить Push, если стейт машина зависла в нефинальном состоянии

При условии дедупликации решается проблема с повторной обработкой запросов

Стейт-машина

«Промокоды для студентов»



При этом можно сходу решить обозначенные проблемы:

При корректном сохранении состояния стейт машины не составит труда восстановить ее выполнение после падения с консистентного состояния

Есть возможность выполнить Push, если стейт машина зависла в нефинальном состоянии

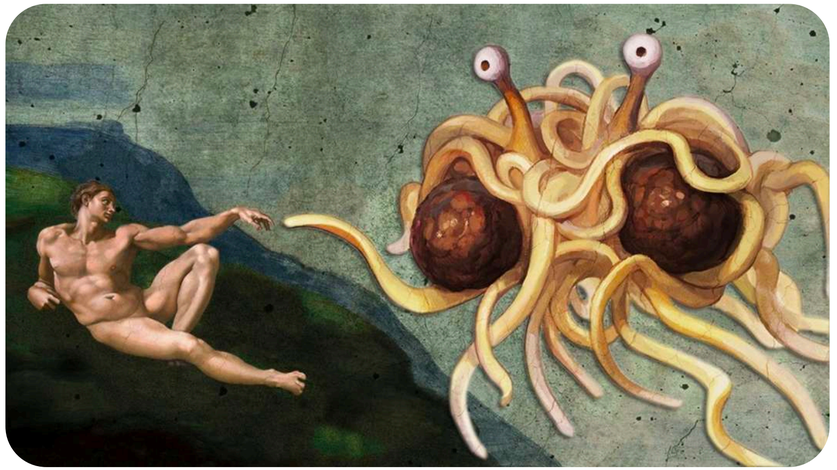
При условии дедупликации решается проблема с повторной обработкой запросов

Выглядит круто.
Как же нам это написать?

Что мы не хотим

```
if (orderResponse.getPaymentStatus().equals("paid")) {
    if (orderResponse.getPaymentOrderId() == null) {
        ExampleUser user = exampleUserManager.findById(orderResponse.getUserId());
        ExampleProduct product = exampleProductManager.findProductByCode(orderResponse.getProductCode());
        initOrder(user, product);
    } else {
        ExampleUser user = exampleUserManager.findById(orderResponse.getUserId());
        onPaidOrder(user, order, orderResponse.getPaymentOrderId());
    }
} else if (orderResponse.getPaymentStatus() == null || orderResponse.getPaymentStatus().equals("new")) {
    ExampleOrder exampleOrder = exampleOrderManager.findById(order.getId());
    if (exampleOrderManager.findById(order.getId()).getStatus().equals("active")) {
        onNotPaidActive(exampleOrder);
    }
} else if (orderResponse.getPaymentStatus().equals("not_paid")) {
    ExampleOrder exampleOrder = exampleOrderManager.findById(order.getId());
    if (exampleOrder.getStatus().equals("active")) {
        cancelOrder(exampleOrder);
    }
} else if (orderInNotUsedState(orderResponse) && exampleFeatureFlags.isGracePeriodEnabled()) {
    onNotUsed(orderResponse);
} else if (orderResponse.getPaymentStatus().equals("refunded")) {
    ExampleOrder exampleOrder = exampleOrderManager.findById(order.getId());
    cancelOrder(exampleOrder);
} else {
    throw new IllegalStateException("Unable to process response " + orderResponse);
}
```

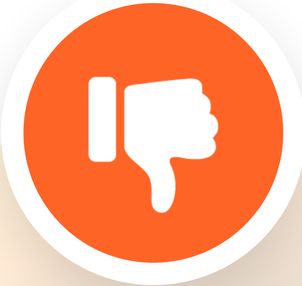
Неструктурированный код на if-ах



https://dzen.ru

Что мы не хотим

```
private void performAction(AnalyzedSubscription analyzedSubscription) {
    switch (analyzedSubscription.state) {
        case INIT -> onInit(analyzedSubscription);
        case PAID -> onPaid(analyzedSubscription);
        case NOT_PAID_ACTIVE -> onNotPaidActive(analyzedSubscription);
        case STOPPED -> onStopped(analyzedSubscription);
        case NOT_USED -> onNotUsed(analyzedSubscription);
        case REFUNDED -> onRefunded(analyzedSubscription);
        case ERROR -> onError(analyzedSubscription);
        default -> throw new NoSuchElementException("Unexpected state: "
    }
}
```



Простой switch

https://ru.pinterest.com

Что мы не хотим

Мастер маскировки

https://funnyart.club

```
private final MapF<Tuple2<ExampleReceiptState, ExampleServiceState>,
        Function4<ExampleReceiptState, ExampleServiceState, ExampleProduct, ExampleUser,
                Option<ExampleOrder>>> handleDecisions =
        Cf.toHashMap(Cf.list(
                Tuple2.tuple(Tuple2.tuple(ExampleReceiptState.NOT_FOUND, ExampleServiceState.NOT_FOUND),
                        this::doNothing),
                Tuple2.tuple(Tuple2.tuple(ExampleReceiptState.NOT_FOUND, ExampleServiceState.BOUGHT_WITH_FIRST_APP),
                        this::errorWrongApplication),
                Tuple2.tuple(Tuple2.tuple(ExampleReceiptState.NOT_FOUND, ExampleServiceState.BOUGHT_WITH_SECOND_APP),
                        this::errorOtherStore),
```

Требования к движку стейт-машин

01

Удобство применения.

От разработчика требуется только задать граф состояний и переходов и реализовать бизнес-логику экшенов.

02

Легкость дебага.

Автоматическое логирование переходов и читабельный код.

03

Отказоустойчивость.

При файле системы стейт-машина может восстановиться из сохраненного контекста.

04

Дедупликация.

Не может выполняться две одинаковые стейт машины.

https://www.drive2.ru

Требования к движку стейт-машин

01

Удобство применения.

От разработчика требуется только задать граф состояний и переходов и реализовать бизнес-логику экшенов.

02

Легкость дебага.

Автоматическое логирование переходов и читабельный код.

03

Отказоустойчивость.

При файле системы стейт-машина может восстановиться из сохраненного контекста.

04

Дедупликация.

Не может выполняться две одинаковые стейт машины.

https://ru.pinterest.com

Требования к движку стейт-машин

01

Удобство применения.

От разработчика требуется только задать граф состояний и переходов и реализовать бизнес-логику экшенов.

02

Легкость дебага.

Автоматическое логирование переходов и читабельный код.

03

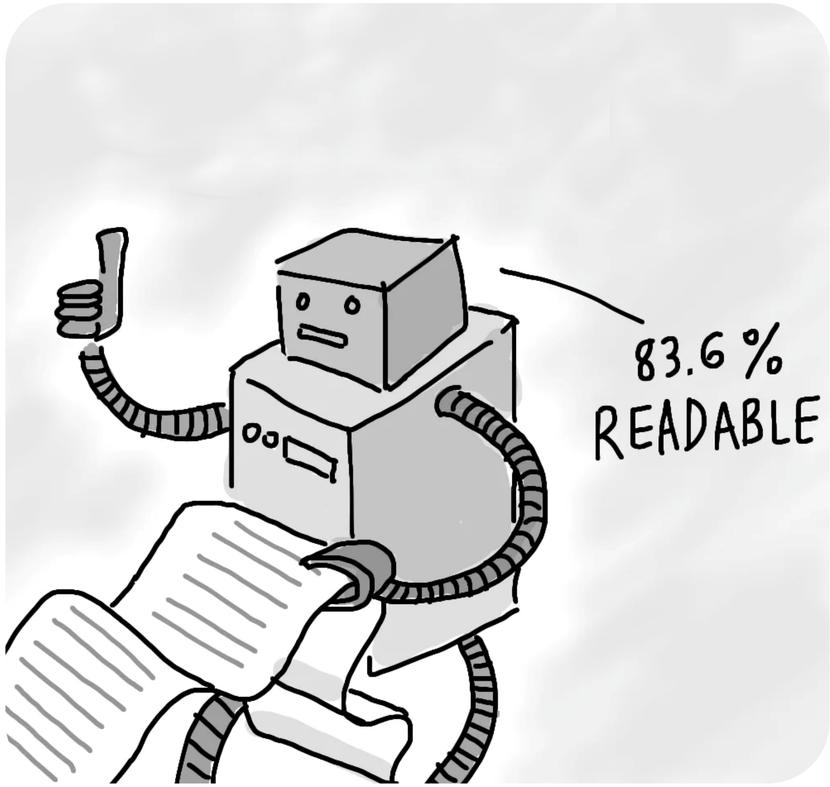
Отказоустойчивость.

При файле системы стейт-машина может восстановиться из сохраненного контекста.

04

Дедупликация.

Не может выполняться две одинаковые стейт машины.



https://frettboard.com

Требования к движку стейт-машин

01

Удобство применения.

От разработчика требуется только задать граф состояний и переходов и реализовать бизнес-логику экшенов.

02

Легкость дебага.

Автоматическое логирование переходов и читабельный код.

03

Отказоустойчивость.

При фейле системы стейт-машина может восстановиться из сохраненного контекста.

04

Дедупликация.

Не может выполняться две одинаковые стейт машины.

https://www.imdb.com

Требования к движку стейт-машин

01

Удобство применения.

От разработчика требуется только задать граф состояний и переходов и реализовать бизнес-логику экшенов.

02

Легкость дебага.

Автоматическое логирование переходов и читабельный код.

03

Отказоустойчивость.

При фейле системы стейт-машина может восстановиться из сохраненного контекста.

04

Дедупликация.

Не может выполняться две одинаковые стейт машины.

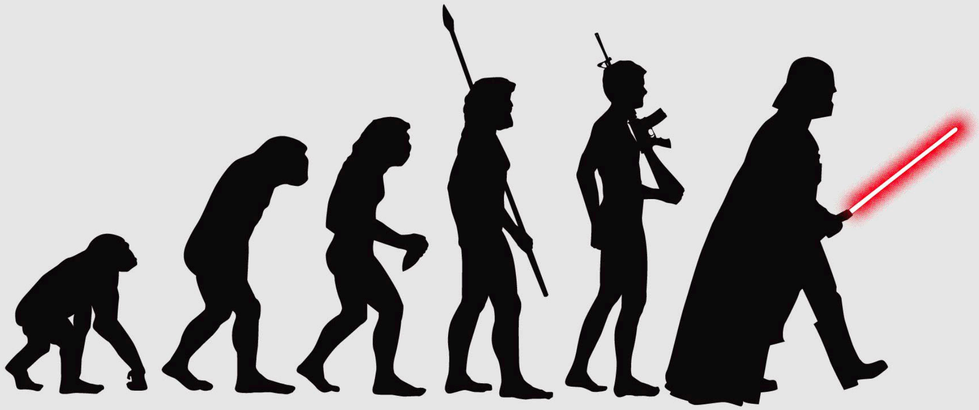
https://dota2.ru/

Требования к движку стейт-машин

05 **Расширяемость.**
Легко совершить изменение флоу (и выкатить изменение, не поломав старые стейт-машины).

06 **Переиспользуемость.**
С помощью движка легко написать новую стейт-машину.

07 **Удобство покрытия тестами.**



https://wallpaper.mob.org

Требования к движку стейт-машин

05 **Расширяемость.**
Легко совершить изменение флоу (и выкатить изменение, не поломав старые стейт-машины).

06 **Переиспользуемость.**
С помощью движка легко написать новую стейт-машину.

07 **Удобство покрытия тестами.**

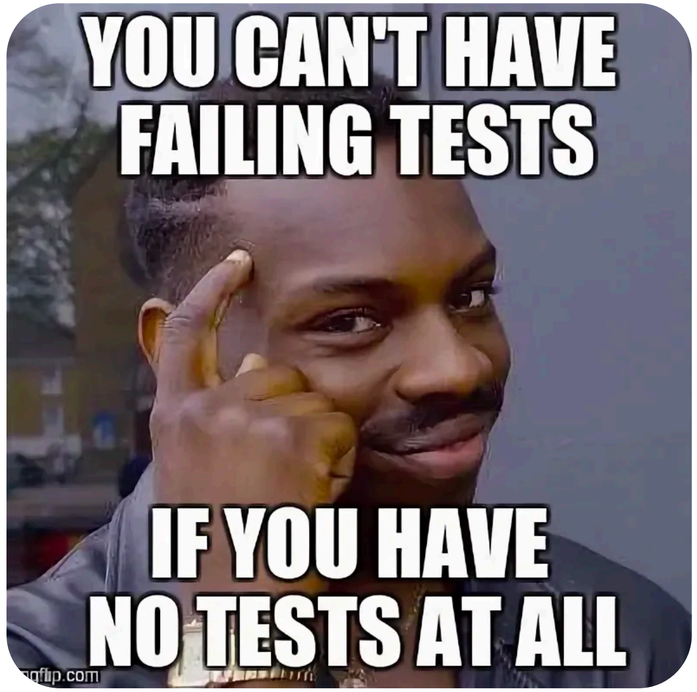
https://ru.pinterest.com

Требования к движку стейт-машин

05 **Расширяемость.**
Легко совершить изменение флоу (и выкатить изменение, не поломав старые стейт-машины).

06 **Переиспользуемость.**
С помощью движка легко написать новую стейт-машину.

07 **Удобство покрытия тестами.**



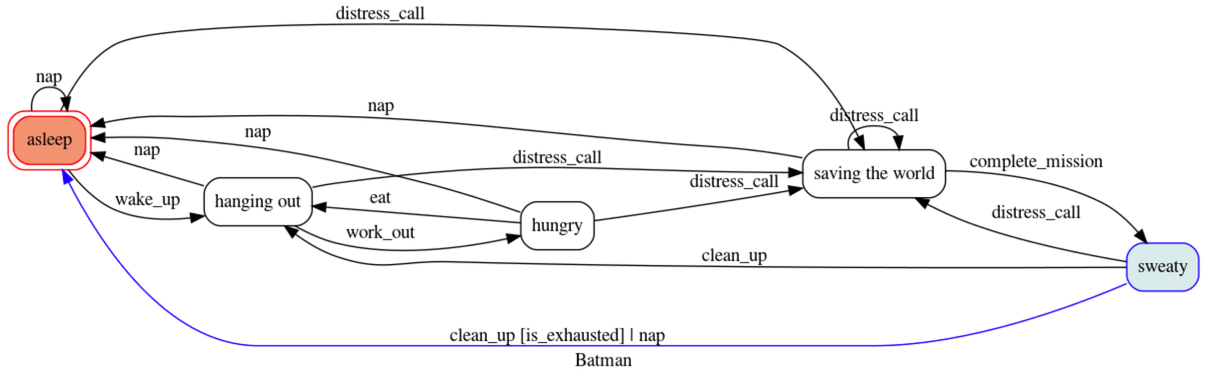
https://tartukhkjena19.blogspot.com

Какие есть варианты?

Какие есть варианты?

pytransitions

- Есть возможности визуализации
- Красивое апи
- Не подходит под наш стек



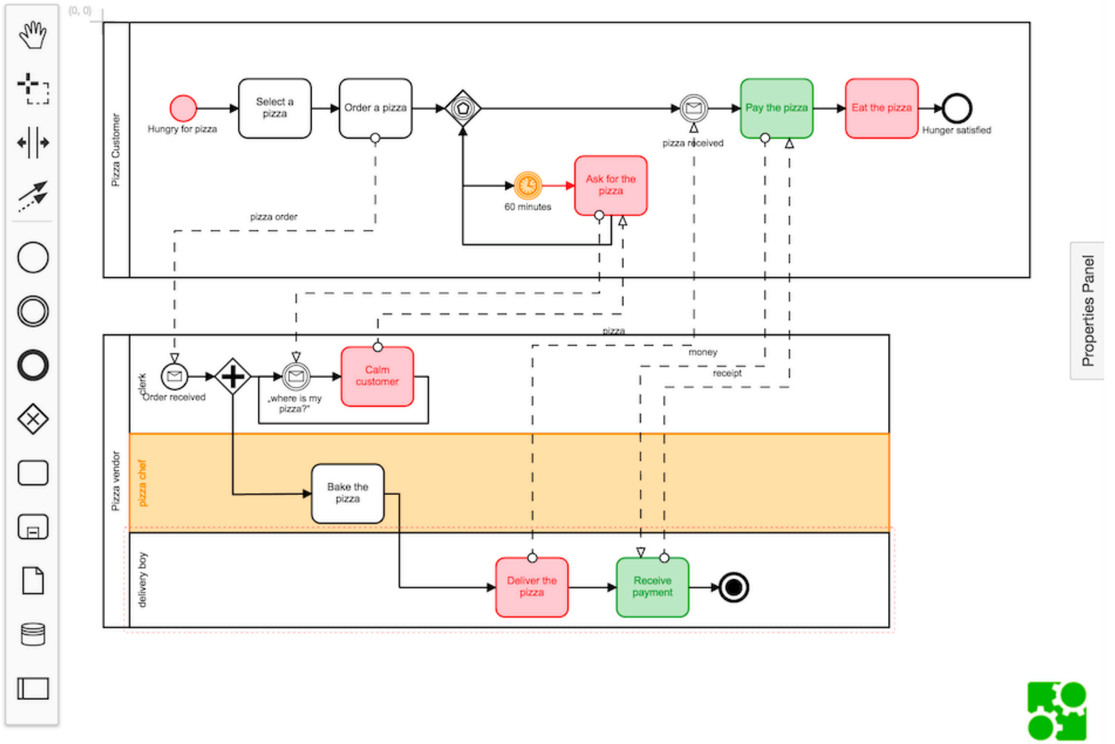
Какие есть варианты?

pytransitions

- Есть возможности визуализации
- Красивое апи
- Не подходит под наш стек

camundarus

- Фреймворк
- Возможность конфигурации процесса
- Есть возможности визуализации
- Тяжело интегрировать в проект
- Если копать глубже, подходит для описания BPM, но не стейт-машины



Какие есть варианты?

pytransitions

- Есть возможности визуализации
- Красивое апи
- Не подходит под наш стек

camundarus

- Фреймворк
- Возможность конфигурации процесса
- Есть возможности визуализации
- Тяжело интегрировать в проект
- Если копать глубже, подходит для описания BPM, но не стейт-машины

spring-statemachine

- Либа - легко затащить
- Совместимо с нашим стеком
- Много возможностей конфигурации
- Красивое апи
- Удобное покрытие тестами
- Сложна для понимания
- Много неочевидных моментов - нужно штудировать документацию

Какие есть варианты?

pytransitions

- Есть возможности визуализации
- Красивое апи
- Не подходит под наш стек

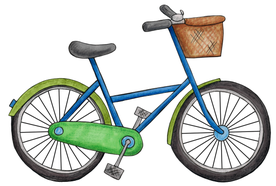
camundarus

- Фреймворк
- Возможность конфигурации процесса
- Есть возможности визуализации
- Тяжело интегрировать в проект
- Если копать глубже, подходит для описания BPM, но не стейт-машины

spring-statemachine

- Либа - легко затащить
- Совместимо с нашим стеком
- Много возможностей конфигурации
- Красивое апи
- Удобное покрытие тестами
- Сложна для понимания
- Много неочевидных моментов - нужно штудировать документацию

https://toppng.com



Написать свой движок

- Решим под свою задачу
- Отсутствие лишнего функционала
- Не нужно добавлять зависимости
- Потенциально дорого
- Может не поддержать кейсы других стейт-машин (не переиспользуема)

Какие есть варианты?

python

pytransitions

- Есть возможности визуализации
- Красивое апи
- Не подходит под наш стек

camunda

camundarus

- Фреймворк
- Возможность конфигурации процесса
- Есть возможности визуализации
- Тяжело интегрировать в проект
- Если копать глубже, подходит для описания BPM, но не стейт-машины

spring

spring-statemachine

- Либа - легко затащить
- Совместимо с нашим стеком
- Много возможностей конфигурации
- Красивое апи
- Удобное покрытие тестами
- Сложна для понимания
- Много неочевидных моментов - нужно штудировать документацию

https://toppng.com

Написать свой движок

- Решим под свою задачу
- Отсутствие лишнего функционала
- Не нужно добавлять зависимости
- Потенциально дорого
- Может не поддержать кейсы других стейт-машин (не переиспользуема)

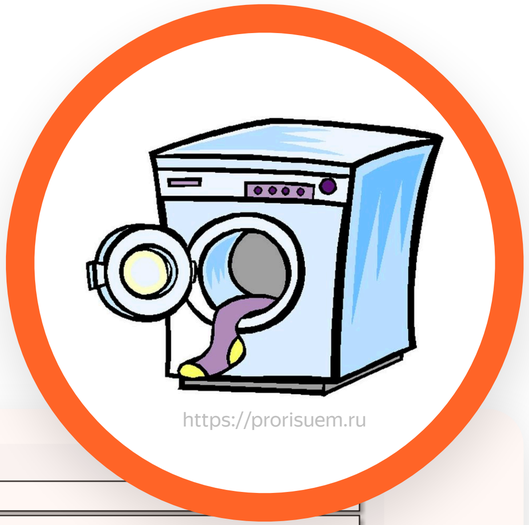
Про spring-state-machine

```
public enum States {
    RUNNING, HISTORY, END,
    WASHING, RINSING, DRYING,
    POWEROFF
}
```

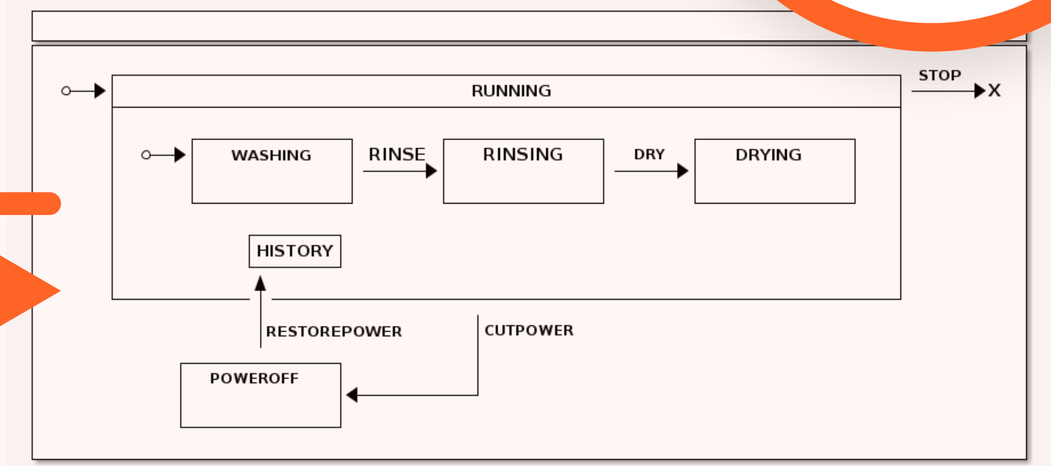
```
public enum Events {
    RINSE, DRY, STOP,
    RESTOREPOWER, CUTPOWER
}
```

States

Events




```
transitions
    .withExternal()
        .source(States.WASHING).target(States.RINSING)
        .event(Events.RINSE)
        .and()
    .withExternal()
        .source(States.RINSING).target(States.DRYING)
        .event(Events.DRY)
        .and()
```



spring-statemachine

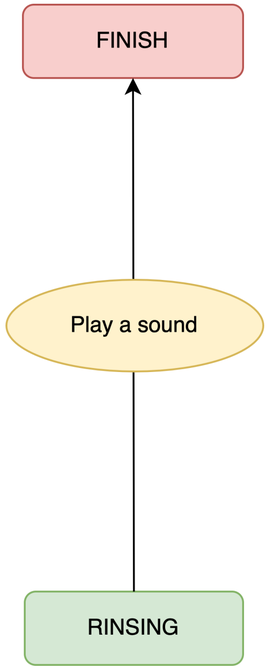
States
Events

Actions
Guards

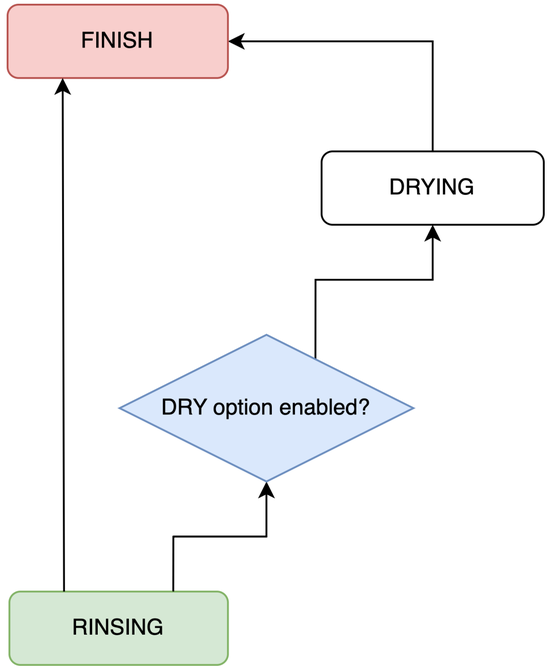
Context

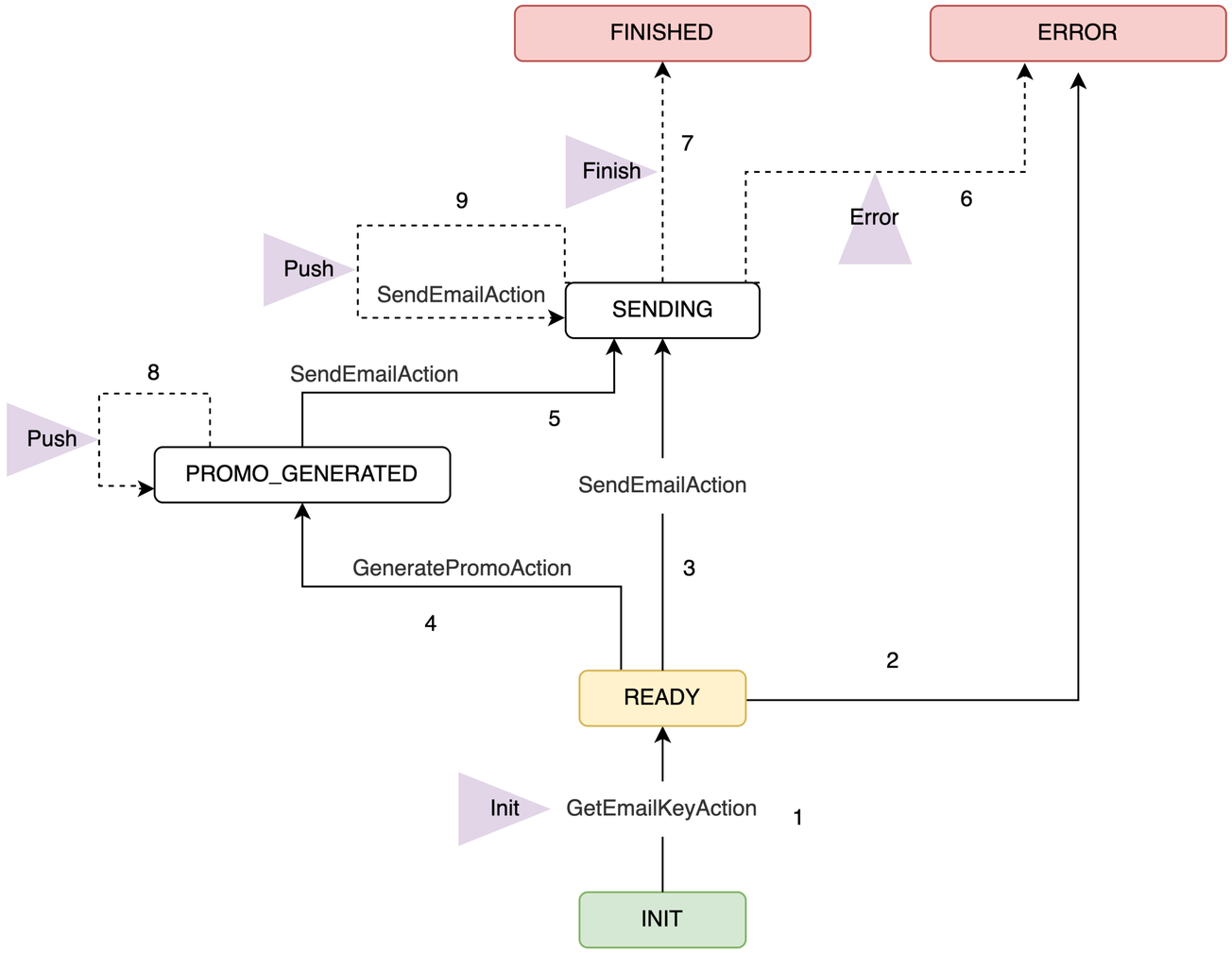
https://prorisuem.ru

Action



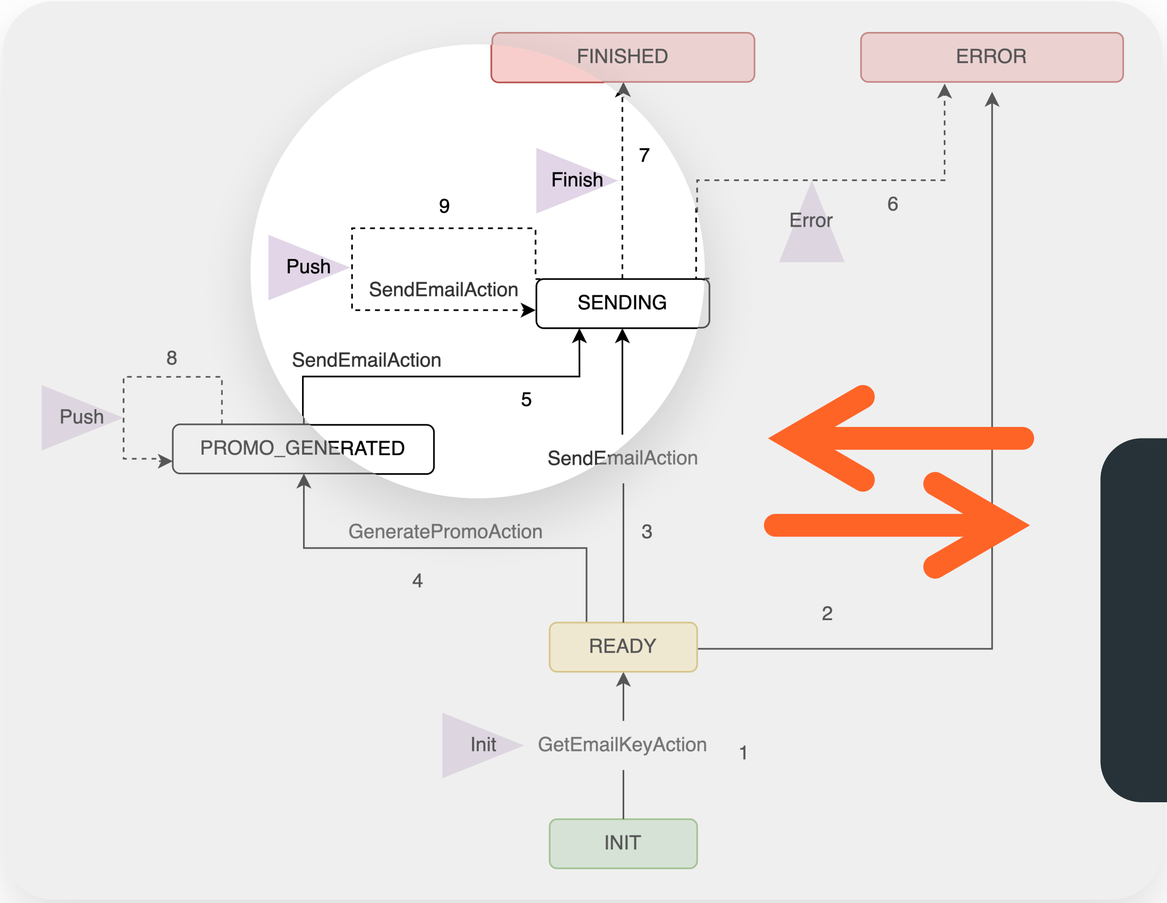
Guard




FINISHED
ERROR
7
Finish
9
Push
SendEmailAction
SENDING
6
Error
8
SendEmailAction
Push
5
PROMO_GENERATED
SendEmailAction
GeneratePromoAction
3
4
2
READY
Init
GetEmailKeyAction
1
INIT

Стейт-машина

«Промокоды для студентов»



Переходы между состояниями задаются конфигурацией

```
.and()
.withExternal()
.source(PromoCodeRequestStatus.SENDING)
.target(PromoCodeRequestStatus.SENDING)
.event(PromoCodeRequestEvent.PUSH)
.guard(resendGuard(bazingaTaskManager))
.action(sendEmailAction(taskScheduler, studentPromoCodeRequestDao))
```

Стейт-машина

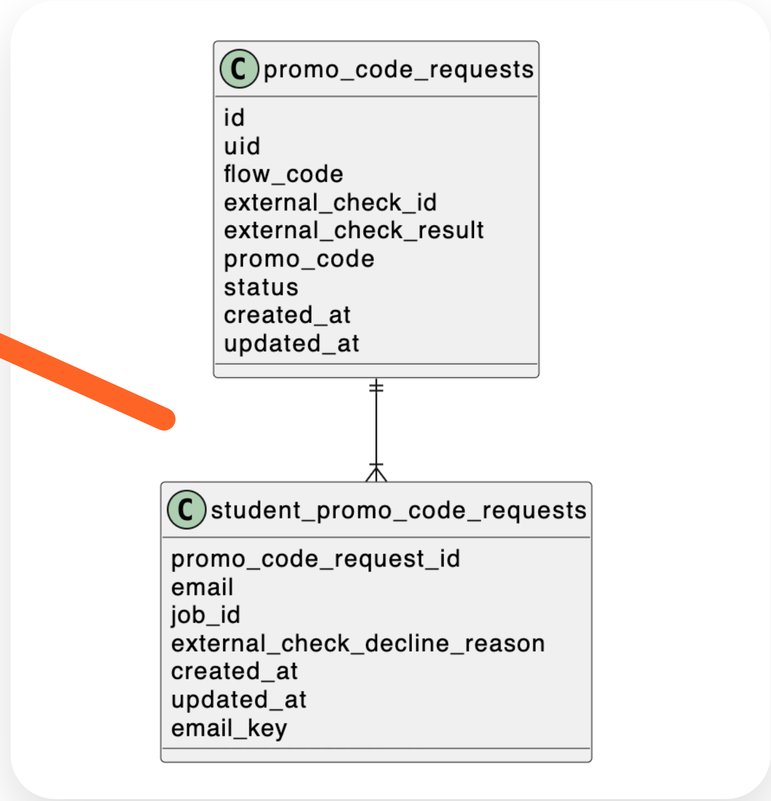
«Промокоды для студентов»

Бизнес-логика принимает **promoCodeRequest** в качестве контекста

```java
@Override
public void execute(PromoCodeRequestEntity promoCodeRequest,
                    StudentPromoCodeRequestEntity studentPromoCodeRequest) {
    MailContext mailContext = MailContext.builder()
            .to(promoCodeRequest.getUid())
            .email(Email.parseSafe(studentPromoCodeRequest.getEmail()))
            .language(Option.of(Language.RUSSIAN))
            .promoCodeRequestId(Option.of(promoCodeRequest.getId()))
            .build();

    Option<FullJobId> jobId = taskScheduler.scheduleLocalizedEmailTask(
            studentPromoCodeRequest.getEmailKey()
                    .orElseThrow(() -> new NoSuchElementException("Email key not found")).value(),
            mailContext, Option.empty(), StudentsLocalizedEmailTaskExecutor.KEY);

    log.debug("Promo code email for request {} was scheduled, jobId {}", promoCodeRequest.getId(), jobId);
}
```



Сервис

```
@AllArgsConstructor
public class AbstractStateMachineService<S, E, ID> {

    private StateMachineFactory<S, E> stateMachineFactory;

    private StateMachinePersister<S, E, ID> persister;


    public void handleEvent(ID stateMachineId, E event) throws Exception {
        StateMachine<S, E> stateMachine = stateMachineFactory.getStateMachine();
        persister.restore(stateMachine, stateMachineId);

        stateMachine.start();
        stateMachine.sendEvent(event);

        persister.persist(stateMachine, stateMachineId);
    }
```

1

2

3

Сервис

```
@AllArgsConstructor
public class AbstractStateMachineService<S, E, ID> {

    private StateMachineFactory<S, E> stateMachineFactory;

    private StateMachinePersister<S, E, ID> persister;


    public void handleEvent(ID stateMachineId, E event) throws Exception {
        StateMachine<S, E> stateMachine = stateMachineFactory.getStateMachine();
        persister.restore(stateMachine, stateMachineId);

        stateMachine.start();
        stateMachine.sendEvent(event);

        persister.persist(stateMachine, stateMachineId);
    }
```

1

2

3

Сервис

```
@AllArgsConstructor
public class AbstractStateMachineService<S, E, ID> {

    private StateMachineFactory<S, E> stateMachineFactory;

    private StateMachinePersister<S, E, ID> persister;


    public void handleEvent(ID stateMachineId, E event) throws Exception {
        StateMachine<S, E> stateMachine = stateMachineFactory.getStateMachine();
        persister.restore(stateMachine, stateMachineId);

        stateMachine.start();
        stateMachine.sendEvent(event);

        persister.persist(stateMachine, stateMachineId);
    }
```

1

2

3

Сервис

```java
@AllArgsConstructor
public class AbstractStateMachineService<S, E, ID> {

    private StateMachineFactory<S, E> stateMachineFactory;

    private StateMachinePersister<S, E, ID> persister;


    public void handleEvent(ID stateMachineId, E event) throws Exception {
        StateMachine<S, E> stateMachine = stateMachineFactory.getStateMachine();
        persister.restore(stateMachine, stateMachineId);

        stateMachine.start();
        stateMachine.sendEvent(event);

        persister.persist(stateMachine, stateMachineId);
    }
```

1

2

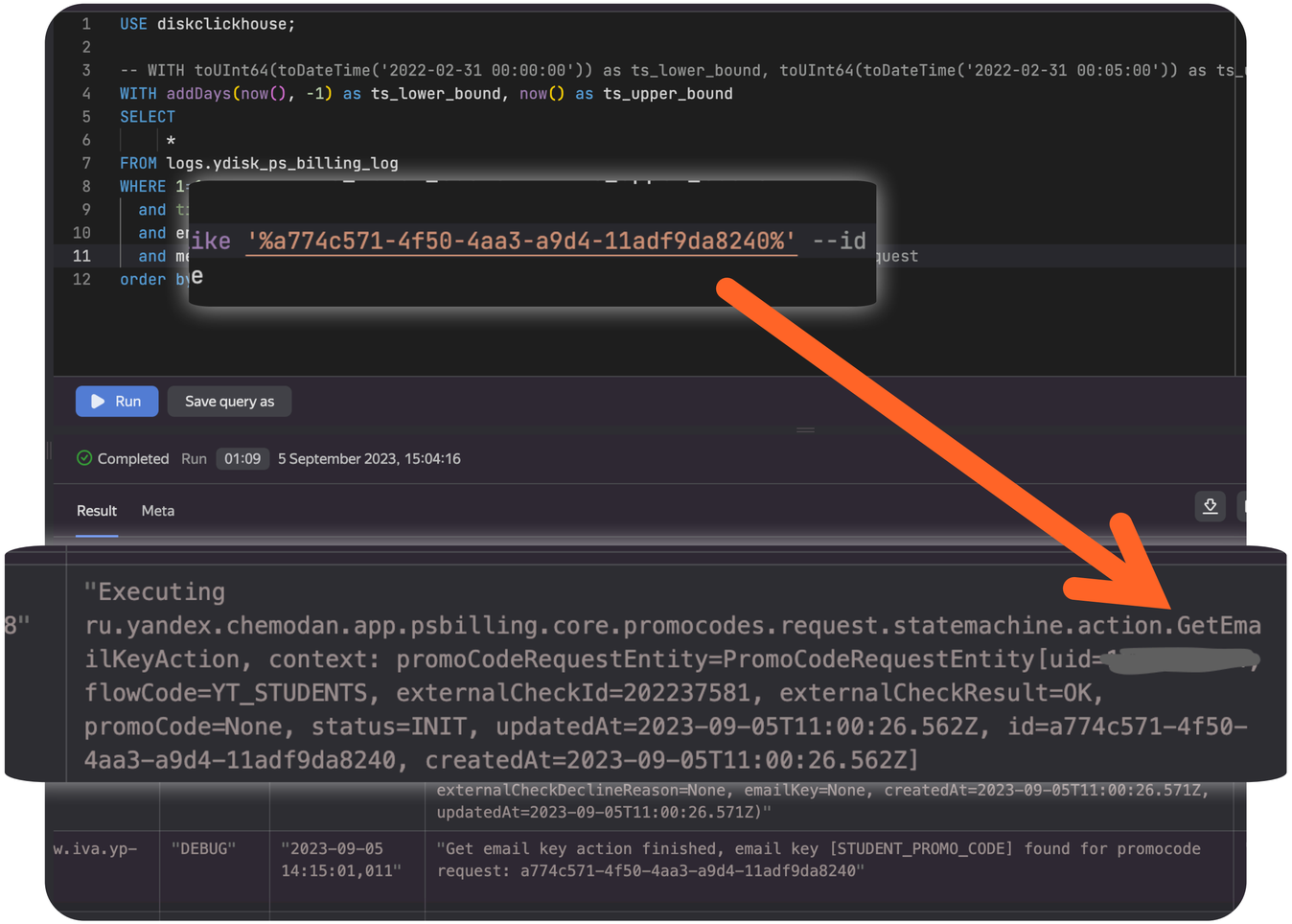
3

Transactional, with locked promoCodeRequest

Команда

Бизнес

Как дебажить стейт-машину



01 Выгрузить логи по id promo_code_request

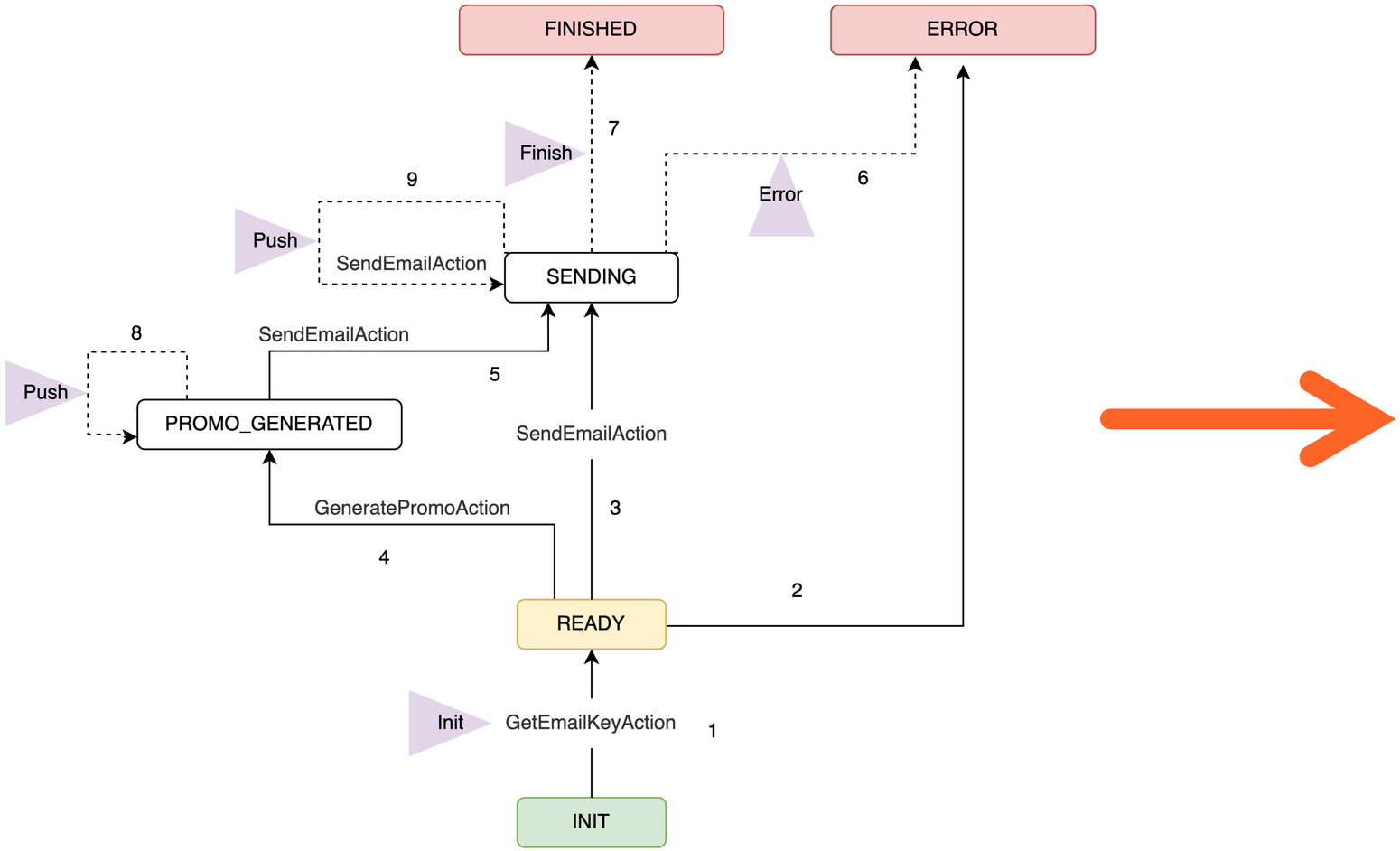
02 Видим логи каждого перехода и каждого экшена

03 Видим контекст выполнения

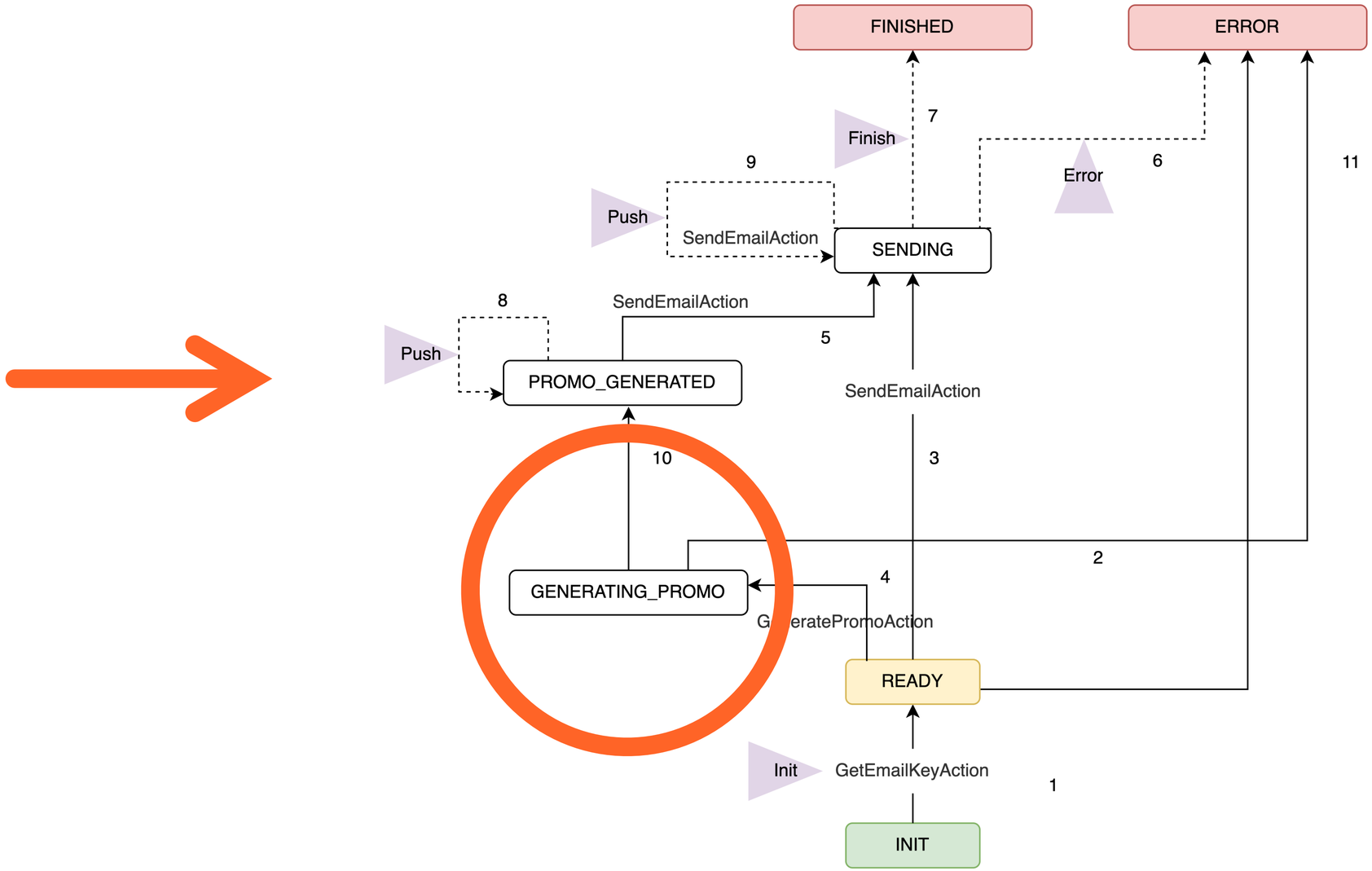
04 Видим логи экшенов

Время дебага = время на 1 выгрузку

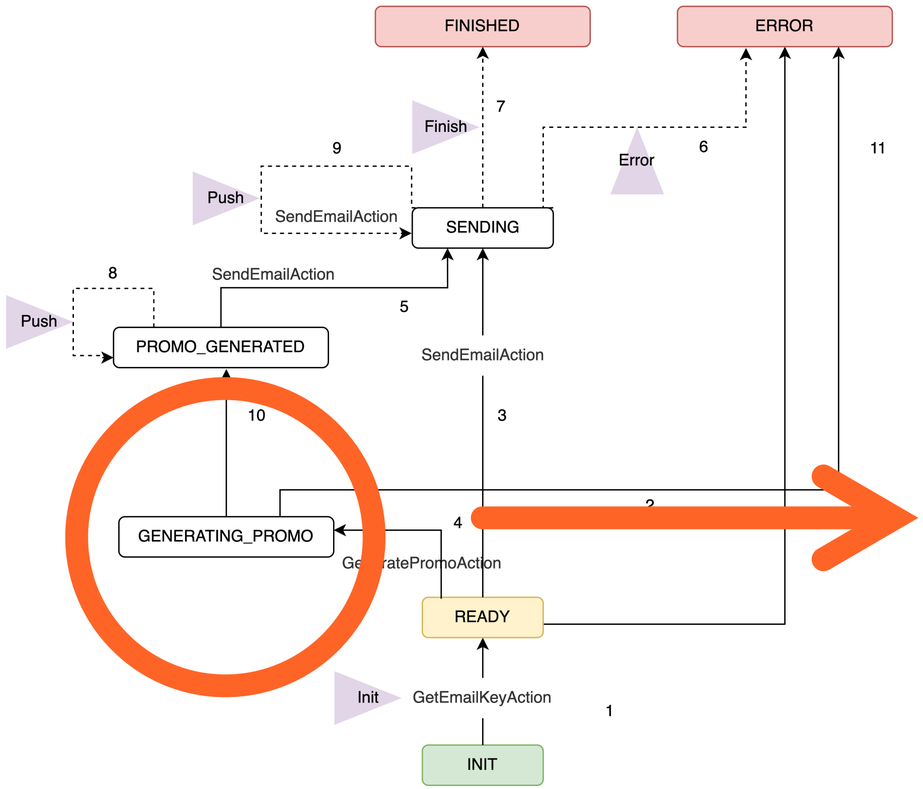
Как внести изменение



Как внести изменение



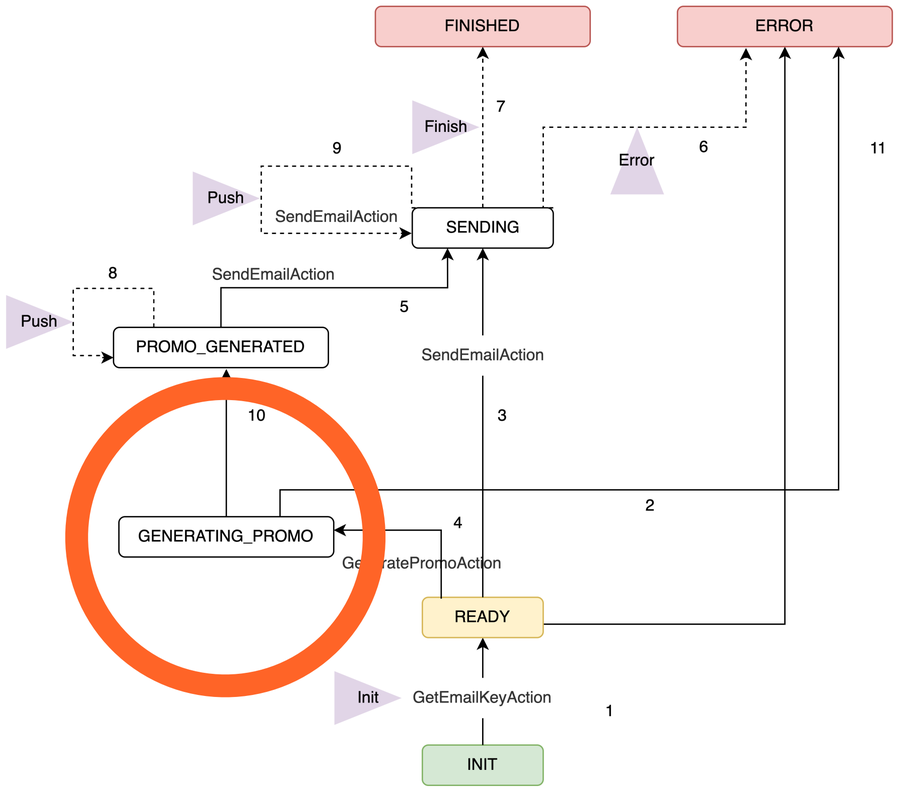
Как внести изменение



01 Добавить переход

```
.and()
.withChoice()
.source(PromoCodeRequestStatus.GENERATING_PROMO)
.then(PromoCodeRequestStatus.PROMO_GENERATED, promoCodeGeneratedGuard(promoCodeRequestDao))
.last(PromoCodeRequestStatus.ERROR)
```

Как внести изменение



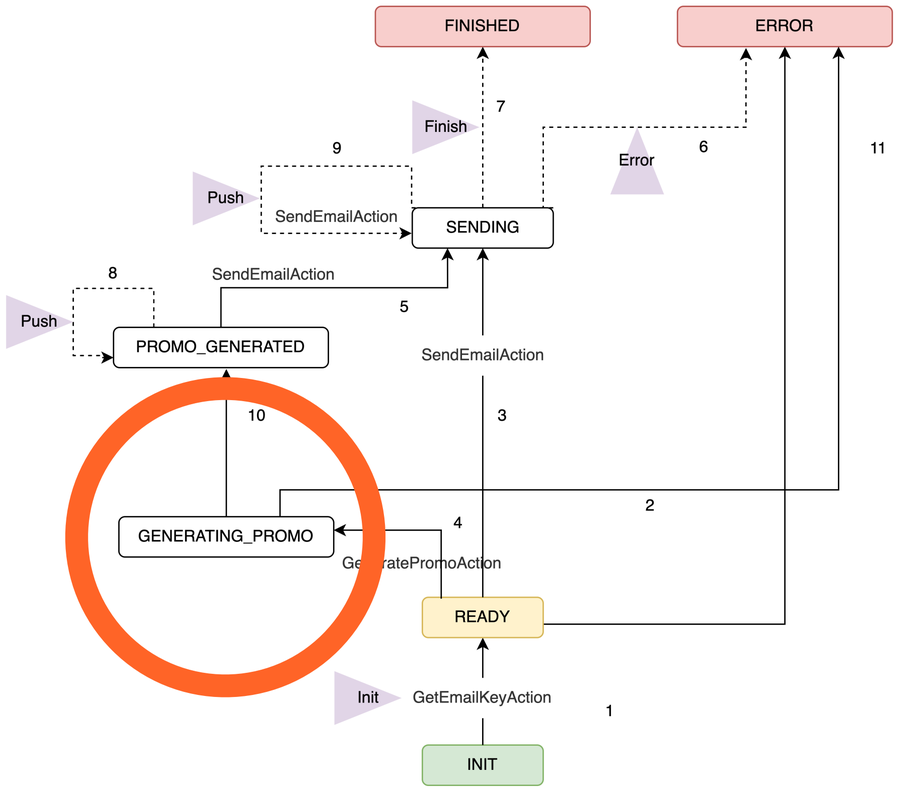
01 Добавить переход

02 Реализовать новые экшены и гарды

```
.and()
.withChoice()
.source(PromoCodeRequestStatus.GENERATING_PROMO)
.then(PromoCodeRequestStatus.PROMO_GENERATED, promoCodeGeneratedGuard(promoCodeRequestDao))
.last(PromoCodeRequestStatus.ERROR)
```

```
2 usages
private Guard<PromoCodeRequestStatus, PromoCodeRequestEvent> promoCodeGeneratedGuard(
        PromoCodeRequestDao promoCodeRequestDao) {
    return new PromoCodeGeneratedGuard(promoCodeRequestContextFactory, promoCodeRequestDao);
}
```

Как внести изменение



01 Добавить переход

02 Реализовать новые экшены и гарды

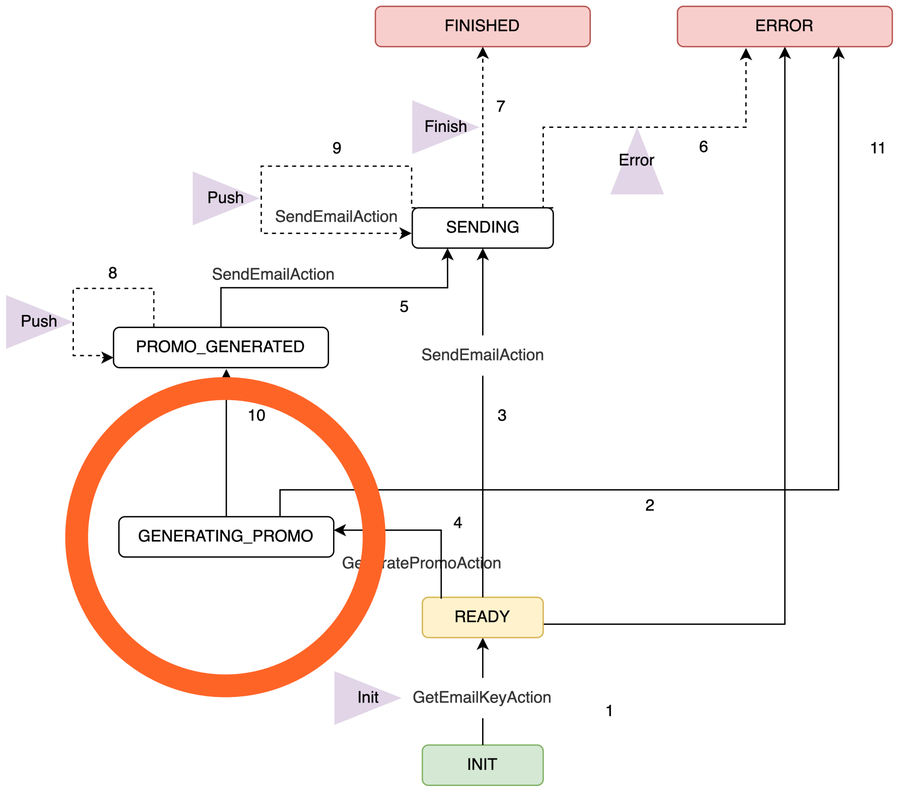
03 Покрыть тестами

```
@Test
public void testInitToSendPromoEmail() throws Exception {
    PromoCodeRequestEntity promoCodeRequestEntity = generatePromoCodeRequest(x -> x
                    .status(PromoCodeRequestStatus.INIT)
                    .externalCheckResult(ExternalCheckResult.OK));
    generateStudentPromoCodeRequest(promoCodeRequestEntity.getId(), x -> x);

    promoCodeRequestFSM.handleEvent(promoCodeRequestEntity.getId(), PromoCodeRequestEvent.INIT);
    //здесь должен сгенериться промокод и отправиться на почту

    checkPromoEmailSent(promoCodeRequestEntity);
}
```

Как внести изменение



01 Добавить переход

02 Реализовать новые экшены и гарды

03 Покрыть тестами

04 Выкатить

Что в итоге получилось

+ Структурированный код

+ Удобство дебага

+ Fail-safe

+ Переиспользуем движок в других проектах

- Продуктовые гэпы

- Много функциональности, которая нам не нужна (оверхед на конфигурацию)

Команда

Окей, мы решили задачу для Биллинга.
А как еще можно подойти к подобной проблеме?

Телемост: Стейт-машина со «Стейт-водителем»

Хотим запустить трансляции в **Телемосте**

Большое количество последовательных интеграций

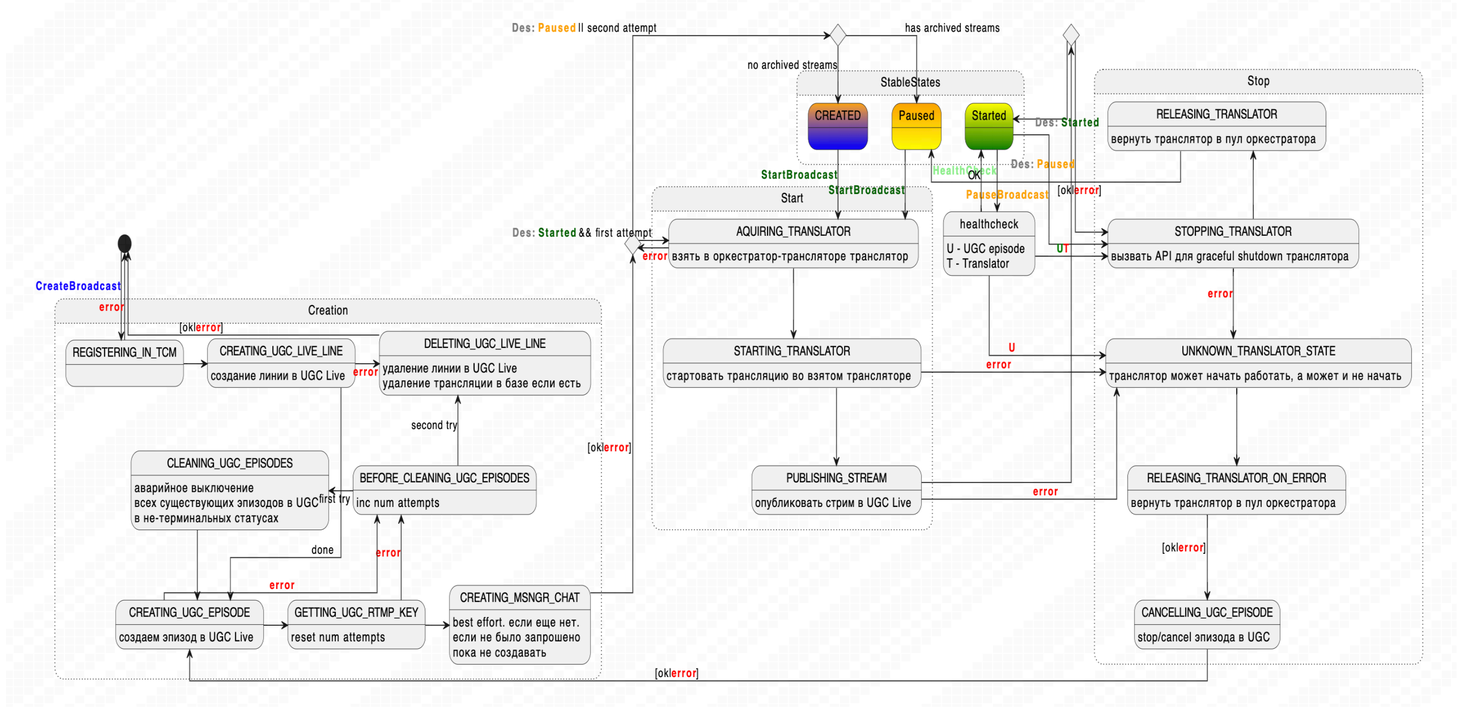
Должны уметь ретраиться

Должны часто сохраняться

А подробнее про это можно посмотреть в презентации автора:

Дмитрий Некрылов (topright)

Телемост: Стейт-машина со «Стейт-водителем»



А подробнее про это можно посмотреть в презентации автора:

Дмитрий Некрылов (topright)

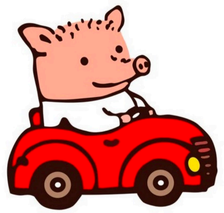
Телемост: Стейт-машина со «Стейт-водителем»

```
public interface BroadcastStateProcessor {
    void perform(BroadcastStateTransitionContext ctx) throws Exception;

    void persistInTransaction(BroadcastStateTransitionContext ctx) throws Except
    //must attempt recover in case of non-idempotent integrations
    //when this recovery is needed, state won't move, task will error out and wi
    void recoverPersistFailed(BroadcastStateTransitionContext ctx, Exception e)

    BroadcastState nextState(BroadcastStateTransitionContext ctx);
}
```

BroadcastStateService



https://got.reactor.cc

Стейт-водитель

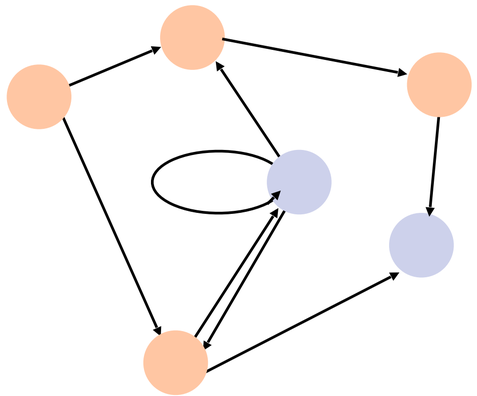
А подробнее про это можно посмотреть в презентации автора: **Дмитрий Некрылов** (topright) www.youtube.com

Стейт-машина решает все проблемы?

Не стейт-машиной единой

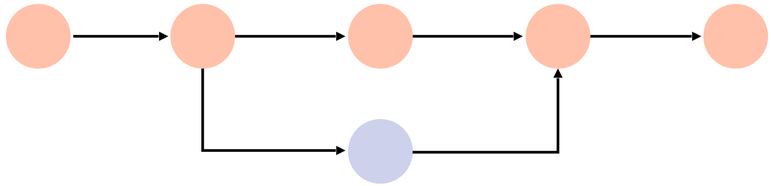
Когда стоит использовать стейт-машину

- Система имеет выраженные состояния
- Система совершает переходы между состояниями
- Система имеет циклические зависимости

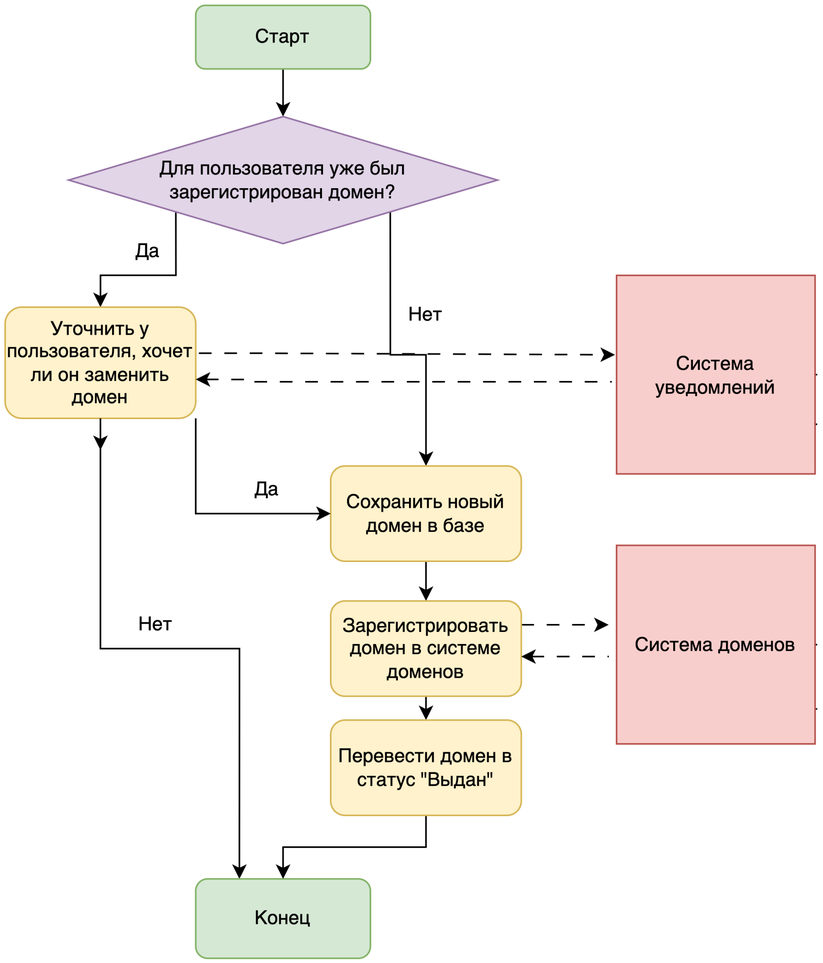


Когда НЕ стоит использовать стейт-машину

- Нужно придумывать состояния, чтобы система работала
- Переходы между состояниями не имеют цикличности и представляют собой прямой процесс
- Граф состояний и переходов стейт-машины делает задачу запутанной



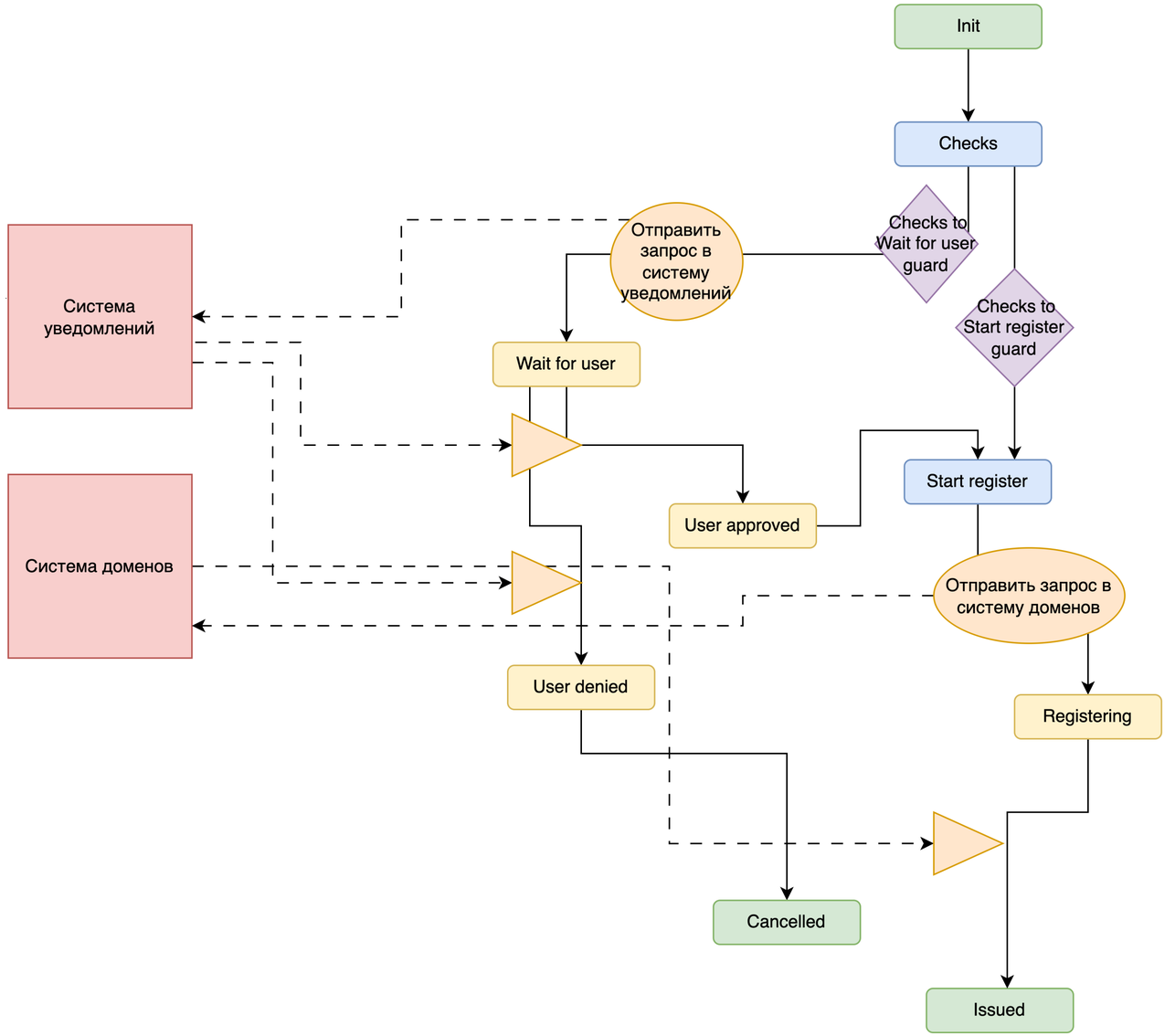
Процесс выдачи домена пользователю



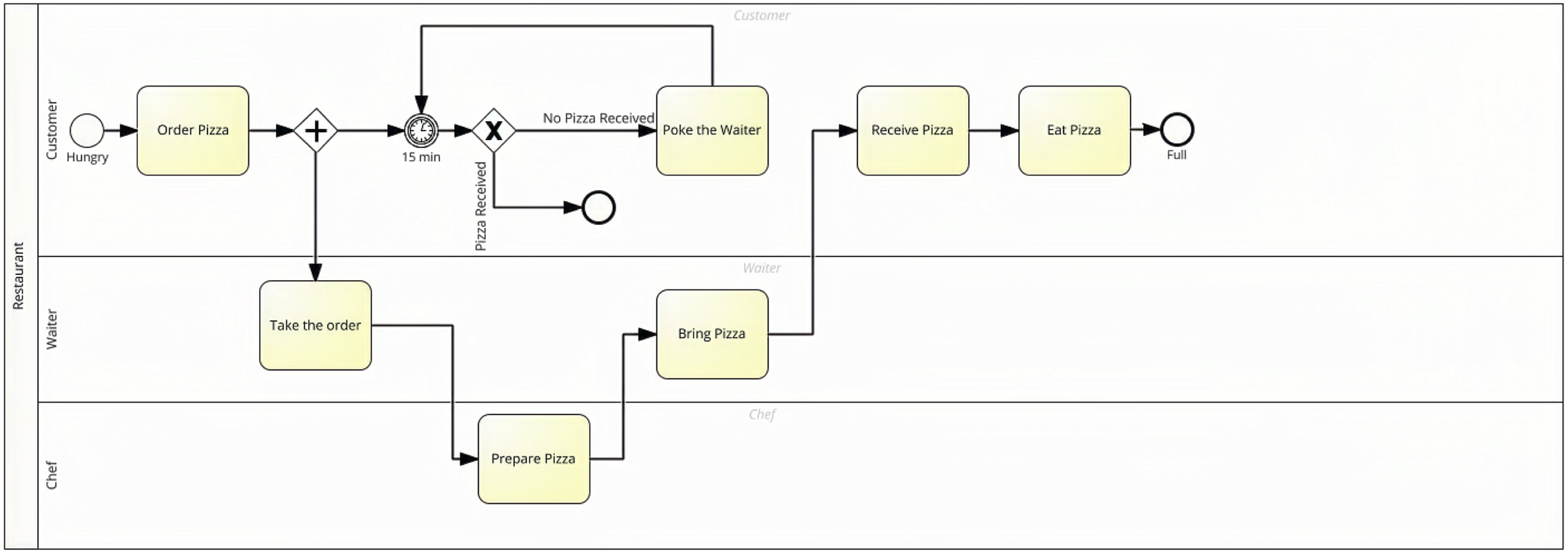
Итого: 4 задачи, 2 условных перехода между ними, 3 безусловных

Процесс выдачи домена пользователю

(попытка применить стейт машину)



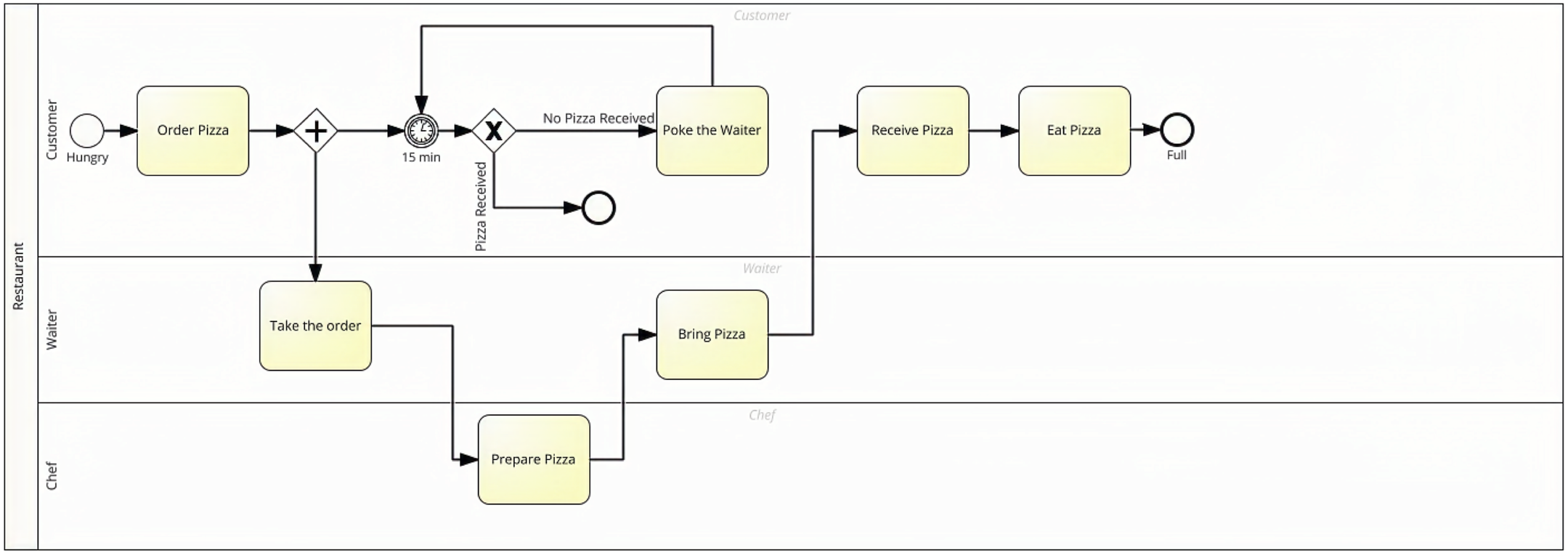
Причем здесь камунда?



https://mavink.com

Рабочий процесс

Причем здесь камунда?



https://mavink.com

Рабочий процесс

camunda

А что в итоге?

Команда

Бизнес

Яндекс 360

Спасибо за внимание!

@teadrunkage

Ищите команду Яндекс 360 на стенде Яндекса