

^ cian
dream
team

^ cian
dream
team

«Backend-Driven UI: как мы это сделали и что из этого вышло»
Александр Киверин

PiterPy 2024



Head of Python / Python Developer

- 4 года в Циан
- Успел поработать со всеми основными компонентами (поиск, карточка, личный кабинет, форма подачи объявления)
- В гильдии улучшаю процессы
- Слежу за DevEx
- И автоматизирую все, что только можно



1. Предыстория

2. Техническая сложность

3. Как тестировать?

4. Продуктовая сложность



1. Предыстория

2. Техническая сложность

3. Как тестировать?

4. Продуктовая сложность

BDUI - это?

- Это подход
- Максимально тонкие клиенты
- Клиент определяет базовые компоненты
- Раньше - клиент собирает из них экран
- Сейчас - бэкенд собирает из них экран



Особенности квартиры

Балконы

Балкон - 0 +

Лоджия - 0 +

Вид из окна

На улицу

Во двор

Санузел

Раздельный - 0 +

Совмещённый - 0 +

Ремонт

Без ремонта

Косметический

Евро

Дизайнерский

[Как выбрать вид ремонта](#) ▾

Лифты

Пассажирский - 0 +

Грузовой - 0 +

BDUI - это?

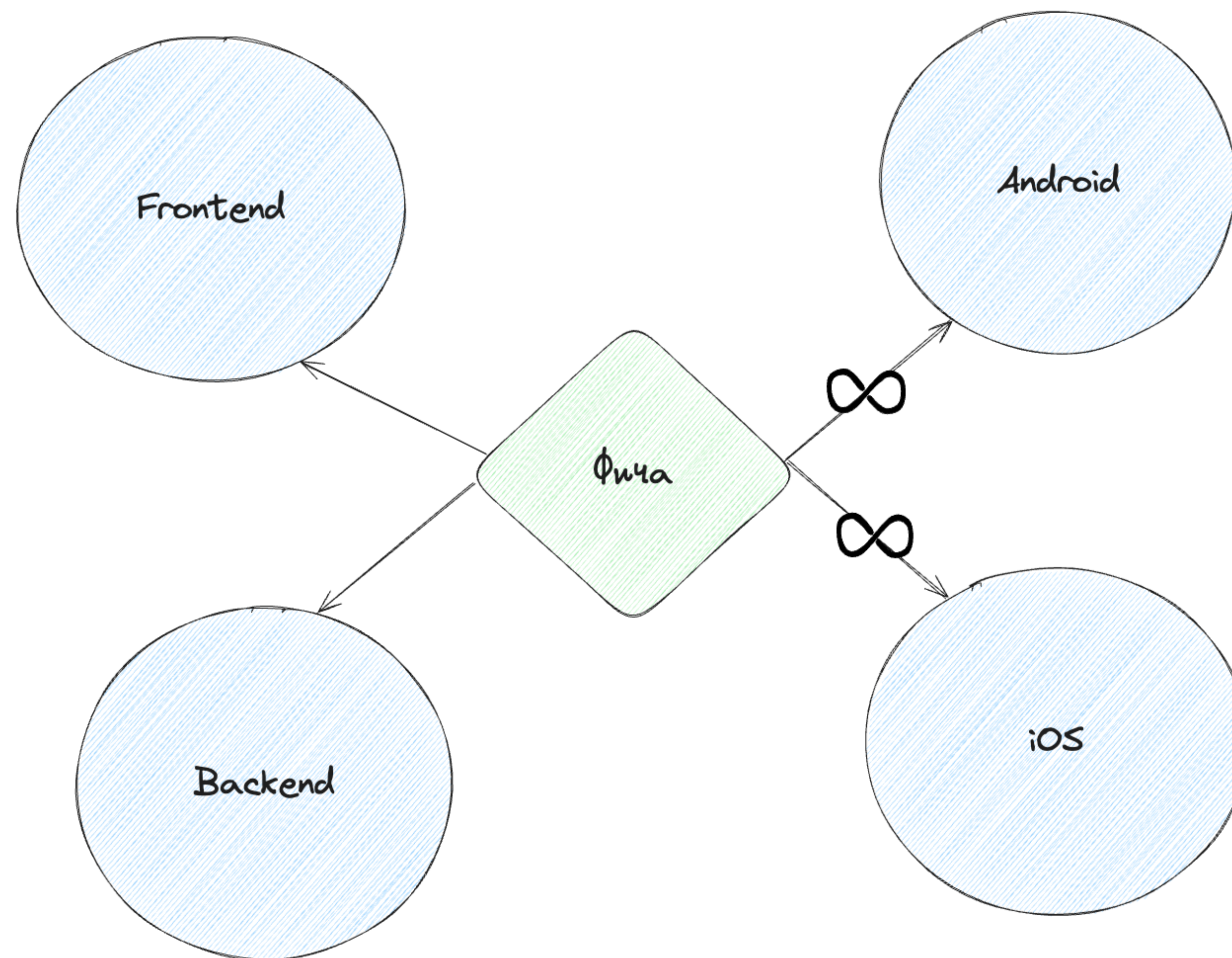


- JSON из API - определяет наполнение экрана на клиенте
- Клиент разбирает JSON и расставляет компоненты на экране
- Для всех платформ

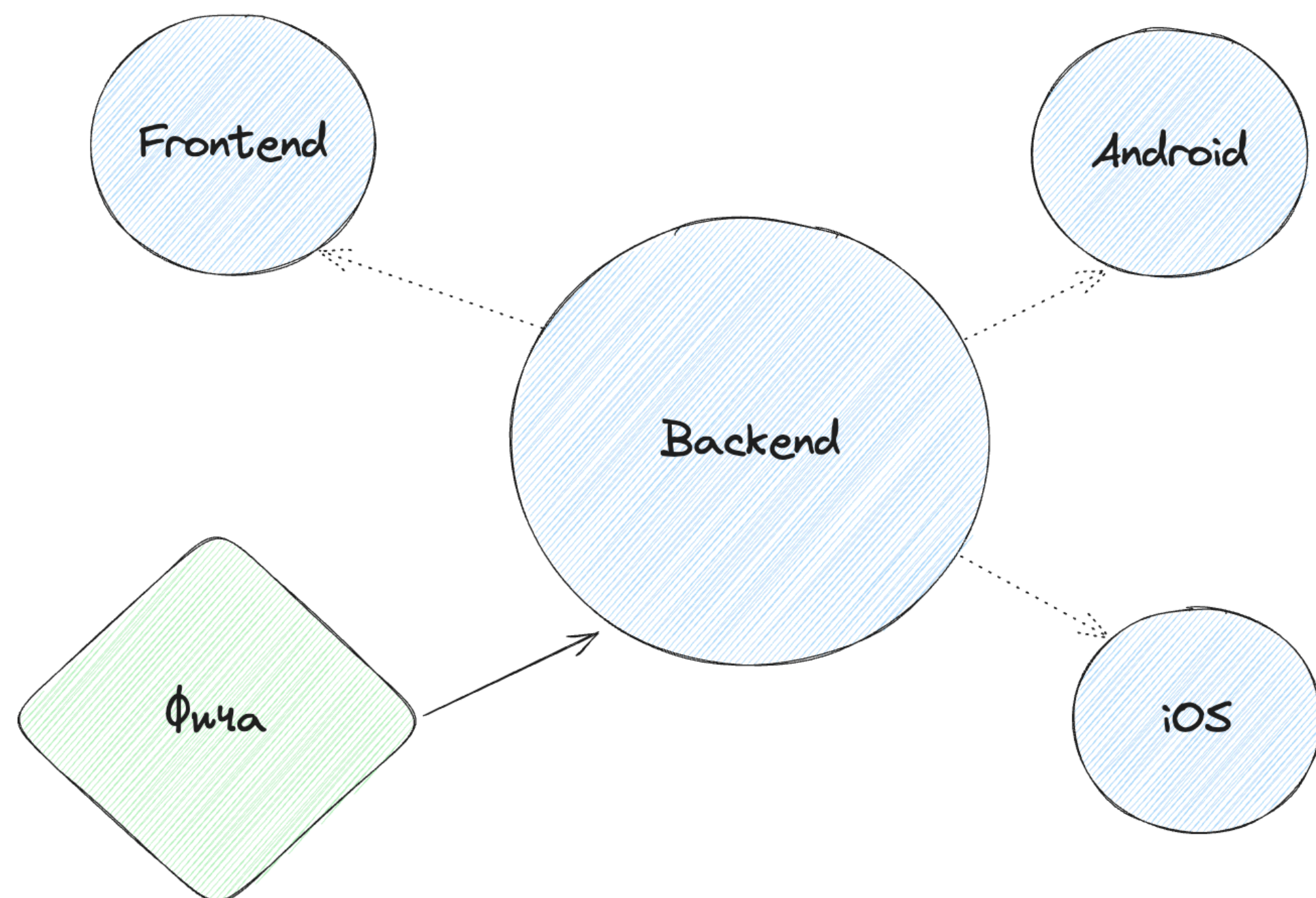
```
return (  
  <>  
    <TitleContainer />  
    <LabelsListContainer />  
    <RatingContainer />  
    <GeoContainer />  
    <AuctionTitleContainer />  
    <PriceContainer  
      isNewHighlighting={isNewHighlighting}  
      MortgageAdFoxBanner={<ContentRow>{mortgageBanner}</  
ContentRow>}  
    />  
    <SimilarLinkContainer />  
    <IdenticalOffersLinkContainer />  
  </>  
)  
);  
}
```

```
{  
  "columns": [  
    {  
      "components": [  
        {  
          "id": "step_description_title_1",  
          "type": "typography"  
        }  
      ]  
    },  
    {  
      "columns": [  
        {  
          "components": [  
            {  
              "id": "step_description_input_0",  
              "type": "input"  
            }  
          ]  
        }  
      ]  
    }  
  ],  
}
```

Разработка до BDUI



Разработка после BDUI



Зачем?

Обходим релизный цикл приложений
Вся логика в 1 месте

Как?

JSON Driven Development
JSON определяет UI и логику на клиентах
Максимум логики лежит на бэкенде

История: Форма подачи объявления



Новое объявление
Собственники могут разместить по одному объявлению в каждой категории бесплатно.
Все условия и тарифы в разделе «Прайс».

Активный черновик ?

Тип объявления [Правила размещения](#)

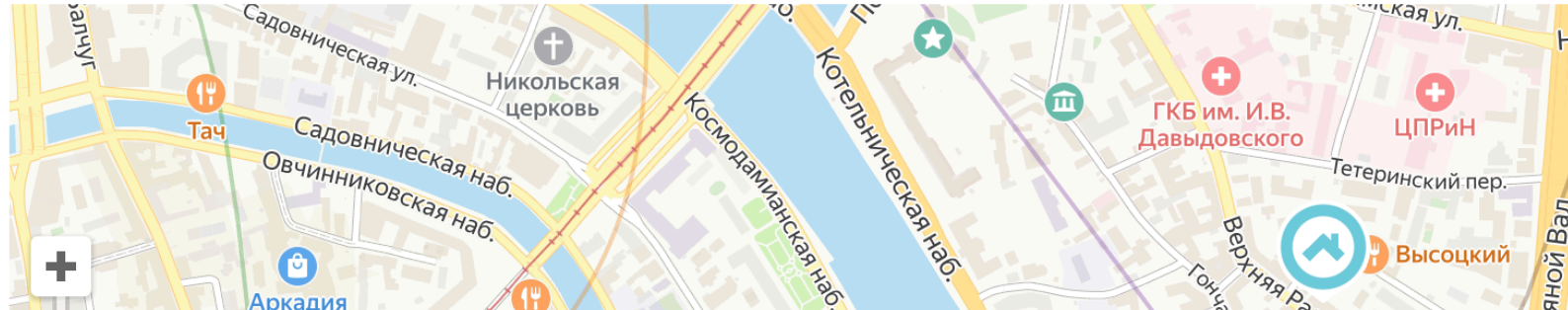
[Продажа комнаты](#) [Изменить](#)

Видимость ? [Видно всем](#) [Видно только агентам](#)

Адрес [Проблема с вводом адреса?](#)

Россия, Москва, Верхняя Радищевская улица, 11с1 ✓

Редактировать адрес можно в течение 2 дней после публикации



Тип объявления ✓
Адрес ✓
Об объекте
О здании ✓
Инфраструктура ✓
Описание и фото ✓
Цена и условия сделки
Контакты
Условия размещения ✓

Адрес можно указать прямо на карте

Подвиньте отметку, чтобы указать или исправить адрес объекта

[Сообщить о проблеме](#)

[Звоните о сайте](#)

Доработки

Долгие - логика делится между 3 клиентами и бэкендом
Трудные - устаревшие технологии и подходы (старый Angular)

Почему именно VDUI для ФП?



- Переписать форму точно нужно
- Есть 3 платформы
- Продукту нужны **быстрые эксперименты**
- Добавить новое поле на всех платформах **за день**
- VDUI хорошо дружит с формами - простая логика, приводится к базовым компонентам
- Есть культура тонких клиентов в компании

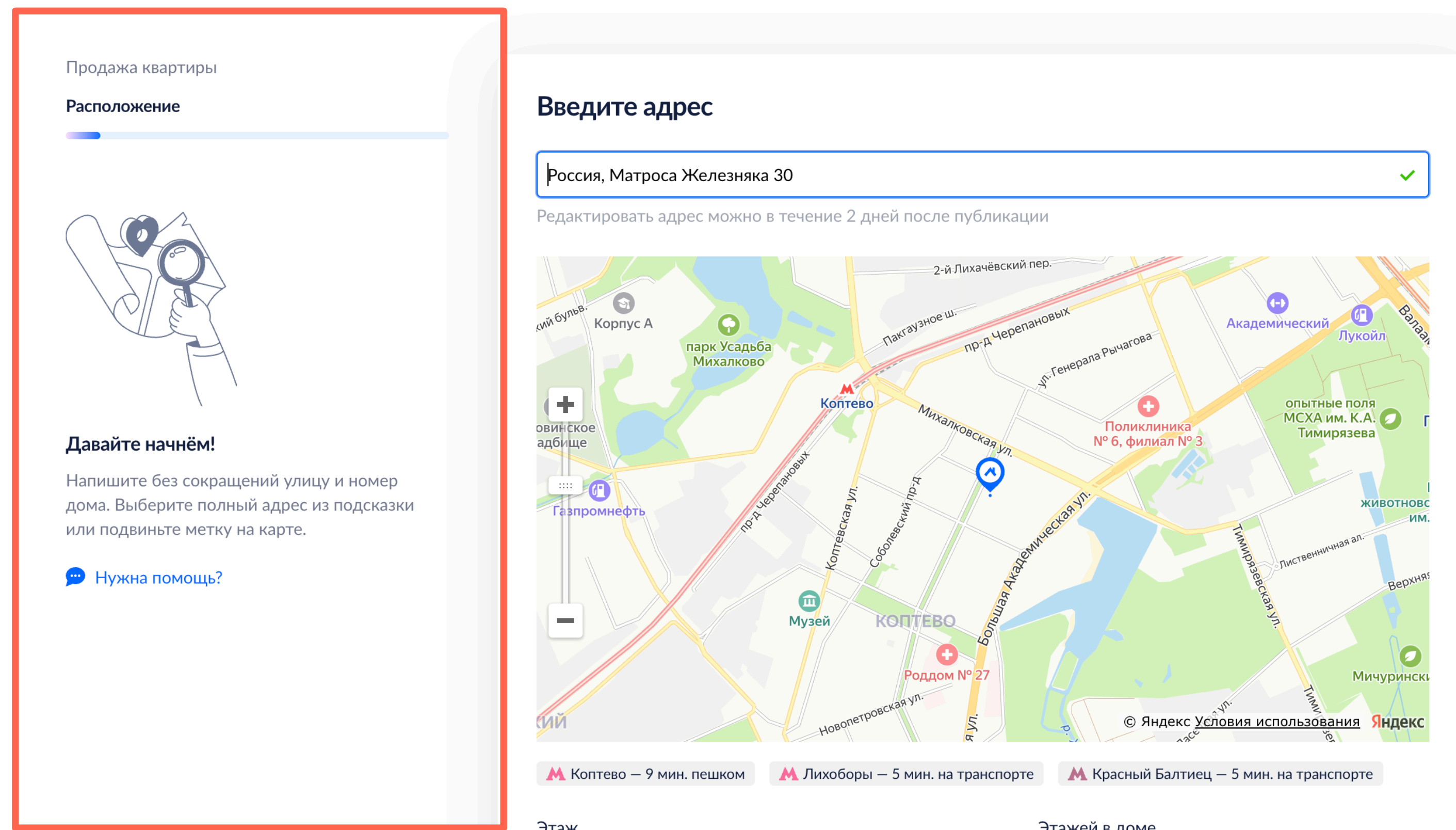
Почему не DivKit?



- Трудно создавать кастомные компоненты - не привязать дизайн системе
- Ограничены возможности анимации
- Нет полноценной поддержки цветовых тем
- Так было в 2022!

История: Разумный подход на web в 2022

- MVP решение
- Специально для формы подачи
- Язык BDUI только под web
- Только основная часть формы
- Так как меняется чаще всего



Это не BDUI часть формы

История: Гибкость на apps в 2023

- Платформенное решение
- Гибкое
- Для любых экранов
- Не поддерживается на web (пока)
- Итого 2 отдельных языка и 1 бэкенд
- Подробнее тут
<https://www.youtube.com/watch?v=b7Wcfm2GvSM>

← Продажа квартиры ...

Параметры квартиры

Количество комнат

Студия 1 2 3 4 5

6+ Свободная планировка

Общая площадь

м²

Жилая площадь

м²

Как посчитать площадь ▾

Кухня

м²

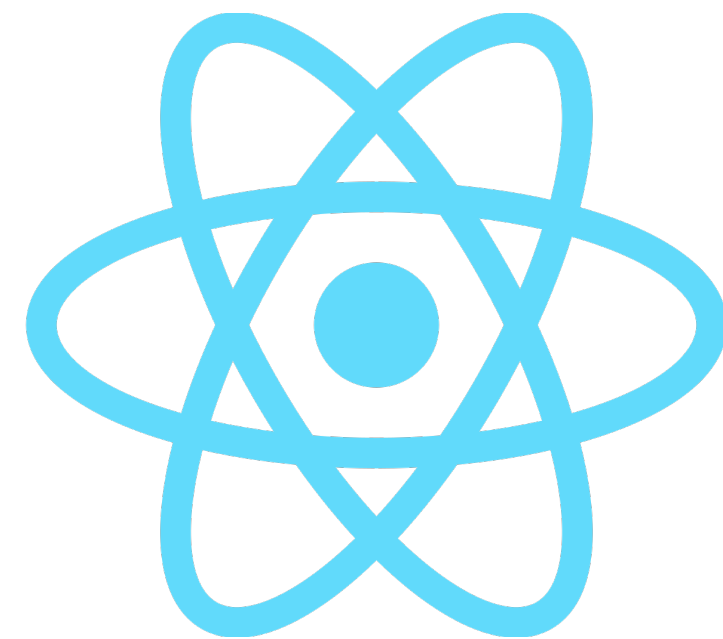
Высота потолков

м

Этаж

Продолжить

Что использовали на клиентах?



Стек

- Web - Typescript + React
- Android - Kotlin
- iOS - Swift



1. Предыстория

2. Техническая сложность

3. Как тестировать?

4. Продуктовая сложность

OpenAPI и Swagger



POST /public/v1/goto-next-step/

Parameters Try it out

Name	Description
body * required object (body)	Example Value Model <pre>[, "oneChoice": [{ "code": "string", "value": "string" }], "phone": [{ "code": "string", "value": "string" }], "photo": [{ "code": "string", "value": [{ "id": 0, "isDefault": true, "isLayout": true, "rotateDegree": 0 }] }]]</pre>

Parameter content type: application/json

Что это?

Язык для описания схем API запросов и ответов

Зачем?

Можно генерировать код API
Можно проверять совместимость схем сервера и потребителя
Можно смотреть на схемы через Swagger

Пример

Схемы API определяются через **dataclasses**

OpenAPI схемы генерируются автоматически

Сериализация срабатывает автоматически

Первая проблема: Клиенты мыслят рекурсивно



```
▼ <body> flex
  ▶ <noscript>...</noscript>
  ▶ <script type="text/javascript">...</script>
  ▶ <style type="text/css">...</style>
  ▶ <header id="header-frontend" class="_25d45facb5--container--wGgfr">...</header>
  ▶ <script type="text/javascript">...</script>
  <script src="//static.cdn-cian.ru/frontend/header-frontend/main.0b96088...js" type="tex
  projectname="header-frontend" data-requestid="0d97499a35509" data-versioncommit="38890
  ▶ <script type="text/javascript">...</script>
  ▼ <main id="frontend-mainpage" class="mainpage">
    ▼ <section class="_025a50318d--mainpage--q6lg6">
      ▼ <div>
        ▼ <div data-name="GeoSwitcher" class="_025a50318d--wrapper--QHy2S"> flex
          ▶ <button class="_025a50318d--button--eBKAY">...</button> flex
          </div>
        </div>
        ▼ <div>
          <div></div>
          ▼ <div data-name="FiltersBlock" data-testid="FiltersBlock" class="_025a50318d--c-
            ▼ <div class="_025a50318d--c-filters-content--nnkcT">
              ▶ <div class="_025a50318d--c-filters-inner--zQgkv">...</div>
              ▶ <div>...</div>
            </div>
            ▼ <div data-name="FiltersBackground" class="_025a50318d--container--nL7e0"> ==
              ▶ <div class="_025a50318d--old-image--VWQB0 _025a50318d--index--eRYeu" style=
                t/get-adfox-content/2774030/240306_adfox_2909546_8467962.f50200ecfb9fa99baf
                </div>
              </div>
            </div>
          </div>
          ▶ <div>...</div>
          ▶ <div data-name="UsefulLinks" class="cg-container-fluid-xs _025a50318d--container-
          ▶ <section data-name="Recommendations" class="cg-container-fluid-xs _025a50318d--c
          ▶ <div class="cg-container-fluid-xs">...</div>
          ▶ <div data-name="Magazine" class="_025a50318d--c-magazine-container--H67iT">...</c
          ▶ <div data-name="SeoContainer" class="_025a50318d--c-seo--nc5ZZ">...</div>
        </div>
      </section>
    </main>
  ▶ <script type="text/javascript">...</script>
```

- Нужно складывать контейнеры внутри контейнеров
- Сериализатор не поддерживает бесконечную рекурсию
- Генератор OpenAPI схем не поддерживает (OpenAPI 2.1):
 - Не заиспользовать кодген
 - Не проверить схемы

Варианты решения



- Ограничиваем вложенность N уровнями
- Нужен новый сериализатор
- В OpenAPI 3.0 есть поддержка рекурсии через атрибут `ref`
- Но OpenAPI 3.0 не совместим с OpenAPI 2.1

Вторая проблема: Клиенты мыслят полиморфно



```
children=[
  Image(
    id=self.build_widget_id('calltrackingIcon'),
    params=ImageParams(
      image=StatedValue(
        value=ImageBinary(
          url=PHONE_ANDROID_ICON_URL
        ),
      ),
    ),
  ),
  Text(
    id=self.build_widget_id('calltrackingText'),
    params=TextParams(
      text=StatedValue(
        value=(
          'Защитим ваш личный номер'
        ),
      ),
    ),
  ),
],
```

- В одном списке могут быть компоненты разных типов
- Сериализатор не поддерживает
- Знает только про List[T]
- При List[T | U] нужно определять, как сериализовать текущий элемент
- Генератор OpenAPI схем не поддерживает (OpenAPI 2.1)



Защитим ваш личный номер от спама

Варианты решения

- Работаем с реестром компонентов.
- Определяем компоненты в одном месте
- Раскладываем по идентификаторам в другом
- Нужен новый сериализатор
- В OpenAPI 3.0 есть поддержка discriminator

```
{
  "category": "field",
  "code": "leaseType",
  "id": "termsAndContactsStep.leaseType",
  "margin": {
    "top": 24
  },
  "source": [
    {
      "label": "Прямая аренда",
      "value": "direct"
    },
    {
      "label": "Субаренда",
      "value": "sublease"
    }
  ],
  "style": "buttonGroup"
},
```

```
"components": [
  {
    "id": "termsAndContactsStep.leaseTypeText",
    "type": "typography"
  },
  {
    "id": "termsAndContactsStep.leaseType",
    "type": "oneChoice"
  }
],
```

Решение у нас



```
Widget = Annotated[
    Button
    Column
    Geo
    Header
    Row
    Box
    Scaffold
    Separator
    Spacer
    Text
    TextField
    ChipsField
    Image
    Stepper
    PhotoPicker
    TextArea
    Underground
    Kp
    SelectField
    Discriminator('type'),
```

Web

- Используем ограниченно вложенные схемы и реестр компонентов - полное покрытие OpenAPI схемами и стандартный сериализатор

Apps

- Используем serpyco-rs как сериализатор
- Умеет и в рекурсию и в полиморфизм
- Схемы описываются через **dataclasses** и **Annotated** (появилось в 3.9)
- Написан на Rust
- <https://github.com/ermakov-oleg/serpyco-rs>
- Переезд на OpenAPI 3.0 в процессе

Вспомним про 2 языка BDUI



- Разный вид JSON для web и для apps
- Сначала был web, сделали под MVP ФП отдельный язык
- Язык web сделан под потребности ФП, гибкость не важна
- Через год появился язык для apps с поддержкой от команды платформы
- Язык apps гибкий и может использоваться для любых экранов

Третья проблема. Как поддержать 2 языках VDUI одновременно



Разные языки - это разные:

- форматы состояния экрана
- форматы представления одних и тех же компонентов

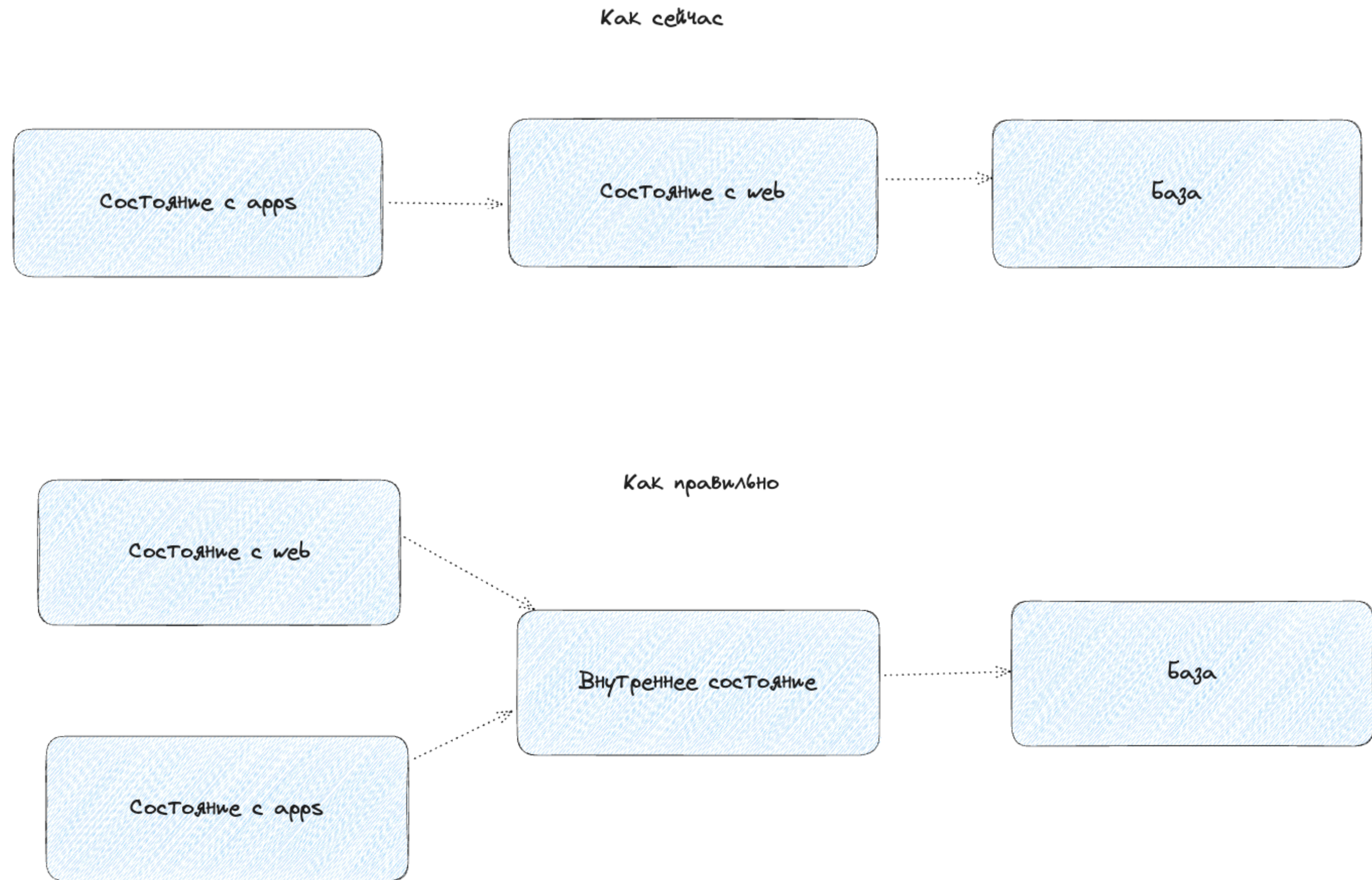
```
'stepState': {  
  'input': [  
    {  
      'code': 'coworkingOperatorName',  
      'value': 'test',  
    },  
  ],  
}
```

это web

```
"newState": {  
  "coworkingOperatorName": "test",  
}
```

это apps

Решение. Состояние



Плюсы

- Общий формат состояния - можно переиспользовать всю логику работы с ним
- Например, правила валидации остаются синхронными для всех платформ

Минусы

- Иногда нужно поддерживать в состоянии web поля которые есть только в apps

Решение. Представление



- Представление в самом низу дерева кода - не влияет на общую логику
- Делаем обертки для BDUI экрана и не завязываемся напрямую на него, используем `TypeVar + Generic`
- **Основной минус** - верстку все еще нужно определять в 2 местах под web и apps отдельно
- Решается поддержкой языка BDUI apps на web

```
StepSchemaT = TypeVar('StepSchemaT')
```

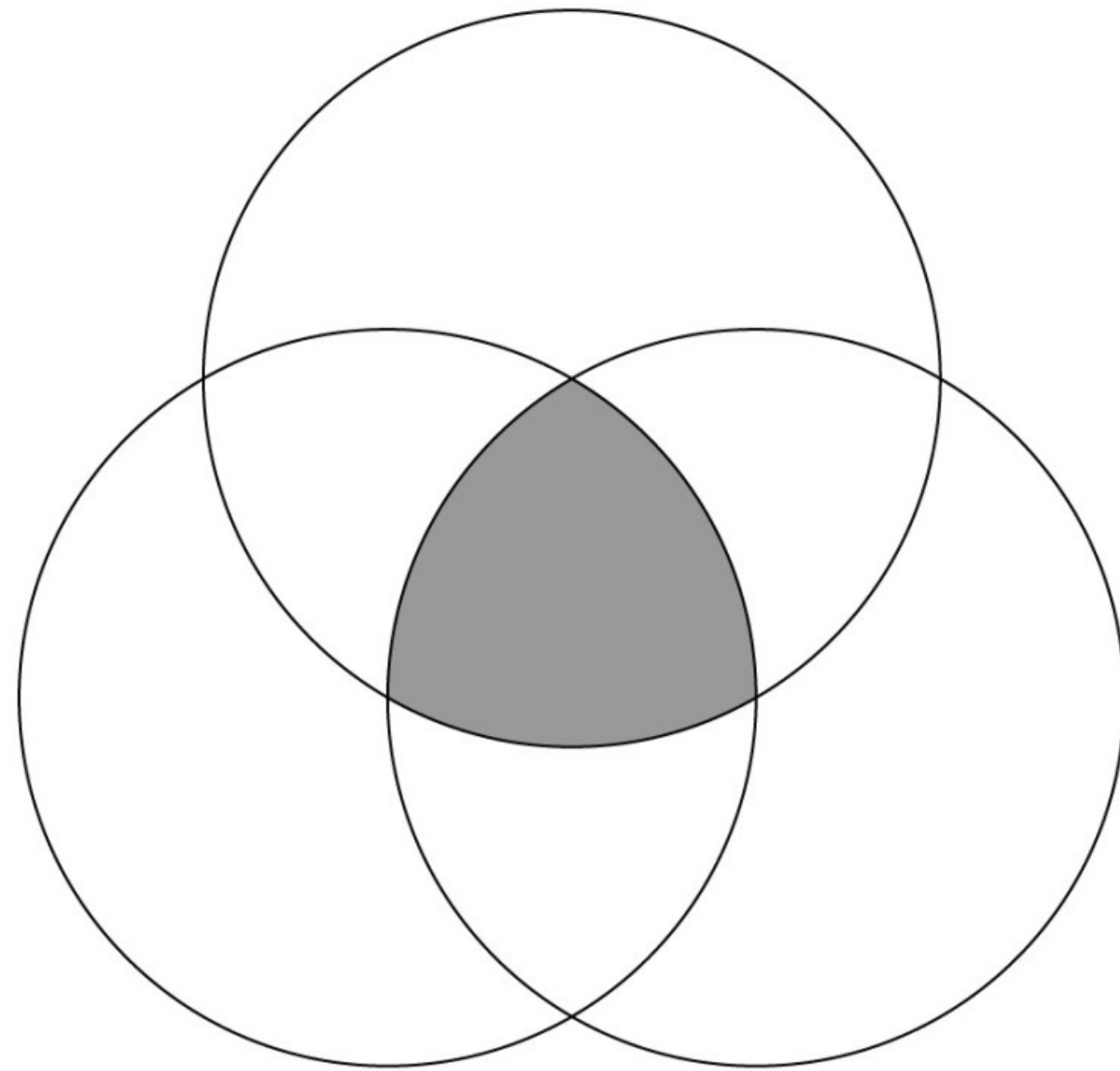
```
class AbstractStepBuilder(ABC, Generic[StepSchemaT]):  
    """Абстрактный класс динамического создания шагов"""
```

```
    def __init__(self, *, step_id: str) -> None:  
        self.step_id: str = step_id  
        self._previous_step_id: str | None = None
```

```
    def build_component_id(self, component_id: str) -> str:  
        return f'{self.step_id}.{component_id}'
```

```
class AppsStepBuilder(AbstractStepBuilder[list[Action]]):
```

```
class WebStepBuilder(AbstractStepBuilder[StepSchema]):
```

Нужен ли VDUI, если у вас только web?

- В большинстве случаев нет, остановитесь на тонком клиенте

Нужен ли VDUI для web, если он у вас будет в apps?

- Для синхронизации логики - да, но смотрите на готовые решения

Можно ли сделать общий язык для web и apps?

- Да, такие решения уже есть
- Усложняется добавление новых компонентов
- Еще дальше от нативного опыта

Четвертая проблема. Дублирование кода



Есть одинаковые:

- экраны в 1 сервисе
- поля на разных экранах

Похожие экраны



Продажа квартиры

Описание



Опишите подробнее

Когда сделан ремонт, что есть рядом: метро, ТЦ, аптеки. Только не вставляйте ссылки и рекламу!

 Нужна помощь?

Описание квартиры

Заголовок

Просторная видовая двушка у парка

0/33

Будет виден только при продвижении  **Топ** и  **Премиум**

Описание

Уютная светлая двушка в тихом спальном районе. Окна на красивые цветущие деревья. Свежий ремонт 2020 года делали для себя. Рядом есть детсад, до метро 10 минут пешком, но ходит автобус. Собственник — я, наследников нет, соседи тихие. Можно заезжать и жить!

0/3000

 Придумать описание за меня

Сохраняем все заполненные поля

[Назад](#)

[Дальше](#)

Похожие экраны



Продажа комнаты или доли

Описание



Опишите подробнее

Когда сделан ремонт, что есть рядом: метро, ТЦ, аптеки. Только не вставляйте ссылки и рекламу!

 Нужна помощь?

Описание квартиры и комнаты

Заголовок

Комната 25 м² с балконом

0/33

Будет виден только при продвижении  **Топ** и  **Премиум**

Описание

Просторная изолированная комната в 2-комнатной квартире. 5 минут от метро, хорошие соседи, проблем нет. Можно жить, можно сдавать.

0/3000

Сохраняем все заполненные поля

[Назад](#)

[Дальше](#)

Решение. Похожие экраны



- Разносим сборку компонентов экрана по отдельным методам - проще ориентироваться
- Лучше принять один формат сборщиков и придерживаться для всех экранов и ответвлений - другим командам сложнее ошибиться.

```
ContainerBuilder(  
    rows = [  
        self._build_desktop_category_title_desktop_row(params),  
        self._build_mobile_category_title_row(),  
        self._build_desktop_address_row(state),  
        self._build_mobile_address_row(state),  
        self._build_mobile_status_label(params),  
    ],  
    view=ContainerView.unframed,  
),  
ContainerBuilder(  
    rows = [  
        self._build_desktop_publish_terms_navigation_row(context),  
        self._build_mobile_publish_terms_navigation_row(context),  
    ],  
    view=ContainerView.standard,  
),
```

Решение. Похожие экраны



- Переиспользуем через наследование
- Переопределяем методы и свойства
- Не нужно дублировать весь экран, если изменения минимальные

```
class Builder(DesktopStepBuilder):
```

```
    @property
    def sidebar_image_link(self) -> str:
        return 'https://cdn.cian.site/thinking.svg'
```

```
class Builder(ResidentialDescriptionBuilder):
```

```
    @property
    def sidebar_image_link(self) -> str:
        return 'https://cdn.cian.site/oncall.svg'
```

Решение. Не все проблемы требуют решения



- Больше переиспользований - сильнее **coupling**
- Изменения в 1 экране могут задеть экраны другой команды
- Контролируемое дублирование в рамках 1 сервиса не такая большая проблема
- Есть IDE и поиск по коду



Главный вопрос

Нужно ли backend разработчикам верстать?



Нужно...

- Но они не одни!
- Есть накопленная база примеров!
- И клиентские разработчики
- Но понимать основу верстки все равно нужно...
- Чтобы общаться на одном языке
- Будущее - конструктор с возможностью двигать компоненты

```
@dataclass(slots=True)
class FlexProperties:
    """Свойства компонента в контейнере"""

    align_self: AlignSelfType | None = None
    """Переопределение выравнивания внутри контейнера"""
    basis: str | None = None
    """Размер элемента по умолчанию до распределения пространства"""
    grow: int | None = None
    """Коэффициент расширения компонента. Положительное число."""
    shrink: int | None = None
    """Коэффициент сжатия компонента"""
```




1. Предыстория

2. Техническая сложность

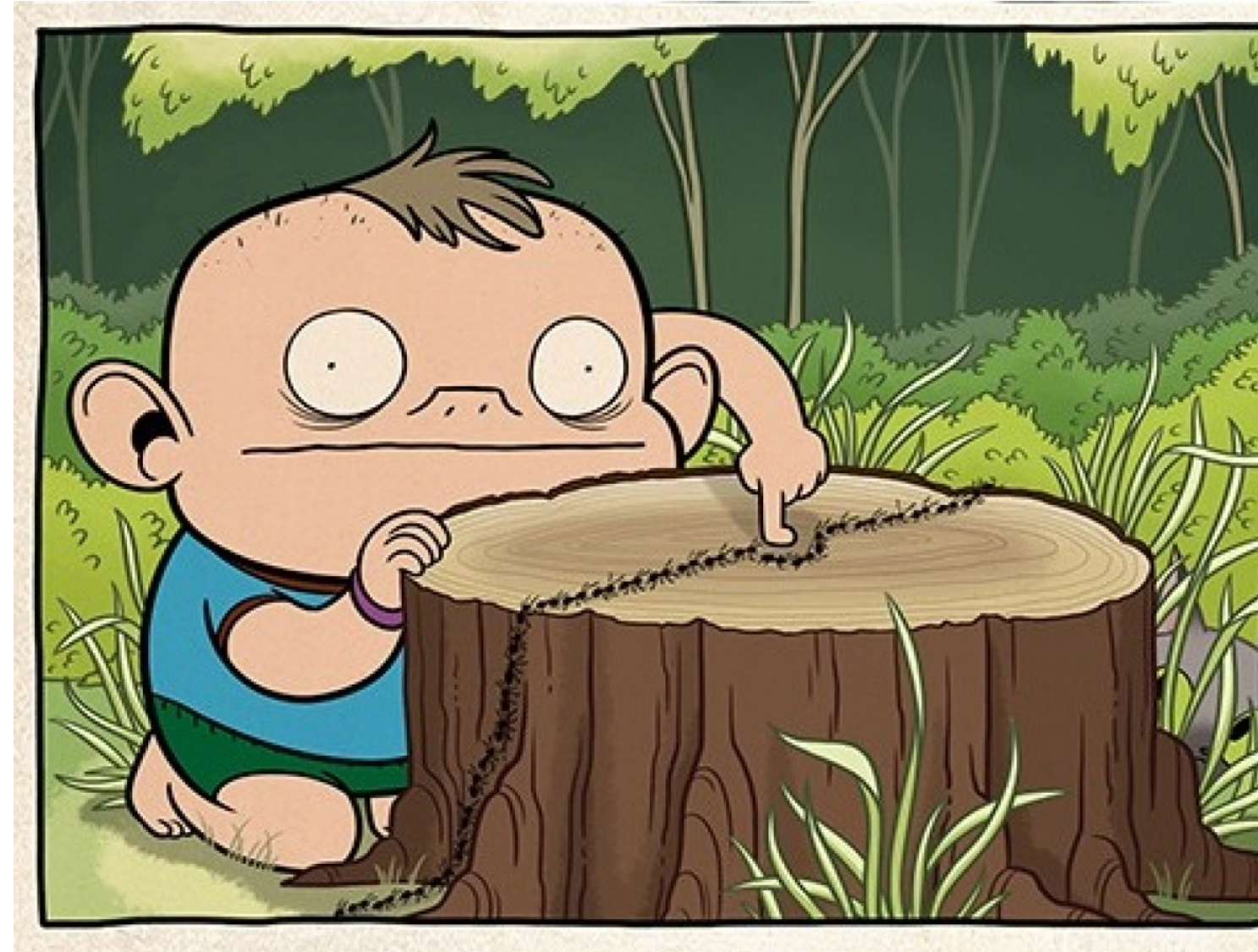
3. Как тестировать?

4. Продуктовая сложность

Используем все возможности



- Надеемся на клиентов
- Конструктор
- Функциональные тесты
- UI Тесты
- JS выражения?



Конструктор



- Проверяем верстку вручную
- Быстрый цикл обратной связи
- Не нужно перекачивать задачу
- Есть для web и apps

```
"id": "featuresStep.changesSavedInfo2.textElement1",  
"type": "text"  
},  
],  
"id": "featuresStep.changesSavedInfo2",  
"margin": {  
  "bottom": 24,  
  "top": 16  
},  
"source": {  
  "text": [  
    {  
      "breakAfter": false,  
      "color": "gray40_100",  
      "id": "featuresStep.changesSavedInfo2.textElement1",  
      "style": "UIText1",  
      "text": "Сохраняем все заполненные поля"  
    }  
  ]  
}  
}  
}
```

Prettify Render

↑

Функциональные тесты



- Grey box тестирование
- Поднимаем сервис и зависимости в Docker,
- mountebank для мока соседей
- Пишем на каждый экран
- Гарантируют, что вероятность случайно сломать экран минимальная
- Не гарантируют рабочий JSON - нужно проверить через конструктор
- Есть возможность автоматического обновления

```
==== 3524 passed, 3 skipped in 146.12s (0:02:26) ====
```

```
pytest --update-step-fixtures tests functional/web/test v1 goto.py
```

UI Тесты



- Помогают очень сильно
- Робот ходит как клиент по реальным экранам

web

- Пишут тестировщики
- Прогоняются автоматически при релизе бэкенда

apps

- Пишут разработчики
- Не прогоняются автоматически при релизе бэкенда

JS код выражений



web

- Валидация
- Условия скрытия компонента

apps

- Валидация
- Форматирование
- Вся работа со стейтом

Тестирование?

- Делаем выражения максимально простыми и проверяем вручную
- PythonMonkey
- <https://github.com/Distributive-Network/PythonMonkey>

```
def build_action_to_expand_tip(
    *,
    icon_field_name: str,
    flag_field_name: str,
    up_arrow_image: str,
    down_arrow_image: str,
) -> JsAction:
    js_code = r"""
function main(globalState, localState, args) {{
    let newState = globalState;
    if (newState['{flag_field_name}'] == false) {{
        newState['{flag_field_name}'] = true;
        newState['{icon_field_name}'] = {{ "url" : "{up_arrow_image}" }};
    }} else {{
        newState['{flag_field_name}'] = false;
        newState['{icon_field_name}'] = {{ "url" : "{down_arrow_image}" }};
    }}
    return newState
}}
"""
    return JsAction(
        id=JsAction.build_id(flag_field_name, 'expandTip'),
        action_data=JsActionData(
            target_state=TargetState.global_state,
            js_code=clear_js_code(js_code),
            animated=True,
        ),
    )
```




1. Предыстория

2. Техническая сложность

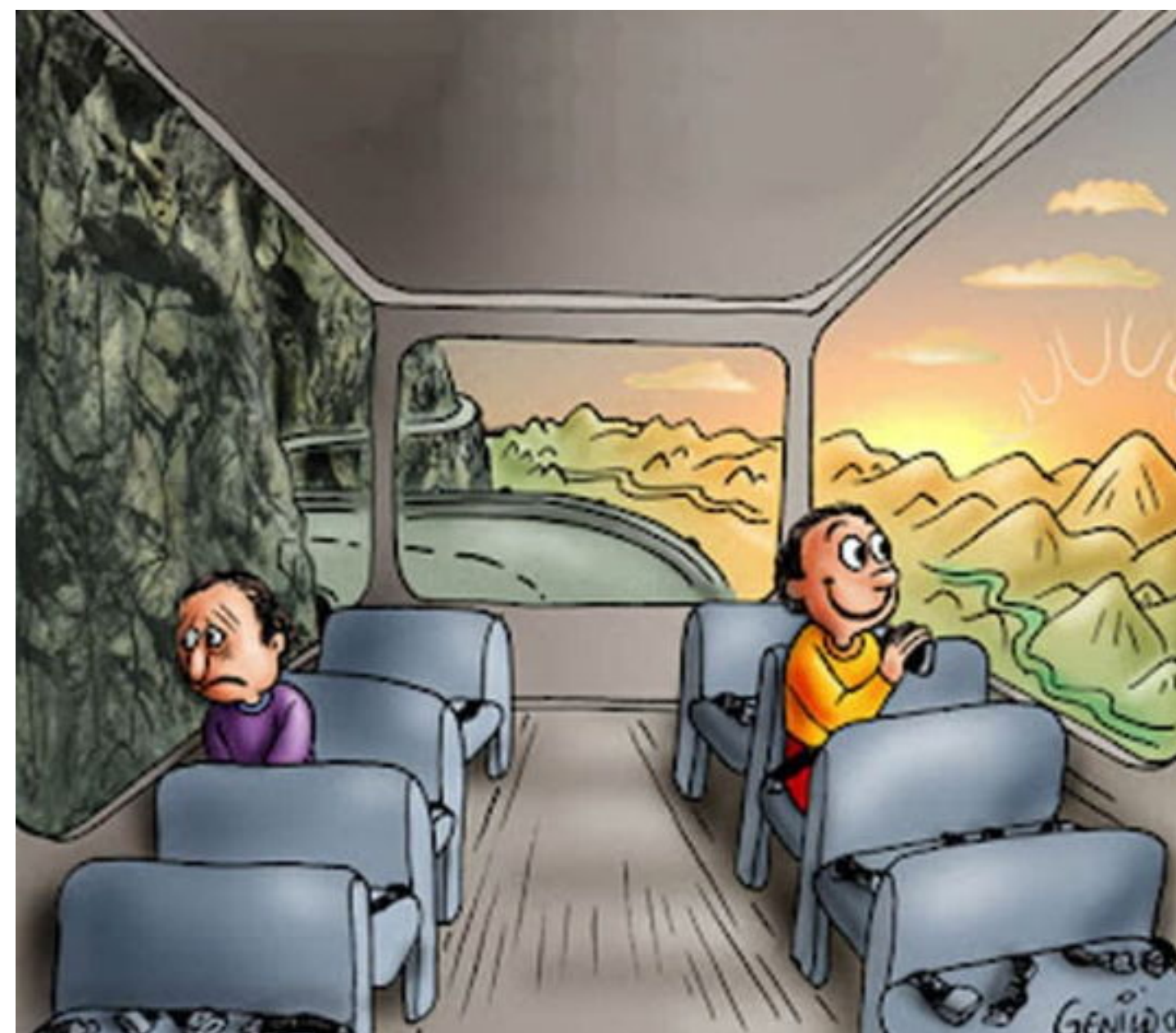
3. Как тестировать

4. Продуктовая сложность

Крупный продукт = сложный



📍 995 Branches



циан | Аренда | Продажа | Новостройки | Строительство | Коммерческая | Ипотека | Сервисы | ПИК | + Разместить объявление

Продажа квартиры

Расположение

🔍 Давайте начнём!

Напишите без сокращений улицу и номер дома. Выберите полный адрес из подсказки или подвиньте метку на карте.

🗨️ Нужна помощь?

Введите адрес

Россия, Матроса Железняка 30

Редактировать адрес можно в течение 2 дней после публикации

Коптево — 9 мин. пешком | Лихоборы — 5 мин. на транспорте | Красный Балтиец — 5 мин. на транспорте

Этаж | Этажей в доме



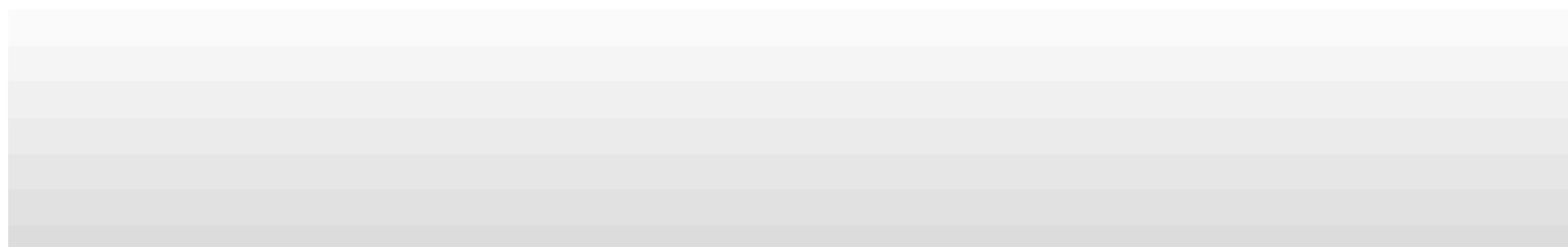
История: Первые результаты

366

задач на бэкенде

132

задачи на фронтенде



https://www.youtube.com/watch?v=fFgKhfE0_Uo



- Бэкендер - мейнтейнер сложности BDUI языка
- Контролируем поток изменений
- **Если много компонентов языка:**
 - Замедляются проекты (релизный цикл apps)
 - Сложнее тестировать
 - Усложняется код (ветвистость для версий)
 - Теряется лаконичность

Упрощаем продукт



- Проще продукт - проще язык (меньше компонентов)
- Не все сценарии возможно реализовать легко
 - и это нормально
- Определяем критичные фичи, которые влияют на успех
- Сопоставляем значимость и трудозатраты
- Учим дизайнера и продакта жить с новым подходом

Сохраняем эффективность



- Задачи до 2 дней - после в master, меньше конфликтов
- **Feature toggle** - лучший друг
- Тестируем на всех платформах - можно сломать всех клиентов разом
- **Devtesting** задач, QA для проектов
- Много быстрых функциональных тестов

^ cian
dream
team

Q&A