contactpay

Ещё один фреймворк

Как мы стали пилить интеграции за спринт вместо квартала

О докладчике

- ☑ Python-разработчик
- ☑ В основном фин-тех
- ☑ Team-lead

contactpay

О докладе

- [x] Опыт многочисленных интеграций

contactpay

О докладе

- ☑ Опыт многочисленных интеграций
- ☑ **Любая предметная область**

contactpay

О докладе

- ☑ Опыт многочисленных интеграций
- ☑ Любая предметная область
- ☑ **Не только для типовых интеграций**

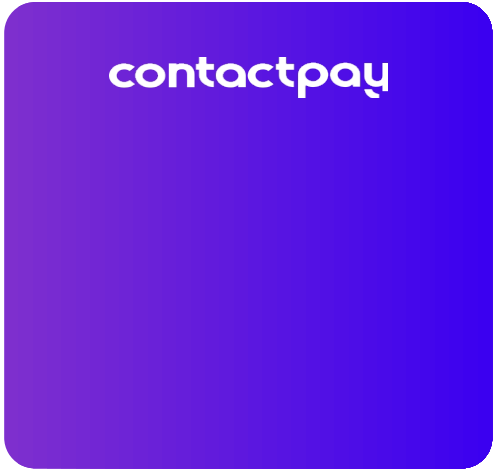
План

- [x] Проблемы роста
- [x] Наши решения
- [x] Анализ результатов
- [x] Нерешённые проблемы
- [x] Выводы

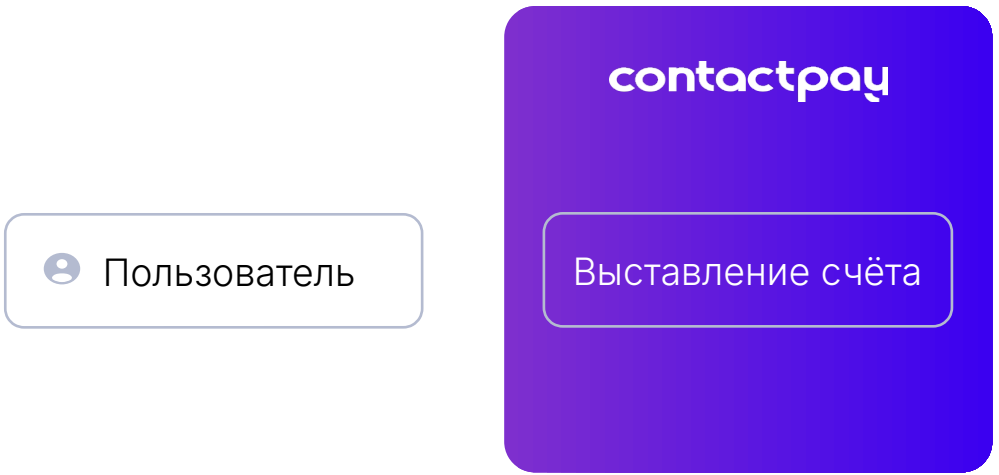
contactpay

В чём проблема?

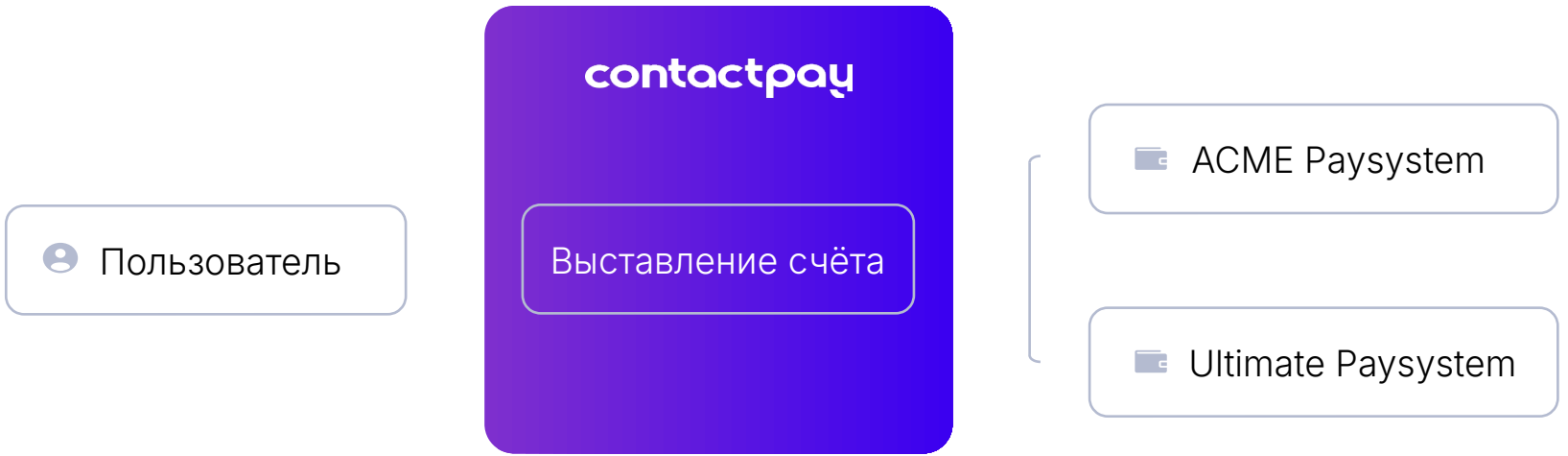
Как всё устроено



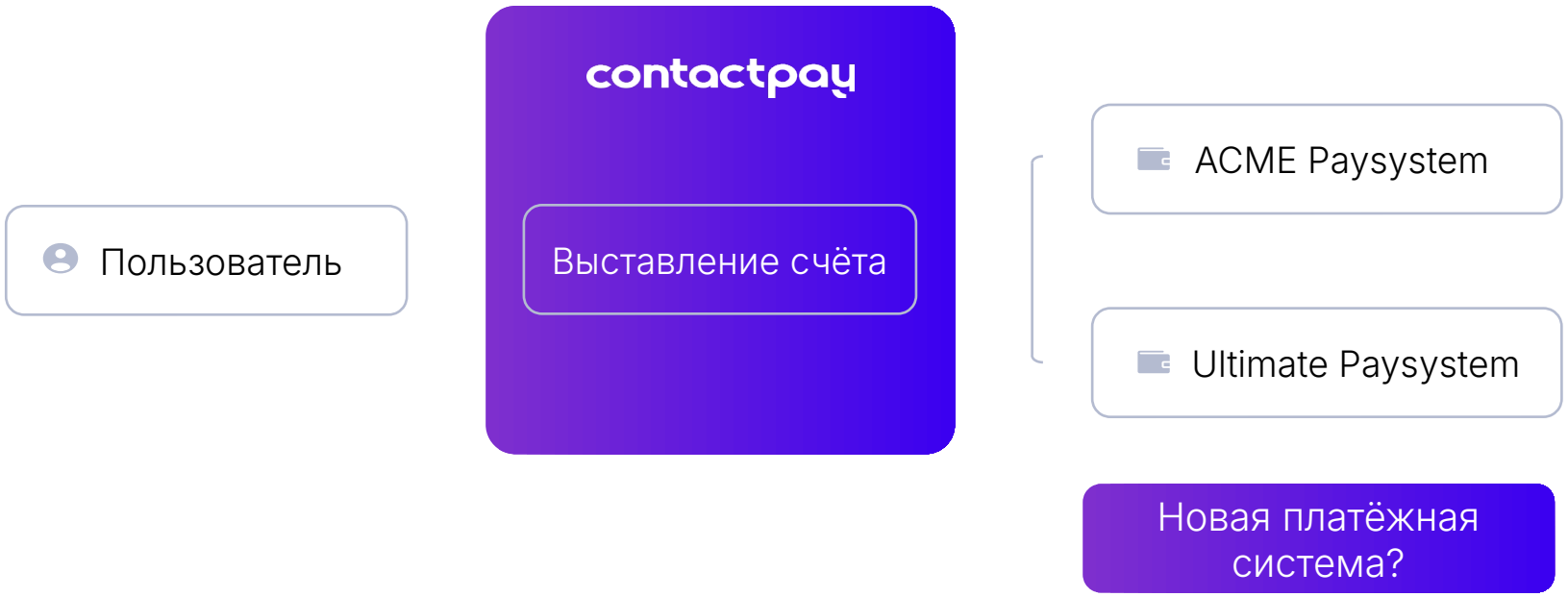
Как всё устроено



Как всё устроено



Как всё устроено



contactpay

Расчехляем Ctrl-C + Ctrl-V

Проблема: Ctrl-C + Ctrl+V

Подходит

- [x] Редкие интеграции
- [x] Написал и забыл
- [x] Сжатые сроки

Проблема: Ctrl-C + Ctrl+V

Подходит

- ☑ Редкие интеграции
- ☑ Написал и забыл
- ☑ Сжатые сроки

Не подходит

- ☑ Большой поток интеграций
- ☑ Частые изменения в бизнес-логике

Копирование: масштабы бедствия

	Без изменений	Именования	Логика	Всего кода
Интеграция 1	**58%**	5%	**37%**	1725
Интеграция 2	**66%**	3%	**31%**	2565
Интеграция 3	**59%**	5%	**36%**	1621
Интеграция 4	**50%**	5%	**45%**	1240

Проблема: **все врут**

- [x] **Системы и сети ненадёжны**

contactpay

Проблема: **все врут**

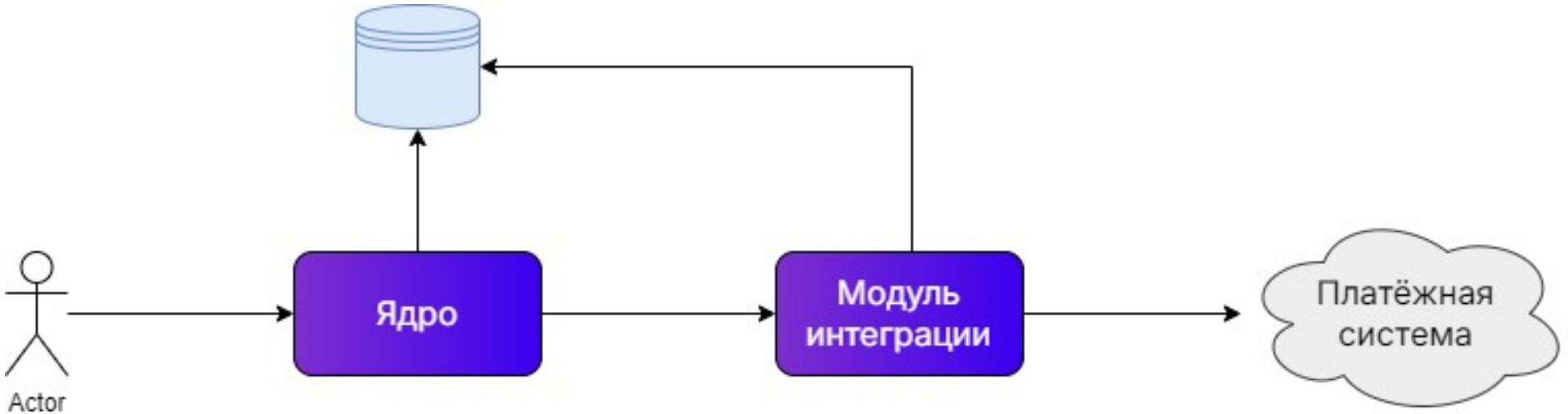
- [x] Системы и сети ненадёжны
- [x] **Документация не отражает реальности**

contactpay

Проблема: **все врут**

- [x] Системы и сети ненадёжны
- [x] Документация не отражает реальности
- [x] **Техническая поддержка ошибается**

Проблема: **влияние на систему**



contactpay

Пример: **влияние на систему**

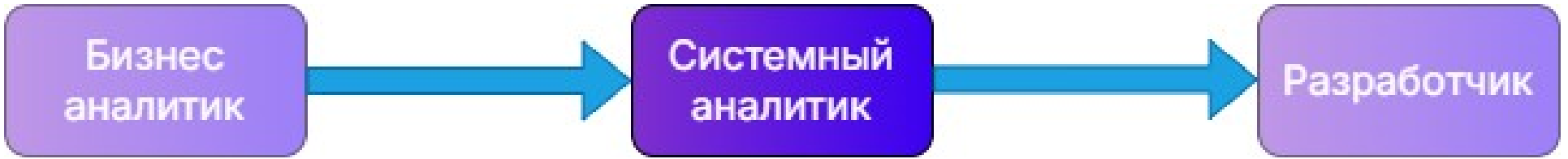
```python
def _handle_success(self, response: PaysystemResponse) -> PaysystemModuleResponse:
    self.invoice.set_status(PaymentStatus.SUCCESS)
    self.invoice.update_paysystem_data(paysystem_invoice_id=str(response.id))
    self.invoice.update_processed()
    self.invoice.save()

    return PaysystemModuleResponse(
        invoice_id=self.invoice.id,
        status=self.invoice.status,
        paysystem_data=self.get_invoice_paysystem_data(),
    )
```

Проблема: копируют не только разработчики

- ☑ Системные аналитики тоже ленятся

Место аналитика в разработке



contactpay

Проблема: **копируют не только разработчики**

- ☑ Системные аналитики тоже ленятся
- ☑ **Требования описывают влияние на систему**

Проблема: **копируют не только разработчики**

- ☑ Системные аналитики тоже ленятся
- ☑ Требования описывают влияние на систему
- ☑ **Системные требования сильно влияют на архитектуру**

Пример: **старые требования**

Если response.status = success, то:

- Изменить статус платежа на SUCCESS
- Записать в поле invoices.paysystem_data значение {"internal_status_code": <response.status>}
- Вернуть ответ {"result": true, "error_code": 0, "message": "success", "error_info": {"code": = <internal_status_code>}

Пример: **старые требования**

Если response.status = success, то:

Низкий уровень обработки

- Изменить статус платежа на SUCCESS
- Записать в поле invoices.paysystem_data значение {"internal_status_code": <response.status>}
- Вернуть ответ {"result": true, "error_code": 0, "message": "success", "error_info": {"code": = <internal_status_code>}

Пример: **старые требования**

Прямое влияние на состояние системы

Если response.status = success, то:

- Изменить статус платежа на SUCCESS
- Записать в поле invoices.paysystem_data значение {"internal_status_code": <response.status>}
- Вернуть ответ {"result": true, "error_code": 0, "message": "success", "error_info": {"code": = <internal_status_code>}

contactpay

Пример: **старые требования**

Если response.status = success, то:

- Изменить статус платежа на SUCCESS
- Записать в поле invoices.paysystem_da
  {"internal_status_code": <response.status>}
- Вернуть ответ {"result": true, "error_code": 0, "message": "success", "error_info": {"code": = <internal_status_code>}

Ответственность за более высокий уровень

Резюме: **проблемы**

- [x] Перешли этап, когда простое копирование работает
- [x] Внешним системам нельзя слепо доверять
- [x] У модуль интеграции большая зона ответственности
- [x] Больше кода, больше требований — больше проблем

Как решали проблемы?

Решение 1: **Выделение ядерной логики**

Что решали:

- [x] Множественное копирование
- [x] Высокое влияние модуля интеграции

Решение 1: Выделение ядерной логики

Что решали:

- [x] Множественное копирование
- [x] Высокое влияние модуля интеграции

Что сделали:

- [x] Перенесли управление сущностями в ядро

Пример: **бизнес-логика**, было

```
def create_invoice(self, invoice: Invoice) -> PaysystemModuleResponse:
    response = requests.post(...)
    return _handle_success(PaysystemResponse.from_http(response))

def _handle_success(self, response: PaysystemResponse) -> PaysystemModuleResponse:
    self.invoice.set_status(PaymentStatus.SUCCESS)
    self.invoice.update_paysystem_data(paysystem_invoice_id=str(response.id))
    self.invoice_update_processed()
    self.invoice.save()

    return PaysystemModuleResponse(
        invoice_id=self.invoice.id,
        status=self.invoice.status,
        paysystem_data=self.get_invoice_paysystem_data(),
    )
```

Пример: **бизнес-логика**, стало

```python
def create_invoice(self, invoice: Invoice) -> CoreResponse:
    integration_module = invoice.get_integration()
    invoice_dto = to_invoice_dto(invoice)
    paysystem_response = integration_module.create_invoice(invoice_dto)

    new_status = paysystem_response.get_target_status()
    invoice.set_status(new_status)

    ps_data = paysystem_response.get_paysystem_data()
    invoice.update_paysystem_data(ps_data)

    invoice_update_processed()
    invoice.save()

    return CoreResponse(
        invoice_id=invoice.id,
        status=new_status,
        paysystem_data=ps_data,
    )
```

Побочный эффект: **Инвертирование зависимости**

```python
class OperationResultInfo(ABC):

    @abstractmethod
    def get_target_invoice_status(self) -> InvoiceStatus | None:
        pass

    @abstractmethod
    def get_paysystem_invoice_id(self) -> str | None:
        pass

    @abstractmethod
    def get_paysystem_data(self) -> AcquirerData | None:
        pass

    @abstractmethod
    def get_redirect_info(self) -> RedirectInfoDTO | None:
        pass

    @abstractmethod
    def get_error_info(self) -> ErrorInfo | None:
        pass
```

«Но вы же просто выделили Core Domain?»

— Адепты DDD

Да.

Да. Но мы делали по SOLID

Решение 2: Good, Bad, Ugly API

Что решали:

- [x] Большое влияние внешних факторов

Решение 2: Good, Bad, Ugly API

Что решали:

- [x] Большое влияние внешних факторов

Что сделали:

- [x] Переосмыслили обработку ответов внешних систем

Решение 2: Good, Bad, Ugly API

Good

- [x] Операция прошла успешно

contactpay

Решение 2: Good, Bad, Ugly API

Good

- [x] Операция прошла успешно

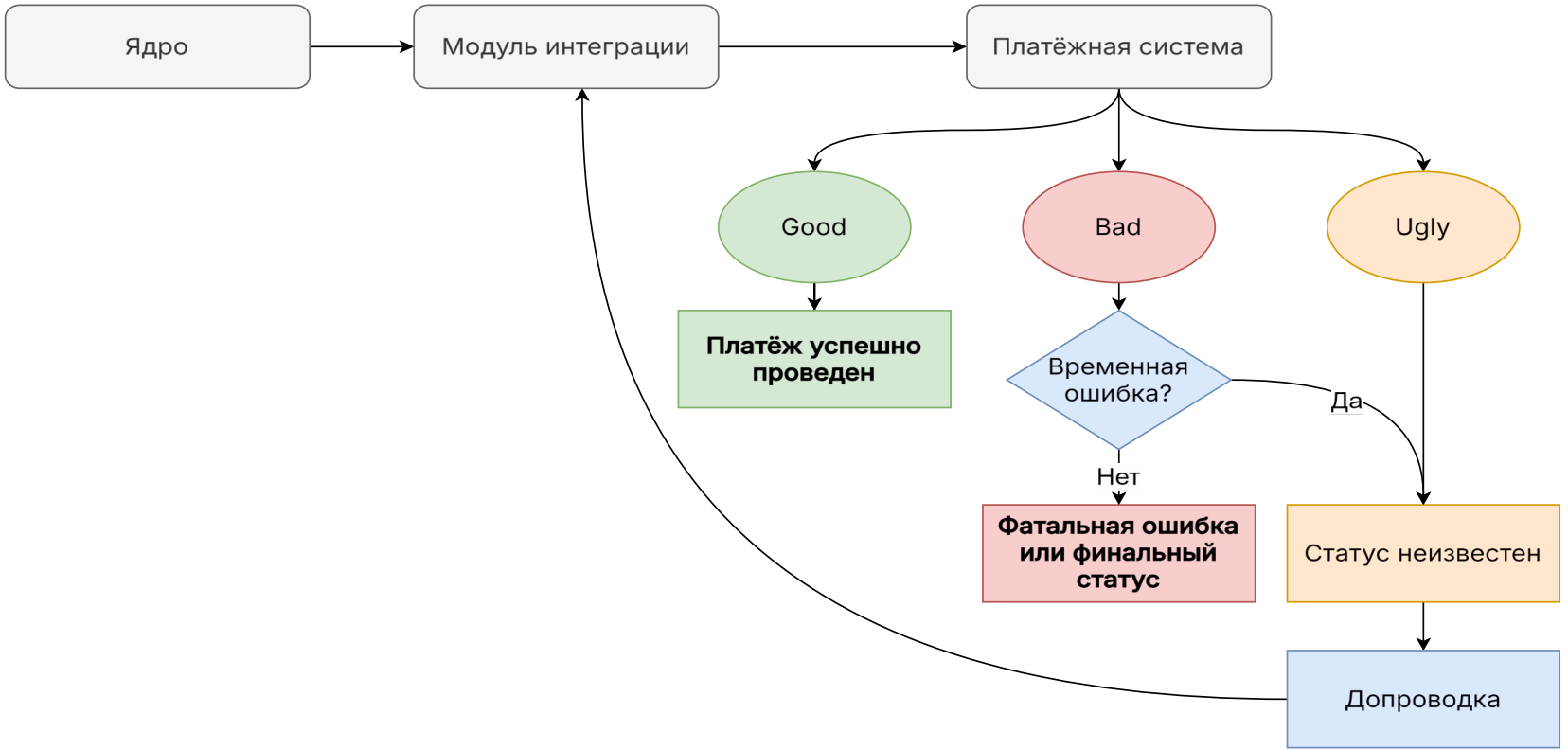
Bad

- [x] Операция завершилась **с явной** ошибкой
- [x] Повтор операции — тоже Bad
- [x] ... или восстановление

Решение 2: **Good, Bad, Ugly API**

Good	Bad	Ugly
☑ Операция прошла успешно	☑ Операция завершилась **с явной** ошибкой	☑ Нет уверенности в том, что произошло
	☑ Повтор операции — тоже Bad	☑ Повтор операции непредсказуем
	☑ ... или восстановление	☑ Требуется механизм восстановления

Как работает **Good, Bad, Ugly?**



contactpay

Решение 3: **Артефакты аналитика**

Что решали:

- [x] Много требований — много проблем
- [x] Аналитик должен знать слишком много

contactpay

Решение 3: **Артефакты аналитика**

Что решали:

- [x] Много требований — много проблем
- [x] Аналитик должен знать слишком много

Что сделали:

- [x] Шаблонизировали документацию и процессы

Пример: **результат операции**

	Good	Bad (1)
Интерпретация	Выплата успешно создана	При создании выплаты произошла ошибка валидации
Условия	HTTP 200 response.status=INIT/HOLD	HTTP 400+
get_status()	PaymentStatus.InProcess	PaymentStatus.Rejected
get_payment_system_status()	response.status	None
get_paysystem_payment_data()	PaySystemPaymentData: - provider_payment_id=response.id	PaySystemPaymentData: - provider_processed=now()
get_error_data()	None	ErrorData: - error_code=PaymentSystemStatus.invalid_request_data - provider_error_code=None - provider_error_message=Склеить response.error (optional) и response.message (см. примеры)
get_payment_existence()	PaymentExistence.EXIST	PaymentExistence.NOT_EXIST

contactpay

Пример: **опросник аналитика**

- [x] Какие параметры являются **обязательными** для создания платежа?
- [x] Какие параметры являются **опциональными** для создания платежа?
- [x] Можно ли указать **собственный идентификатор** платежа при создании на стороне платёжной системы
- [x] Используются ли механизмы обеспечения **идемпотентости** при создании платежа
- [x] Какой параметр ответа отражает **идентификатор платежа** на стороне платёжной системы?

Резюме: **решения**

- ☑ Выделение ядерной логики
- ☑ Good, Bad, Ugly API
- ☑ Артефакты аналитика

Что получилось в итоге?

Разработчики счастливы

- [x] Бизнес логика написана один раз и (почти) навсегда
- [x] Нужно писать меньше кода
- [x] Порог вхождения снизился

Аналитики счастливы

- ☑ Не нужно копировать сложные требования
- ☑ Не нужно разбираться во внутреннем устройстве системы
- ☑ Нужно заполнить всего несколько таблиц

Бизнес счастлив

- [x] Время на разработку **-50%**
- [x] Время Time to Market **-30%**
- [x] Средние затраты на разработку **-80%**
- [x] Стоимость фреймворка **1-senior-month**

contactpay

Не всё так безоблачно…

Проблемы, которые **не решили** полностью

- [x] Человеческий и технический фактор

contactpay

Проблемы, которые **не решили** полностью

- [x] Человеческий и технический фактор
- [x] **Boiler-plate теперь другой**

Проблемы, которые **не решили** полностью

- [x] Человеческий и технический фактор
- [x] Boiler-plate теперь другой
- [x] **Старый код остаётся**

Подводя итоги

Итого

- [x] Меньше кода — выше скорость разработки

Итого

- [x] Меньше кода — выше скорость разработки
- [x] **Хорошая архитектура — меньше кода**

Итого

- [x] Меньше кода — выше скорость разработки
- [x] Хорошая архитектура — меньше кода
- [x] **Понимание бизнес процесса помогает отбросить лишнее**

Итого

- [x] Меньше кода — выше скорость разработки
- [x] Хорошая архитектура — меньше кода
- [x] Понимание бизнес процесса помогает отбросить лишнее
- [x] **Фреймворк это не только про код, но и про процессы**

Итого

- [x] Меньше кода — выше скорость разработки
- [x] Хорошая архитектура — меньше кода
- [x] Понимание бизнес процесса помогает отбросить лишнее
- [x] Фреймворк это не только про код, но и про процессы
- [x] **До фреймворка нужно вырасти**

contactpay

На связи

Дмитрий Иванюшин

d.ivanyushin@contactpay.com
defance@gmail.com

@defance