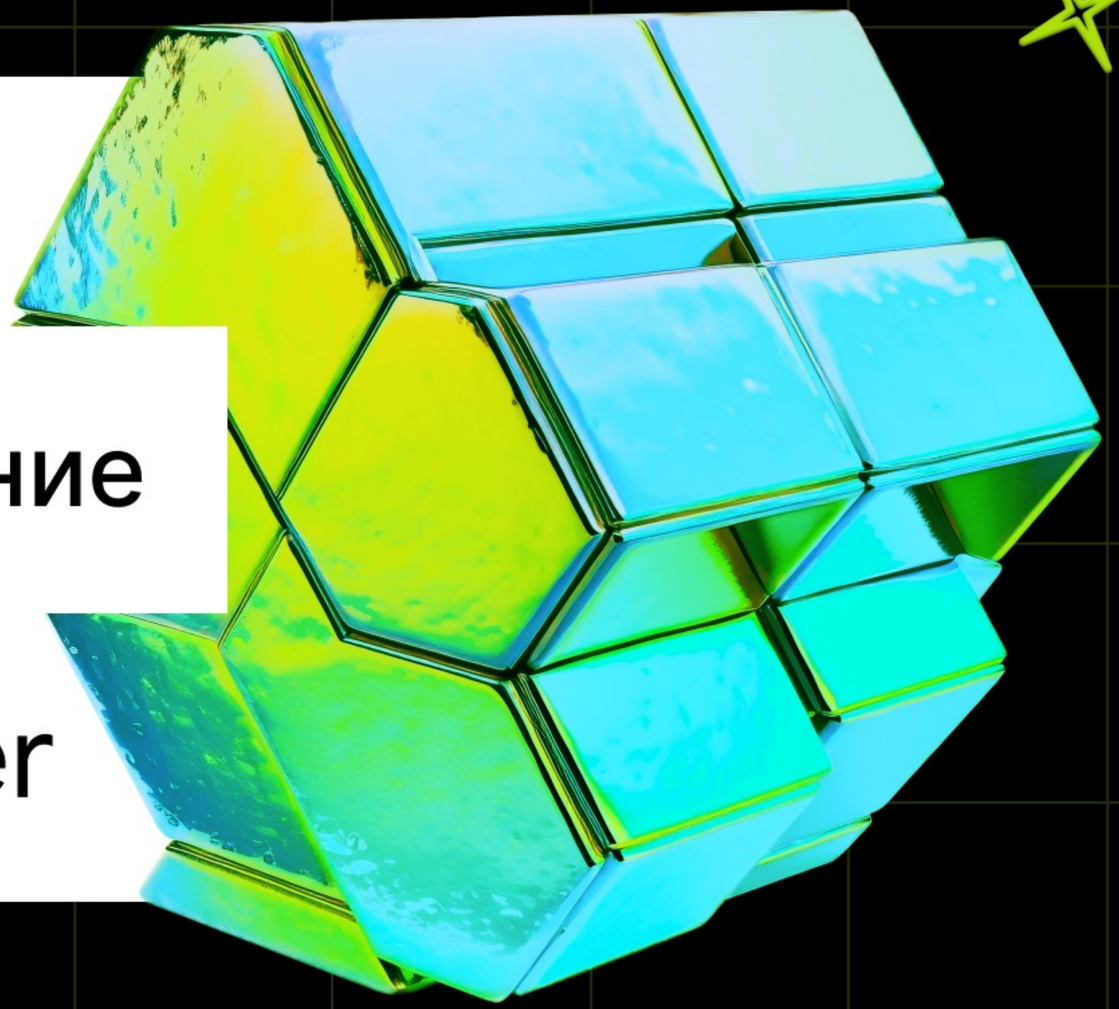
Многомодульное приложение на Flutter

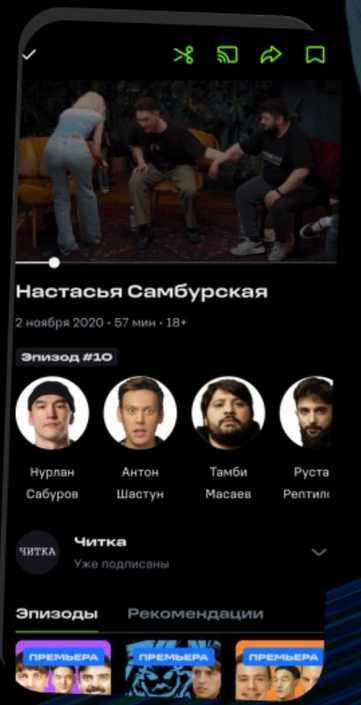


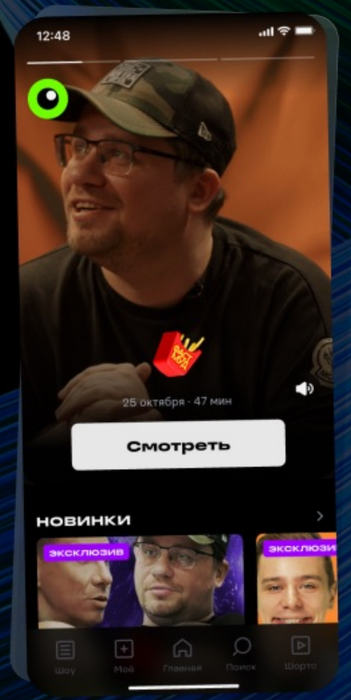
Кирилл Адещенко

Head of Mobile development в РСХБ

tg: @kaparray

kaparray@gmail.com





Medium Quality - The hole

tg: @kaparray

kaparray@gmail.com

Кирилл Адещенко

Head of Mobile development в РСХБ

Arenum Games
(первое крупное коммерческое приложение на Flutter в СНГ) и Yappy

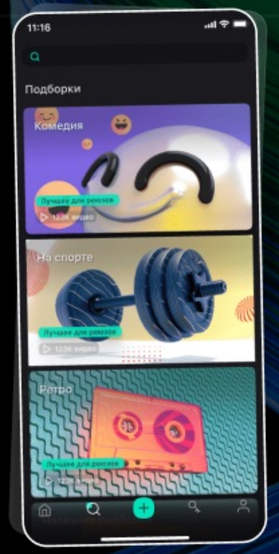
Medium Quality - The hole

Кирилл Адещенко

Head of Mobile development в РСХБ

tg: @kaparray

kaparray@gmail.com



Arenum Games
(первое крупное коммерческое приложение на Flutter в СНГ) и Yappy

Medium Quality - The hole

tg: @kaparray

kaparray@gmail.com

Кирилл Адещенко

Head of Mobile development в РСХБ

Arenum Games
(первое крупное коммерческое приложение на Flutter в СНГ) и Yappy

Medium Quality - The hole

Кирилл Адещенко

Head of Mobile development в РСХБ

tg: @kaparray

kaparray@gmail.com

ДБО ЮЛ и ФЛ

Arenum Games
(первое крупное коммерческое приложение на Flutter в СНГ) и Yappy

Medium Quality - The hole

Кирилл Адещенко

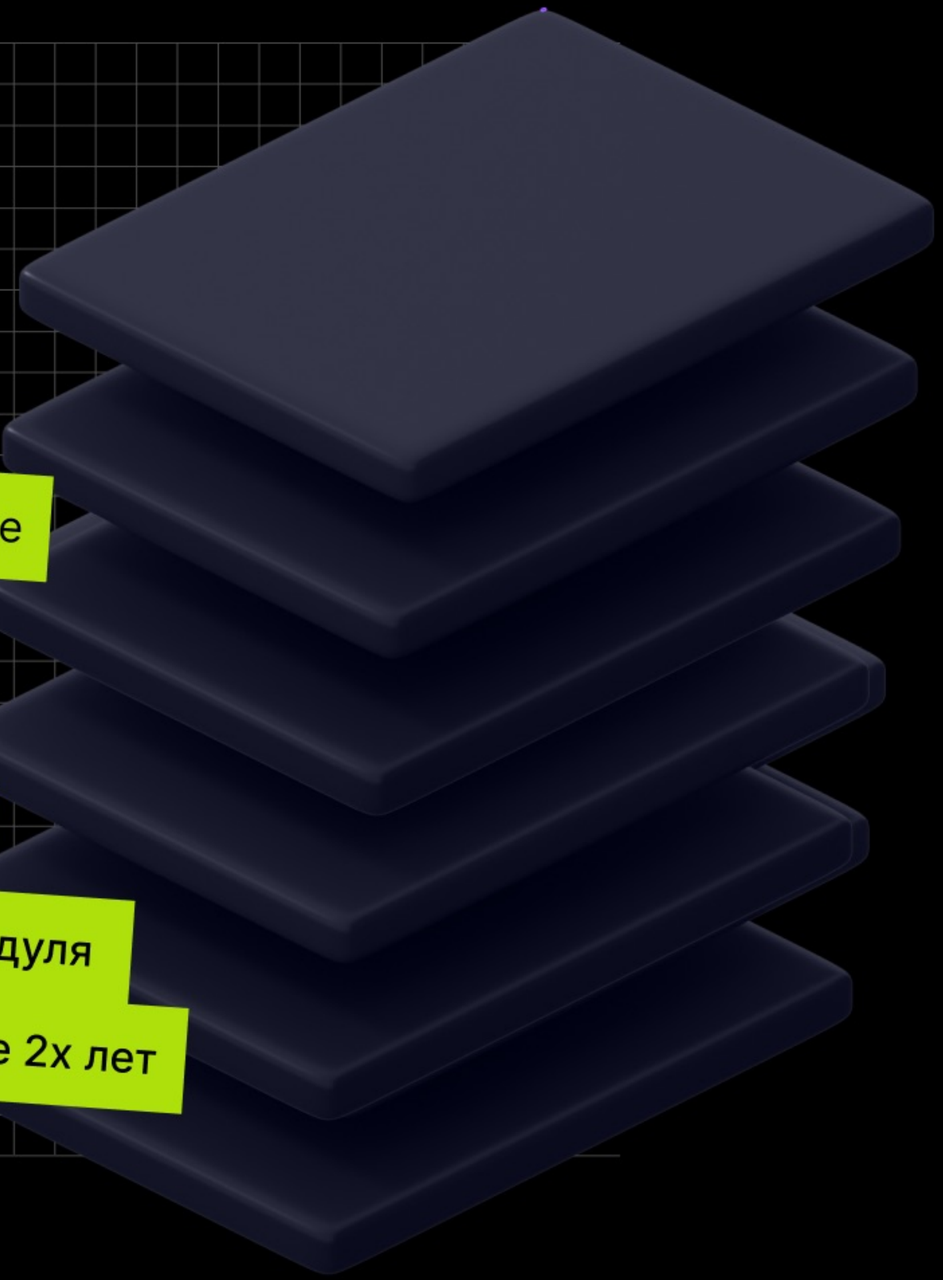
Head of Mobile development в РСХБ

tg: @kaparray

kaparray@gmail.com

План доклада

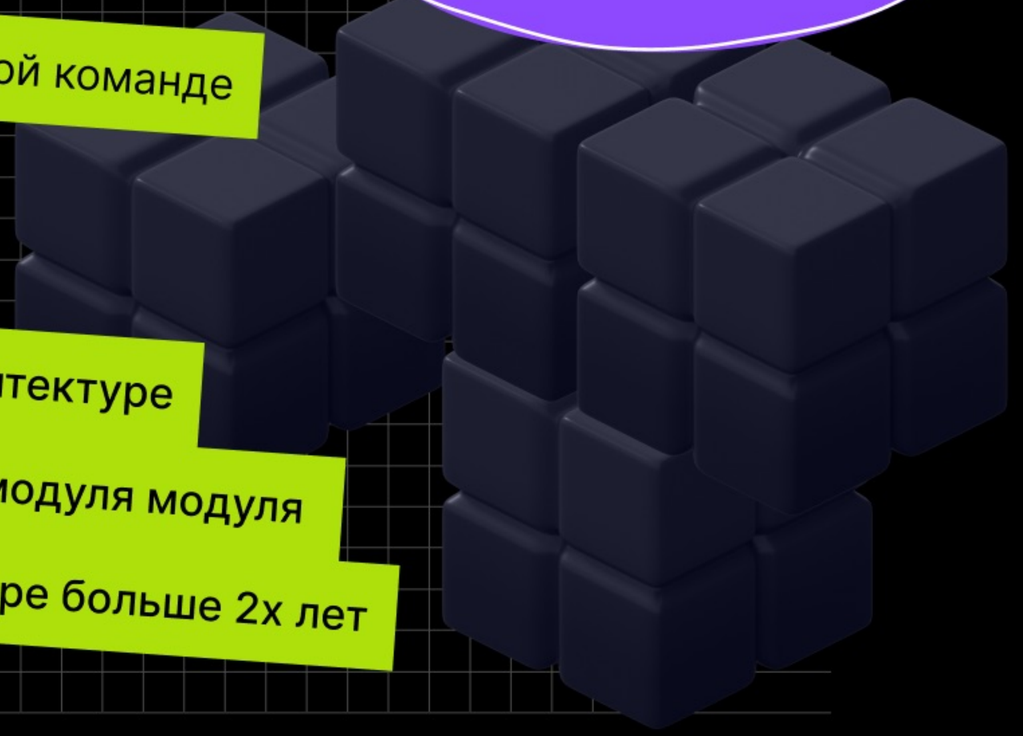
1. Проблемы монолитного Flutter приложения в большой команде
2. Описание основных подходов решения проблем
3. Общая картина модульной архитектуры в Flutter
4. Имплементация каждого слоя в модульной архитектуре
5. Описание структуры модуля на примере продуктового модуля модуля
6. Результаты работы в проде на модульной архитектуре больше 2х лет

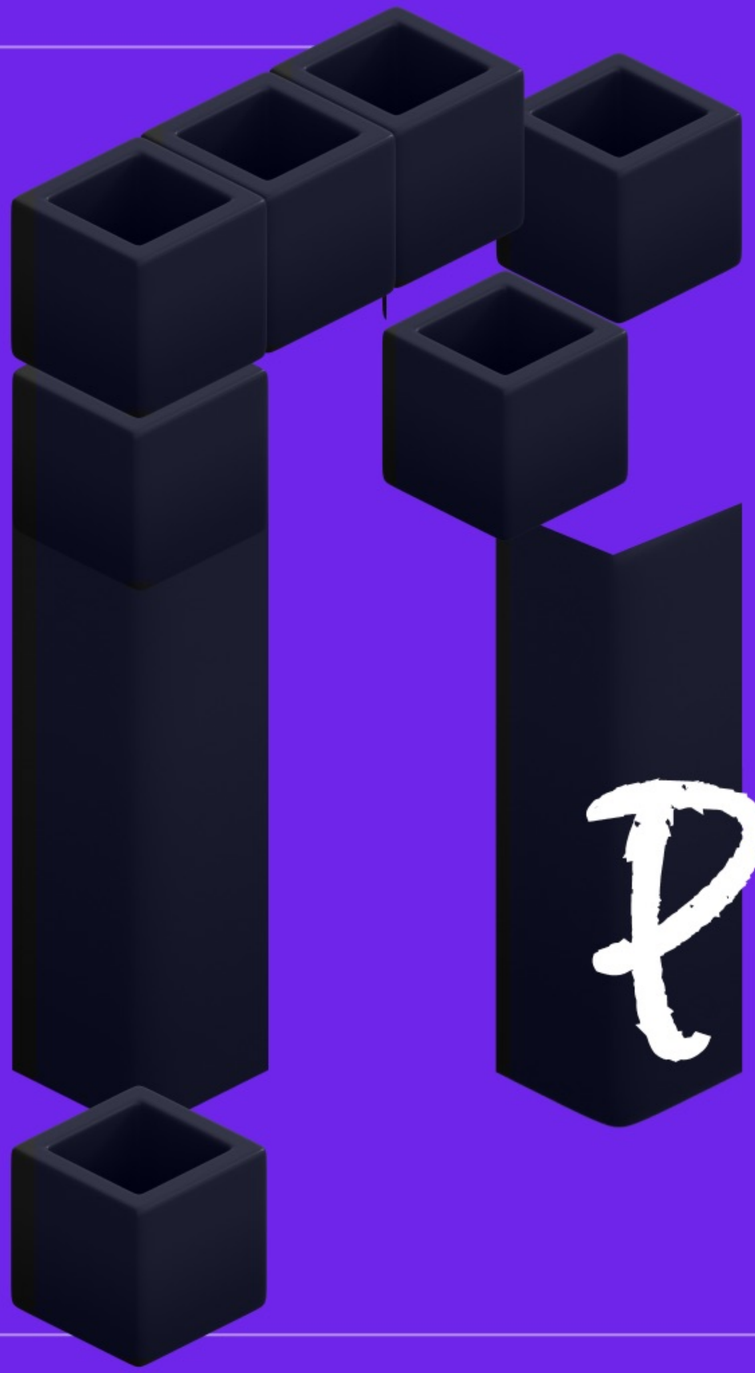


План доклада

1. Проблемы монолитного Flutter приложения в большой команде
2. Описание основных подходов решения проблем
3. Общая картина модульной архитектуры в Flutter
4. Имплементация каждого слоя в модульной архитектуре
5. Описание структуры модуля на примере продуктового модуля модуля
6. Результаты работы в проде на модульной архитектуре больше 2х лет

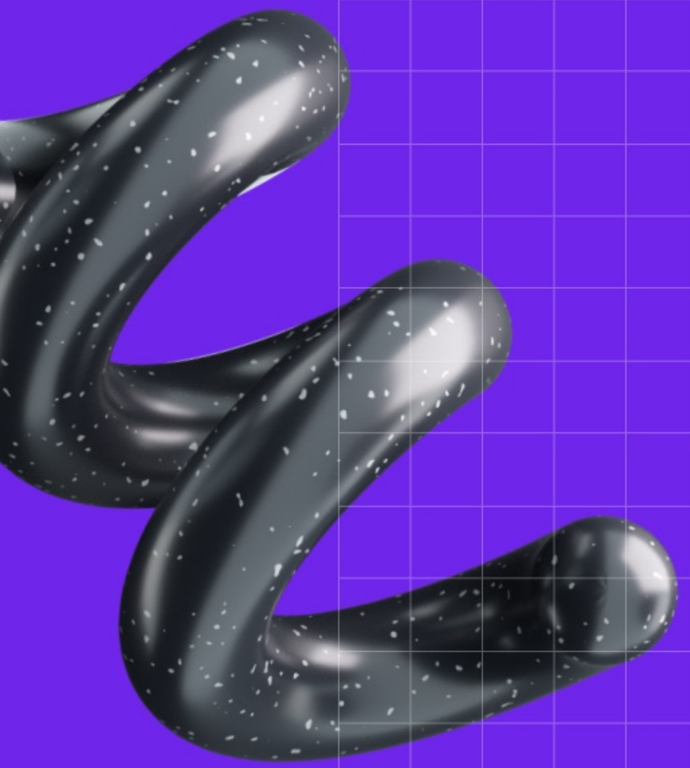
Мы пошли по пути *многомодульности*





Проблемы

Контекст



01

У нас матричная структура с 20+ командами разбитые на продукты

02

Команды независимы и автономны

03

Возможность быстро итерировать над продуктом и не блокировать друг друга

04

Интеграция приложения со старыми системами банка

МОНОЛИТ VS МОДУЛЬНОСТЬ

1. Очередь на вливание
2. Неоптимальность рабочего времени
3. Синхронизация команд
4. Размытие ответственности
5. Связанность

1.Переиспользование кода
2.Прозрачность (зависимости, потребители, API, etc)

1.Сложность архитектуры
2.Интеграция модулей между собой
3.Усложненная отладка
4.Увеличенное время разработки на старте
5.Отсутствие прозрачности

1.Разделение обязанностей
2.Легкость масштабирования и поддержки
3.Параллелизация разработки

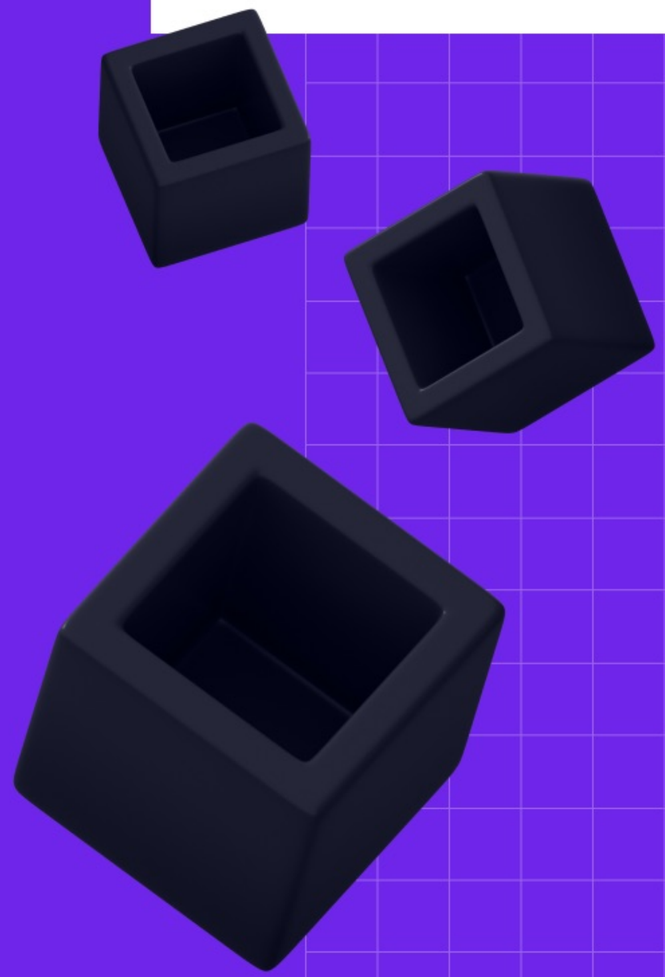
Почему

выбрали модульность

1. Риск внесения ошибок в критически важные части проекта
2. Каждый продукт
   - Разрабатывается независимо
   - Может переиспользоваться в других приложениях
3. Продуктовым командам не нужно разрабатывать и думать про инфраструктуру

Описание основных подходов решения проблем

Principles



01

Слабосвязность

02

Модуль - это независимая диница проекта.

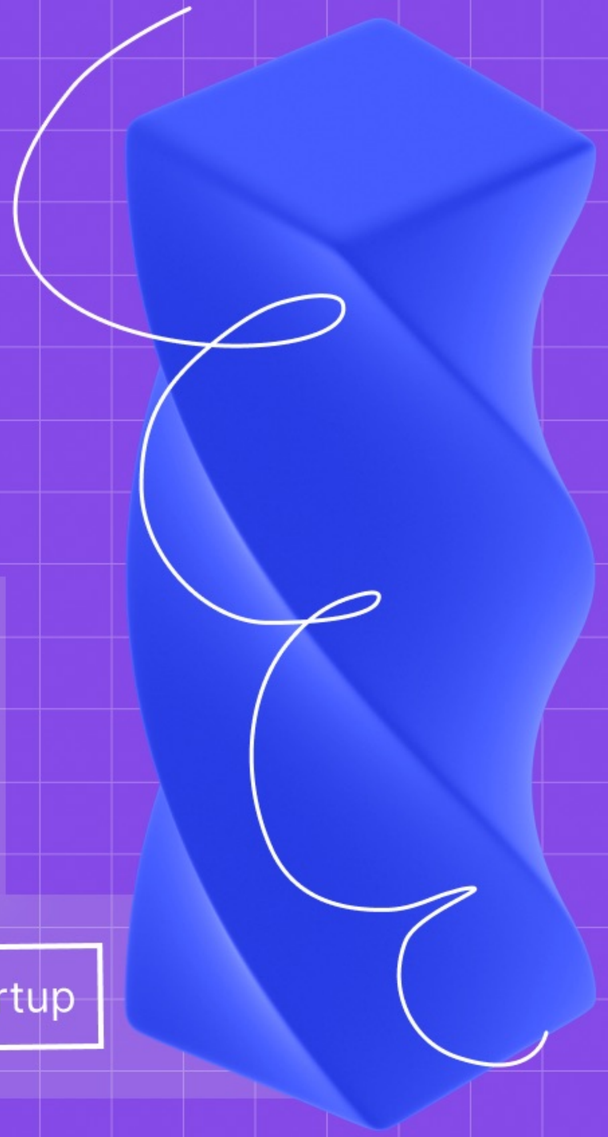
03

Модули должны быть разделены на продуктовые фичи.

04

Применение инфраструктурных пакетов.

Flutter Cross Functional



1. Business

Ui and Business logic

Onboarding

Personalization

Payroll

Communication

Filters

Payments

2. Infrastrcture

Services and packages

Networking

Routing

Storage

Analytics

Biometrics

Lifecycle

3. Foundation

Services and packages

Architecture

Di

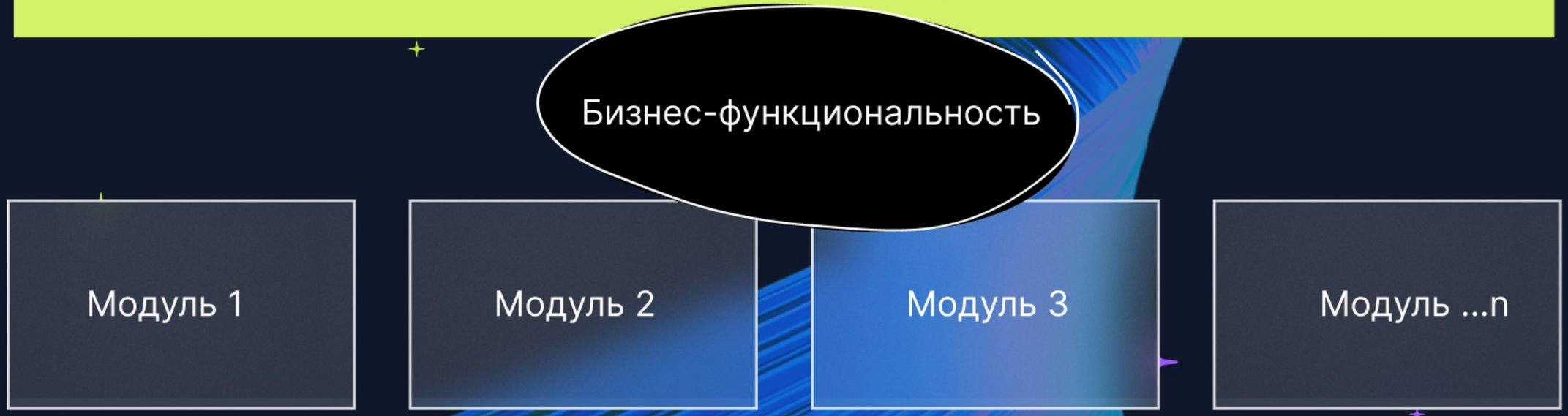
Startup

4. Tooling

Business

Бизнес-слой - это набор бизнес-логики, которые описывают пользовательское поведение и имеют решающее значение для функционирования приложения.

В Flutter Cross Functional они описываются при помощи модулей.



Infrastructure

Это расширяемый набор инструментов выполняющий прикладные задачи. На этом уровне находятся все компоненты ввода-вывода: база данных, фреймворки, устройства и т.д.

Устройство модуля внутри

```
class OnboardingModule implements Module {
  Future<void> start(IDiContainer di) async {
    final IAccountsApi accountsApi = AccountsApi(httpProvider: di.getNetworkClientProvider());
    final IAccountsRepository accountsRepository = AccountsRepository(accountsApi: accountsApi);

    final homeScreen = HomeScreen(
      accountsRepository: accountsRepository,
      router: di.getRouter(),
      notificationService: di.getNotificationService(),
    );

    di.getRoutingConfig().addRoutes(
      [
        PageRoutingEntry(
          name: 'home',
          pageBuilder: homeScreen.view,
        ),
      ],
    );
  }
}
```

Foundation

1. Application

Application определяет набор модулей, отвечает за конфигурацию сборки и запуск приложения.
Он является точкой входа в приложения, в нем содержится функция main. Application не содержит бизнес-логики, он управляет тем, какие модули будут подключены в проект и отвечает за то, чтобы инициировать их запуск.

2. Core

Core-пакет предоставляет набор абстракций, определяющих внешний API-модулей и структуру модулей. Core также предоставляет абстракции для разделения бизнес-логики и представления в рамках экрана, посредством MVVM паттерна.

3. Startup

Startup отвечает за конфигурирование зависимостей. Он выполняет следующие операции:

- создает DI-контейнер;
- регистрирует общие зависимости;
- выполняет стартовую конфигурацию;
- инициирует конфигурирование модулей;

Startup помогает запускать модули по отдельности из example, не дублируя код по конфигурированию зависимостей.

4. DI

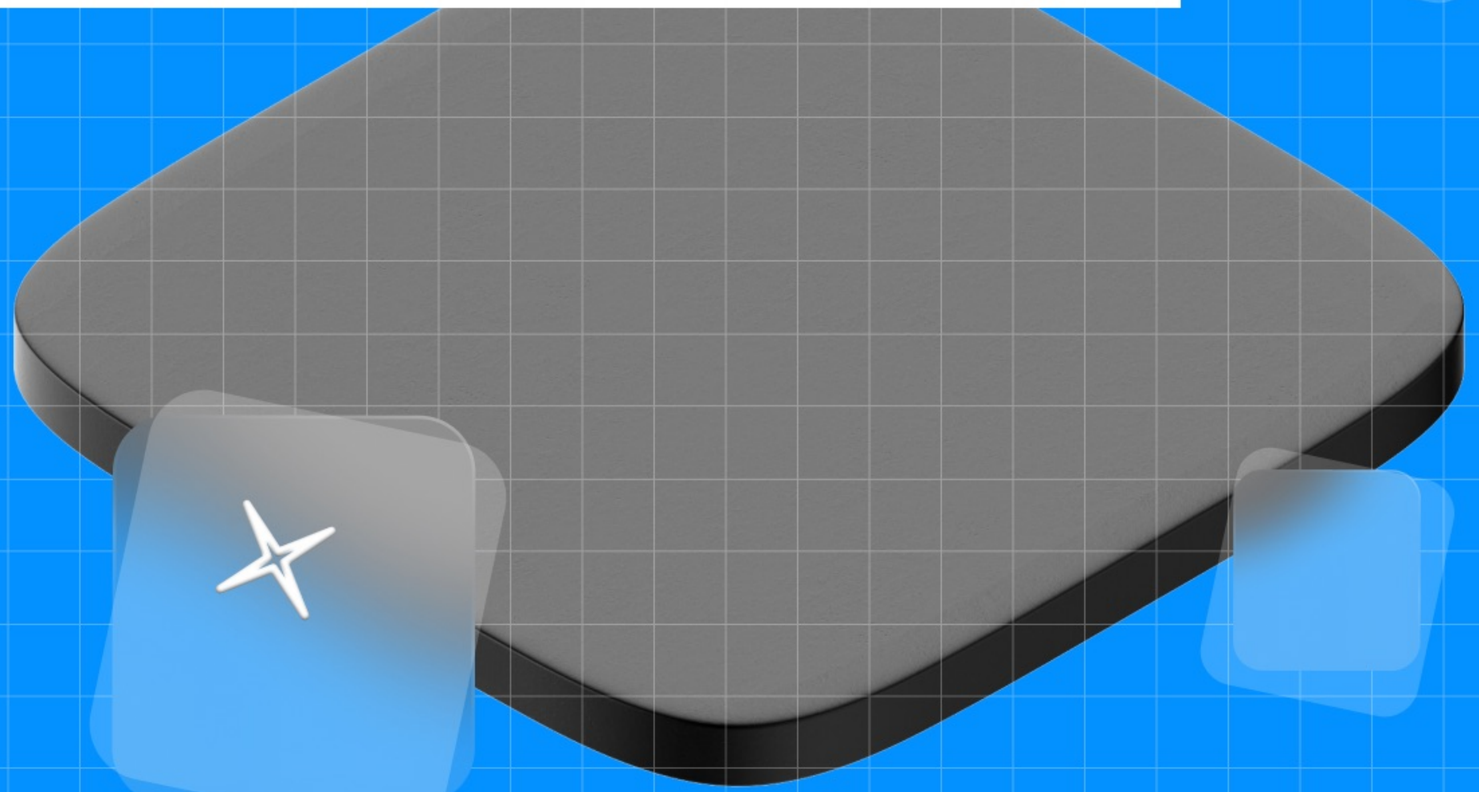
Этот пакет является простым DI-контейнером, который предоставляет статичный API для получения зависимостей.

Tooling

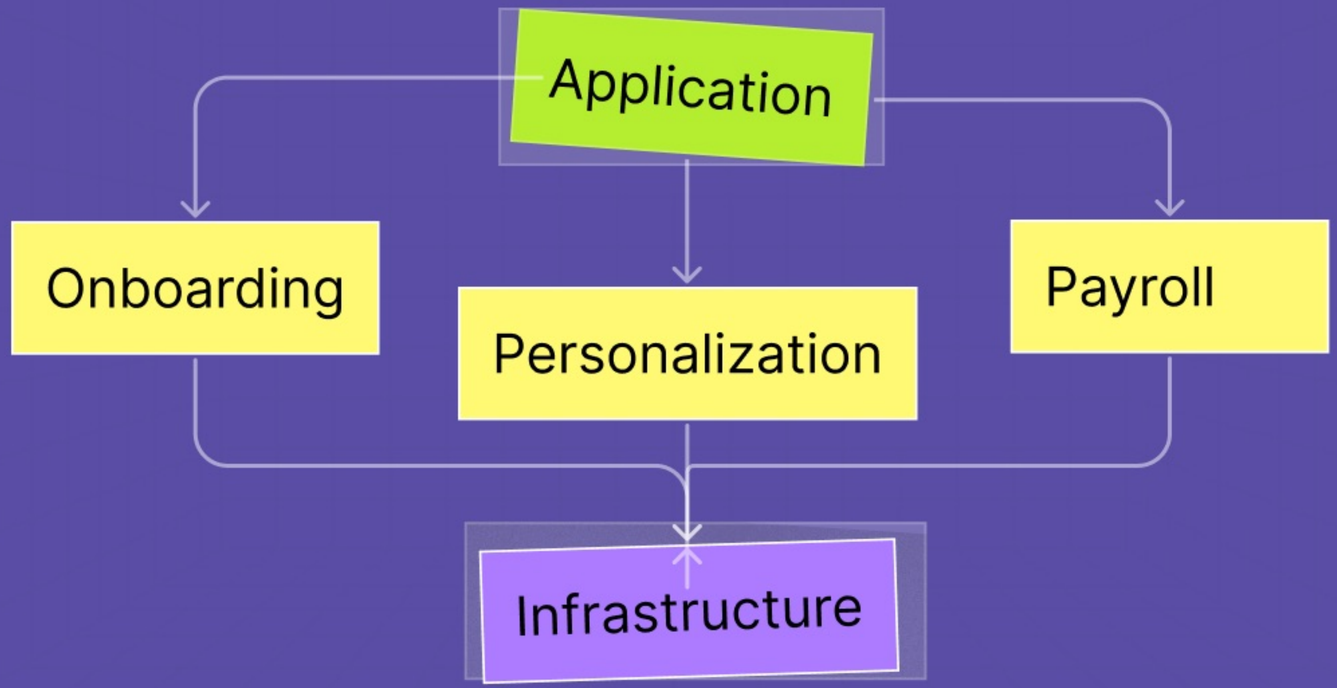
Tooling слой предоставляет пакеты, помогающие при разработке.

Как правило это различные генераторы кода и инструменты, упрощающие разработку.

Описание структуры модуля на примере продуктового



Устройство модуля



внутри

Устройство модуля

```dart
class OnboardingModule implements Module {
  Future<void> start(IDiContainer di) async {
    final IAccountsApi accountsApi = AccountsApi(httpProvider: di.getNetworkClientProvider());
    final IAccountsRepository accountsRepository = AccountsRepository(accountsApi: accountsApi);

    final homeScreen = HomeScreen(
      accountsRepository: accountsRepository,
      router: di.getRouter(),
      notificationService: di.getNotificationService(),
    );

    di.getRoutingConfig().addRoutes(
      [
        PageRoutingEntry(
          name: 'home',
          pageBuilder: homeScreen.view,
        ),
      ],
    );
  }
}
```

внутри

Устройство модуля

```
class OnboardingModule implements Module {
  Future<void> start(IDiContainer di) async {
    final IAccountsApi accountsApi = AccountsApi(httpProvider: di.getNetworkClientProvider());
    final IAccountsRepository accountsRepository = AccountsRepository(accountsApi: accountsApi);

    final homeScreen = HomeScreen(
      accountsRepository: accountsRepository,
      router: di.getRouter(),
      notificationService: di.getNotificationService(),
    );

    di.getRoutingConfig().addRoutes(
      [
        PageRoutingEntry(
          name: 'home',
          pageBuilder: homeScreen.view,
        ),
      ],
    );
  }
}
```

изнутри

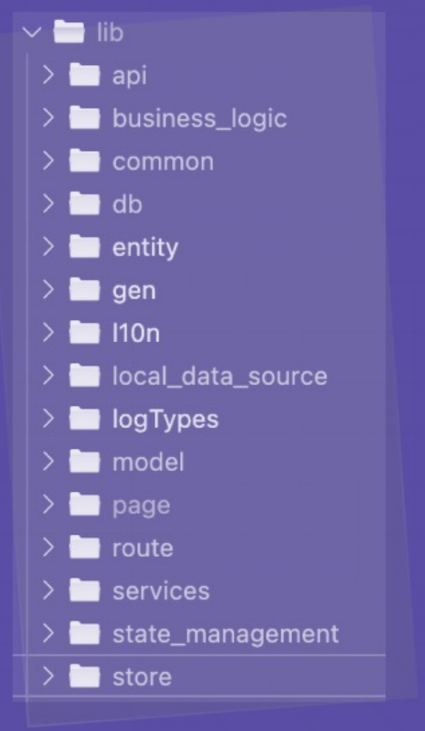
Устройство модуля

```dart
class OnboardingModule implements Module {
  Future<void> start(IDiContainer di) async {
    final IAccountsApi accountsApi = AccountsApi(httpProvider: di.getNetworkClientProvider());
    final IAccountsRepository accountsRepository = AccountsRepository(accountsApi: accountsApi);

    final homeScreen = HomeScreen(
      accountsRepository: accountsRepository,
      router: di.getRouter(),
      notificationService: di.getNotificationService(),
    );

    di.getRoutingConfig().addRoutes(
      [
        PageRoutingEntry(
          name: 'home',
          pageBuilder: homeScreen.view,
        ),
      ],
    );
  }
}
```

внутри

Устройство модуля

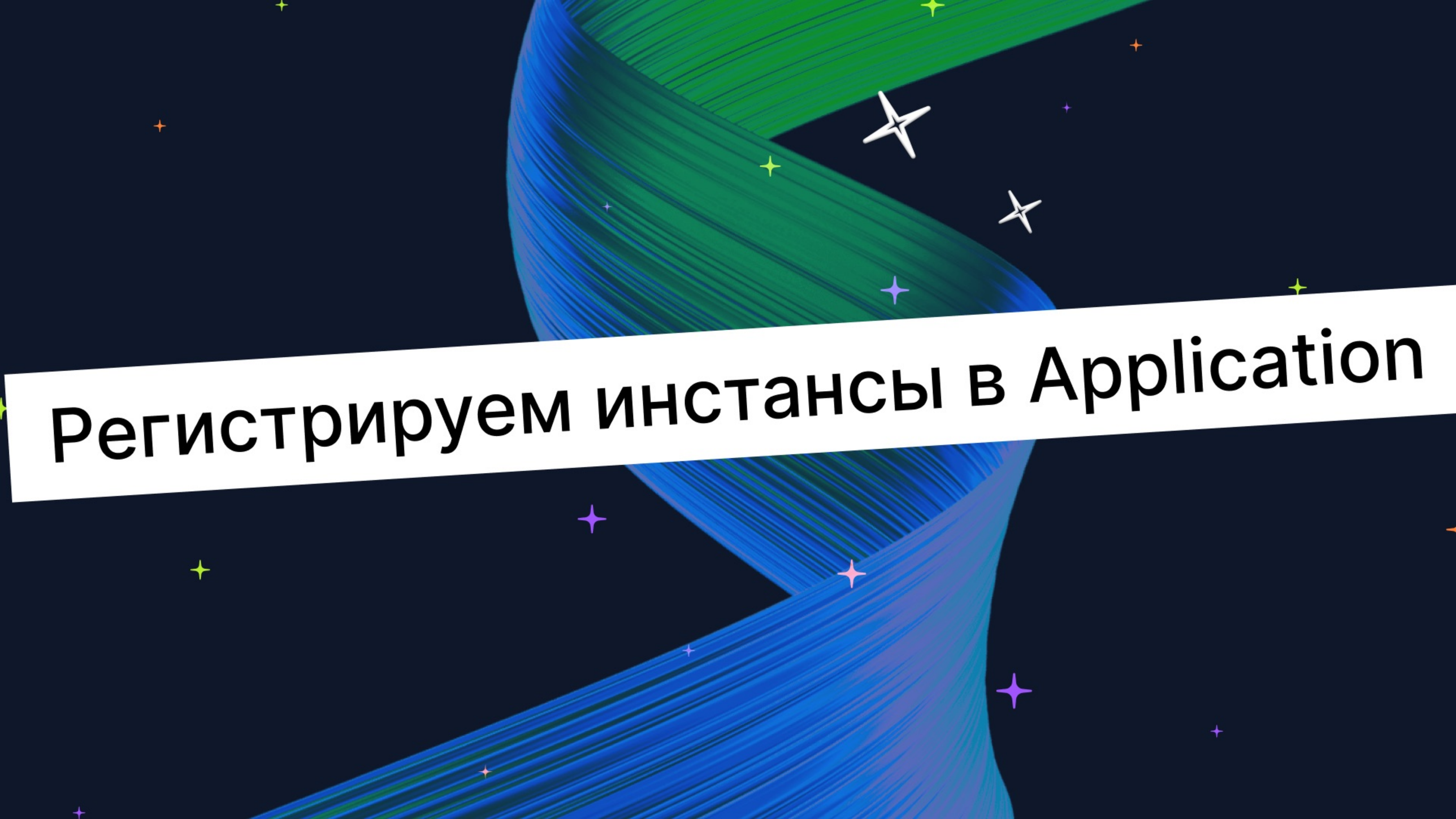
- screens
- repositories
- entities
- api
- store
- l10n



внутри

Сборка проекта

```
onboarding_module:
  git: https://gitlab.rshbdev.ru/rshbintech/crft/dboul/ndbo/mobile/modules/onboarding.git
  version: v0.1.17
  class_name: OnboardingModule
filters_module:
  git: https://gitlab.rshbdev.ru/rshbintech/crft/dboul/ndbo/mobile/modules/filters_module.git
  version: v0.2.4
  class_name: FiltersModule
cs_operations_rubles_module:
  git: https://gitlab.rshbdev.ru/rshbintech/crft/dboul/ndbo/mobile/modules/cs_operations_rubles_module.git
  version: v0.2.24
  class_name: CSOperationsRublesModule
app_module:
  class_name: AppModule
personalization_module:
  git: https://gitlab.rshbdev.ru/rshbintech/crft/dboul/ndbo/mobile/modules/personalization.git
  version: v0.1.21
  class_name: PersonalizationModule
debug_tools_module:
  git: https://gitlab.rshbdev.ru/rshbintech/crft/dboul/ndbo/mobile/modules/debug_tools
  version: v0.1.2
  class_name: DebugToolsModule
salary_module:
  git: https://gitlab.rshbdev.ru/rshbintech/crft/dboul/ndbo/mobile/modules/payroll_project
  version: v0.1.3
  class_name: SalaryModule
account_statement:
  git: https://gitlab.rshbdev.ru/rshbintech/crft/dboul/ndbo/mobile/modules/account-statement
  version: v0.1.1
  class_name: AccountStatementModule
pdf_viewer_module:
  git: https://gitlab.rshbdev.ru/rshbintech/crft/dboul/ndbo/mobile/modules/pdf_viewer
  version: v0.0.1
  class_name: PDFViewerModule
```


Регистрируем инстансы в Application

Регистрируем инстансы в Application

```
final List<Module> modules = [
  OnboardingModule(),
  FiltersModule(),
  CSOperationsRublesModule(),
  AppModule(),
  PersonalizationModule(),
  DebugToolsModule(),
  SalaryModule(),
  AccountStatementModule(),
  PDFViewerModule()
];
```

Запуск модулей в Application

Запуск модулей в Application

```
Future<void> main() async {
   final di = await Startup().start(modules);
   runApp();
}
```

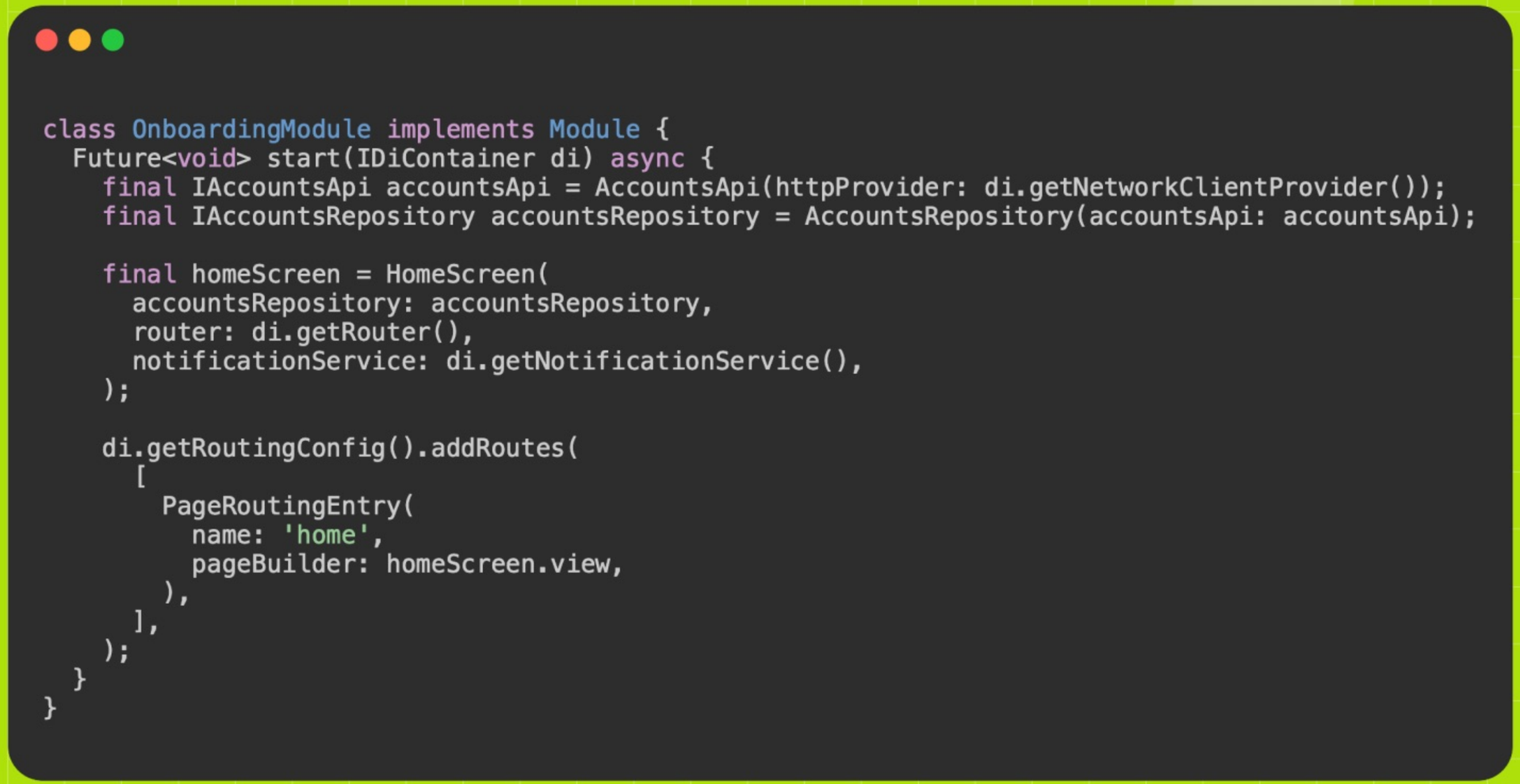

Under the hood

Under the hood

```
await Future.wait(modules.map((module) => module.start(di)));
```

Навигация

Навигация



```
class OnboardingModule implements Module {
  Future<void> start(IDiContainer di) async {
    final IAccountsApi accountsApi = AccountsApi(httpProvider: di.getNetworkClientProvider());
    final IAccountsRepository accountsRepository = AccountsRepository(accountsApi: accountsApi);

    final homeScreen = HomeScreen(
      accountsRepository: accountsRepository,
      router: di.getRouter(),
      notificationService: di.getNotificationService(),
    );

    di.getRoutingConfig().addRoutes(
      [
        PageRoutingEntry(
          name: 'home',
          pageBuilder: homeScreen.view,
        ),
      ],
    );
  }
}
```

Навигация

```
class OnboardingModule implements Module {
  Future<void> start(IDiContainer di) async {
    final IAccountsApi accountsApi = AccountsApi(httpProvider: di.getNetworkClientProvider());
    final IAccountsRepository accountsRepository = AccountsRepository(accountsApi: accountsApi);

    final homeScreen = HomeScreen(
      accountsRepository: accountsRepository,
      router: di.getRouter(),
      notificationService: di.getNotificationService(),
    );

    di.getRoutingConfig().addRoutes(
      [
        PageRoutingEntry(
          name: 'home',
          pageBuilder: homeScreen.view,
        ),
      ],
    );
  }
}
```

Навигация

```
di.getRoutingConfig().addRoutes([
      PageRoutingEntry(
        name: 'payment_info/:payment_id',
        pageBuilder: paymentInfoScreen.view,
      ),
      PageRoutingEntry(
        name: 'create_payment',
        pageBuilder: paymentTypeSelectorScreen.view,
        routeBuilder: ModalBottomSheetRouteBuilder().build,
      ),
      PageRoutingEntry(
        name: 'payment_order',
        pageBuilder: (pathArgs, queryArgs) {
          final configuration = PaymentOrderScreenArgsAdapter.adaptConfiguration(queryArgs);

          return paymentOrderScreen.view(pathArgs, queryArgs, configuration: configuration);
        },
      ),
]);
```

Навигация

```
di.getRoutingConfig().addRoutes([
      PageRoutingEntry(
        name: 'payment_info/:payment_id',
        pageBuilder: paymentInfoScreen.view,
      ),
      PageRoutingEntry(
        name: 'create_payment',
        pageBuilder: paymentTypeSelectorScreen.view,
        routeBuilder: ModalBottomSheetRouteBuilder().build,
      ),
      PageRoutingEntry(
        name: 'payment_order',
        pageBuilder: (pathArgs, queryArgs) {
          final configuration = PaymentOrderScreenArgsAdapter.adaptConfiguration(queryArgs);

          return paymentOrderScreen.view(pathArgs, queryArgs, configuration: configuration);
        },
      ),
]);
```

Навигация

```
di.getRoutingConfig().addRoutes([
      PageRoutingEntry(
        name: 'payment_info/:payment_id',
        pageBuilder: paymentInfoScreen.view,
      ),
      PageRoutingEntry(
        name: 'create_payment',
        pageBuilder: paymentTypeSelectorScreen.view,
        routeBuilder: ModalBottomSheetRouteBuilder().build,
      ),
      PageRoutingEntry(
        name: 'payment_order',
        pageBuilder: (pathArgs, queryArgs) {
          final configuration = PaymentOrderScreenArgsAdapter.adaptConfiguration(queryArgs);

          return paymentOrderScreen.view(pathArgs, queryArgs, configuration: configuration);
        },
      ),
]);
```

Навигация

```
di.getRoutingConfig().addRoutes([
      PageRoutingEntry(
        name: 'payment_info/:payment_id',
        pageBuilder: paymentInfoScreen.view,
      ),
      PageRoutingEntry(
        name: 'create_payment',
        pageBuilder: paymentTypeSelectorScreen.view,
        routeBuilder: ModalBottomSheetRouteBuilder().build,
      ),
      PageRoutingEntry(
        name: 'payment_order',
        pageBuilder: (pathArgs, queryArgs) {
          final configuration = PaymentOrderScreenArgsAdapter.adaptConfiguration(queryArgs);

          return paymentOrderScreen.view(pathArgs, queryArgs, configuration: configuration);
        },
      ),
]);
```

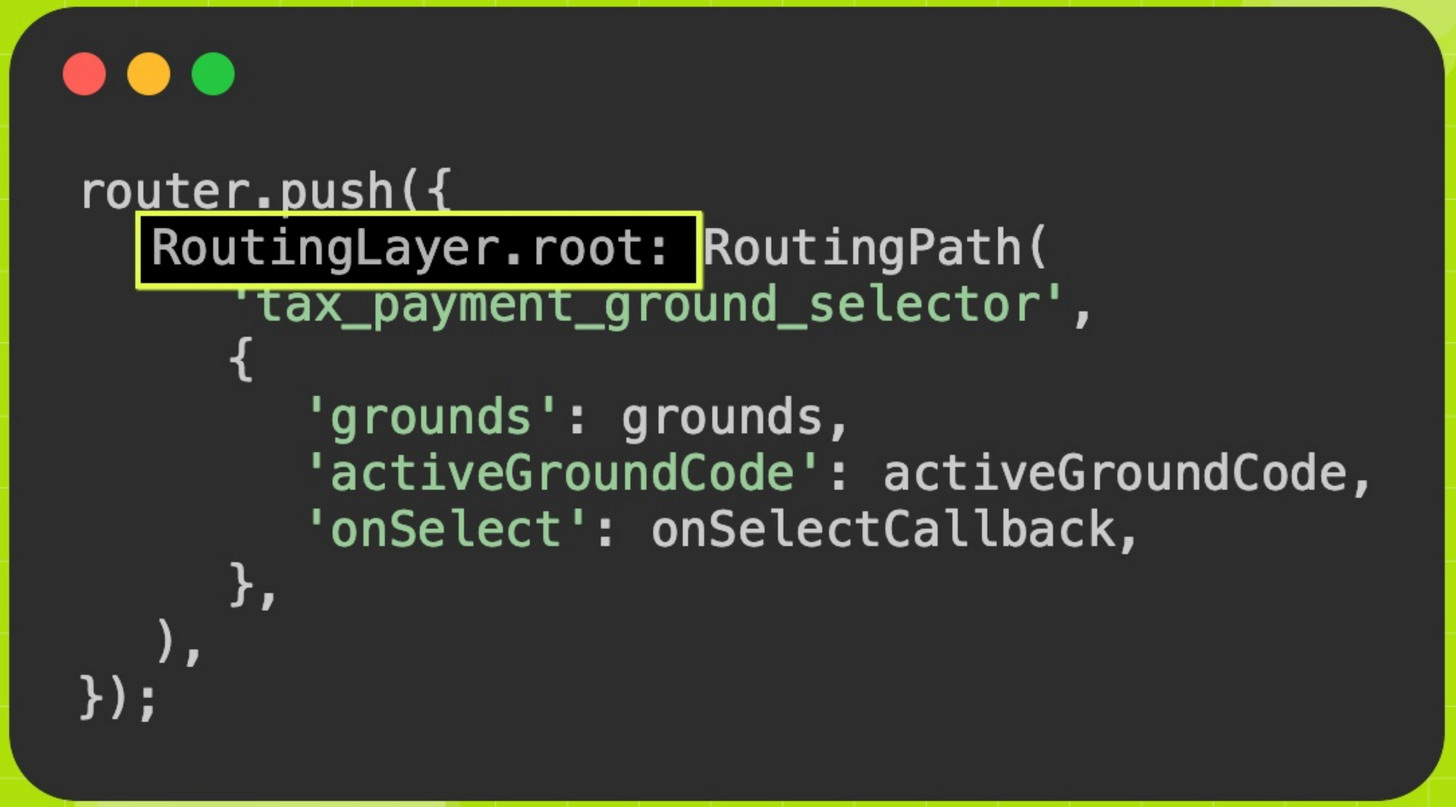
Навигация

```
di.getRoutingConfig().addRoutes([
      PageRoutingEntry(
        name: 'payment_info/:payment_id',
        pageBuilder: paymentInfoScreen.view,
      ),
      PageRoutingEntry(
        name: 'create_payment',
        pageBuilder: paymentTypeSelectorScreen.view,
        routeBuilder: ModalBottomSheetRouteBuilder().build,
      ),
      PageRoutingEntry(
        name: 'payment_order',
        pageBuilder: (pathArgs, queryArgs) {
          final configuration = PaymentOrderScreenArgsAdapter.adaptConfiguration(queryArgs);

          return paymentOrderScreen.view(pathArgs, queryArgs, configuration: configuration);
        },
      ),
]);
```

Навигация

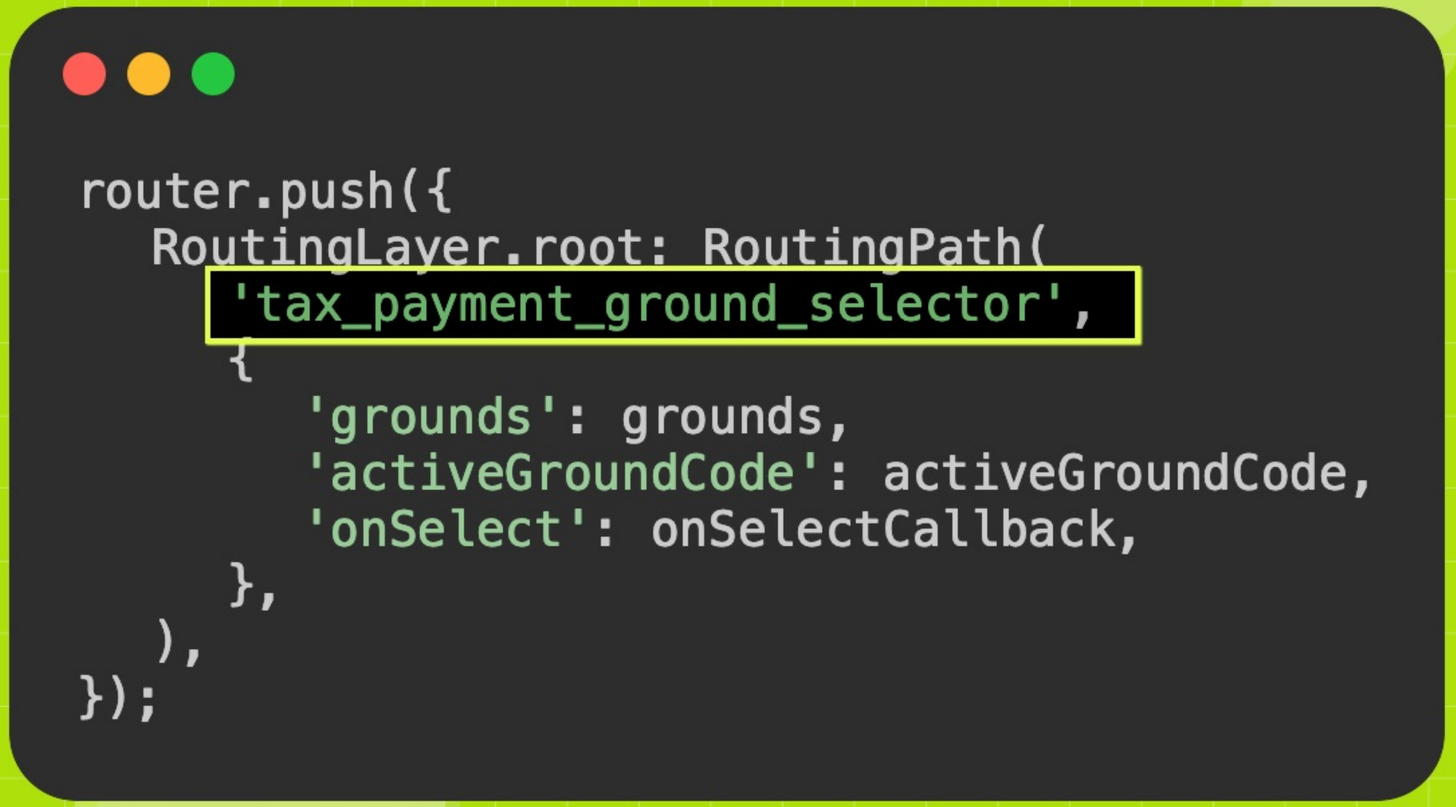
```
router.push({
   RoutingLayer.root: RoutingPath(
      'tax_payment_ground_selector',
      {
         'grounds': grounds,
         'activeGroundCode': activeGroundCode,
         'onSelect': onSelectCallback,
      },
   ),
});
```

Навигация



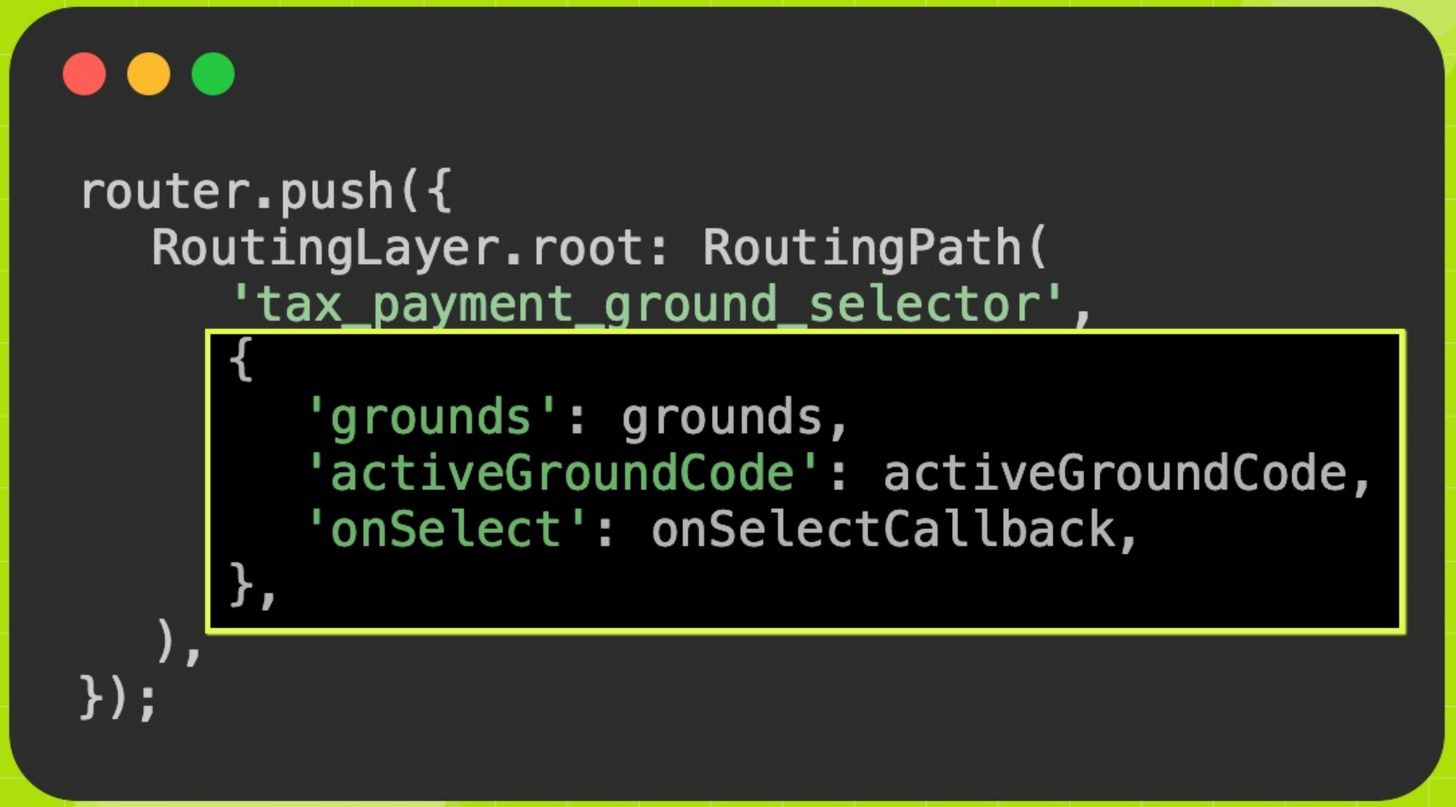
```
router.push({
   RoutingLayer.root: RoutingPath(
      'tax_payment_ground_selector',
      {
         'grounds': grounds,
         'activeGroundCode': activeGroundCode,
         'onSelect': onSelectCallback,
      },
   ),
});
```

Навигация



```
router.push({
   RoutingLayer.root: RoutingPath(
      'tax_payment_ground_selector',
      {
         'grounds': grounds,
         'activeGroundCode': activeGroundCode,
         'onSelect': onSelectCallback,
      },
   ),
});
```

Навигация



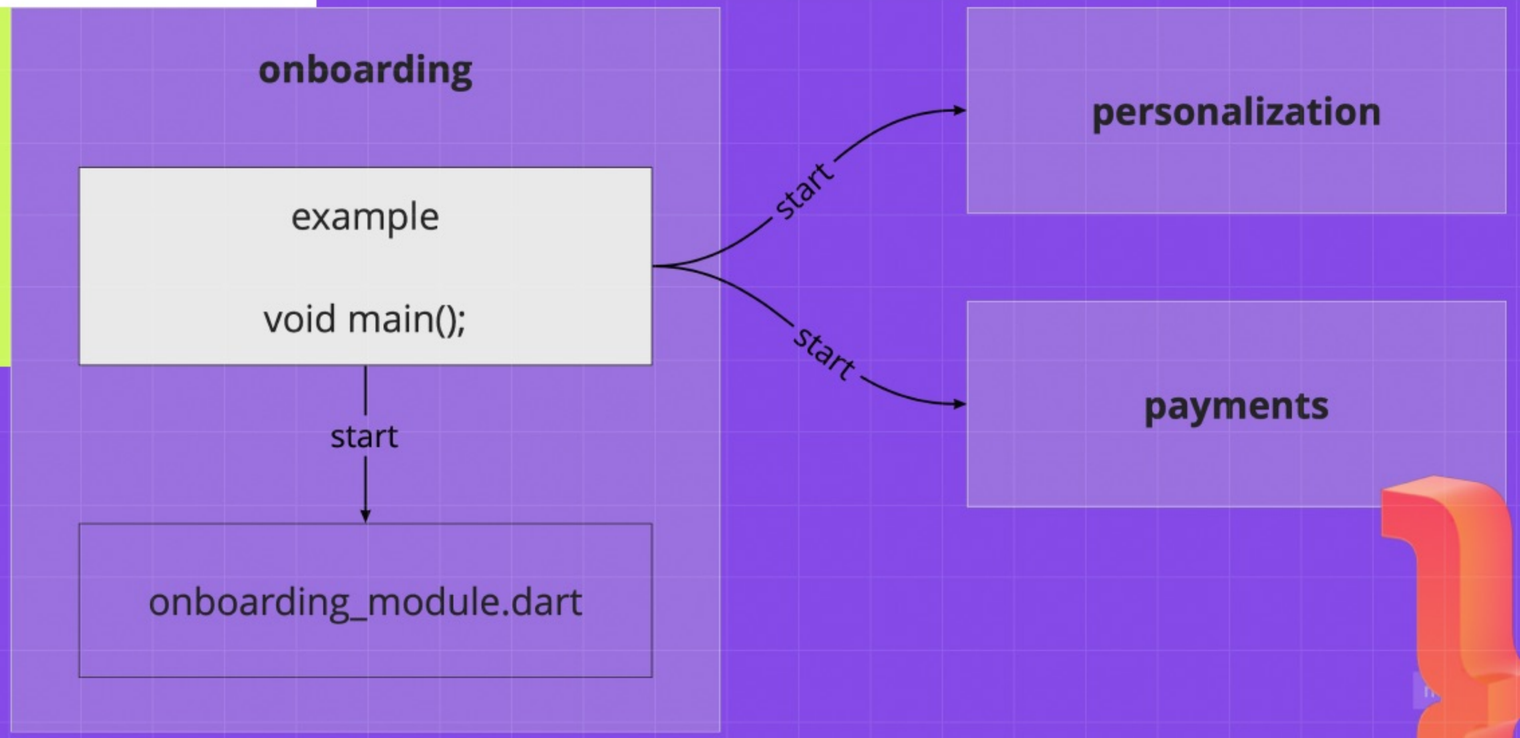
```
router.push({
   RoutingLayer.root: RoutingPath(
      'tax_payment_ground_selector',
      {
         'grounds': grounds,
         'activeGroundCode': activeGroundCode,
         'onSelect': onSelectCallback,
      },
   ),
});
```

Быстрый запуск

Как запустить модуль отдельно

Как запустить модуль отдельно

Для удобства разработки
и тестирования
в проекте предусмотрена
возможность
запуска каждого
модуля по отдельности.




Apache Kafka in Flutter

Как

модули общаются друг с другом

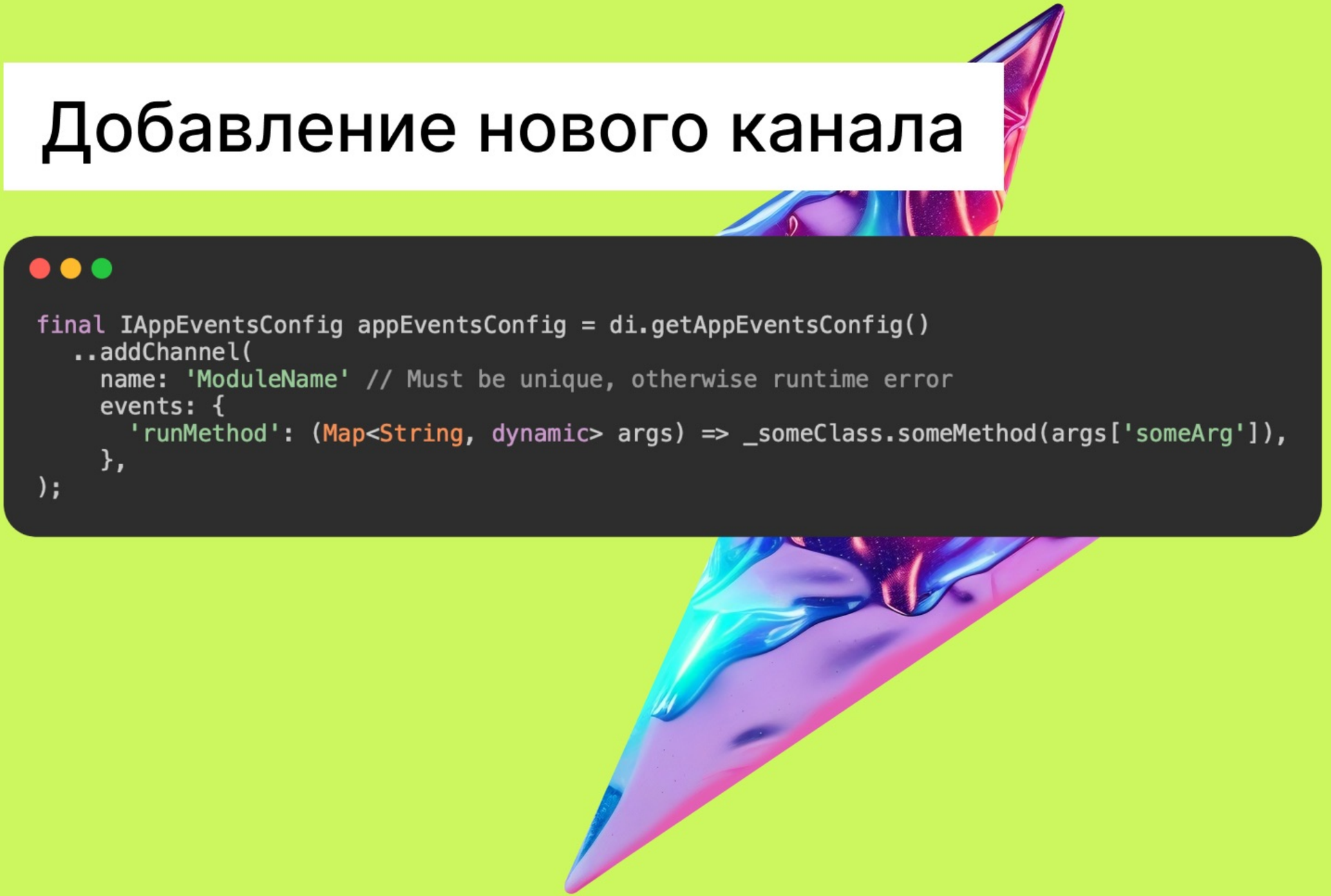
Когда не дублировать код

- Если логика сильно связана с другим модулем

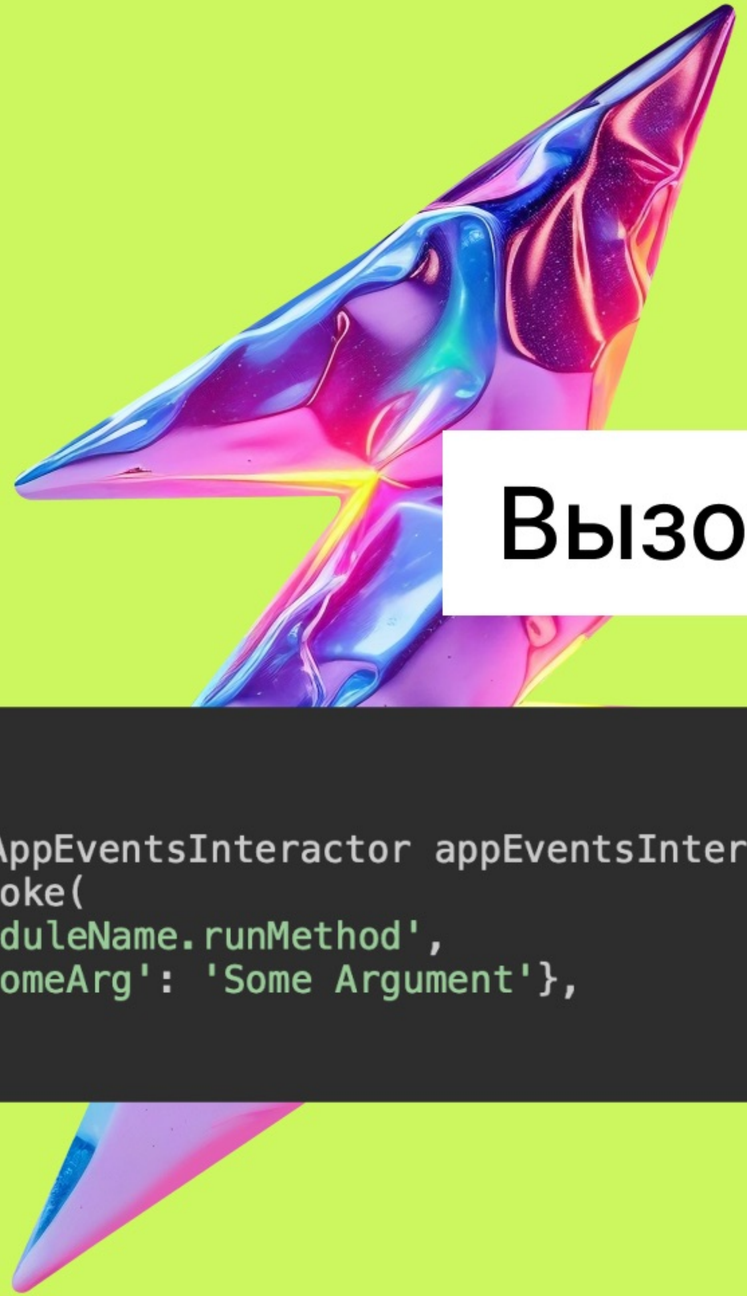
- Если требуется продублировать большую часть кода отдельного модуля

Решение: использовать механизм сообщений для взаимодействия с нужным модулем

Добавление нового канала



```dart
final IAppEventsConfig appEventsConfig = di.getAppEventsConfig()
  ..addChannel(
    name: 'ModuleName' // Must be unique, otherwise runtime error
    events: {
      'runMethod': (Map<String, dynamic> args) => _someClass.someMethod(args['someArg']),
    },
);
```



Вызов эвента канала

```
final IAppEventsInteractor appEventsInteractor = di.getAppEventsInteractor()
  ..invoke(
    'ModuleName.runMethod',
    {'someArg': 'Some Argument'},
);
```

Результаты

1. Улучшение качества кода:

   Более структурированный и понятный код.

2. Ускорение разработки:

   Сокращение времени на разработку новых фич.

3. Легкость интеграции новых функций:

   Быстрая и простая интеграция новых функциональностей.

Результаты

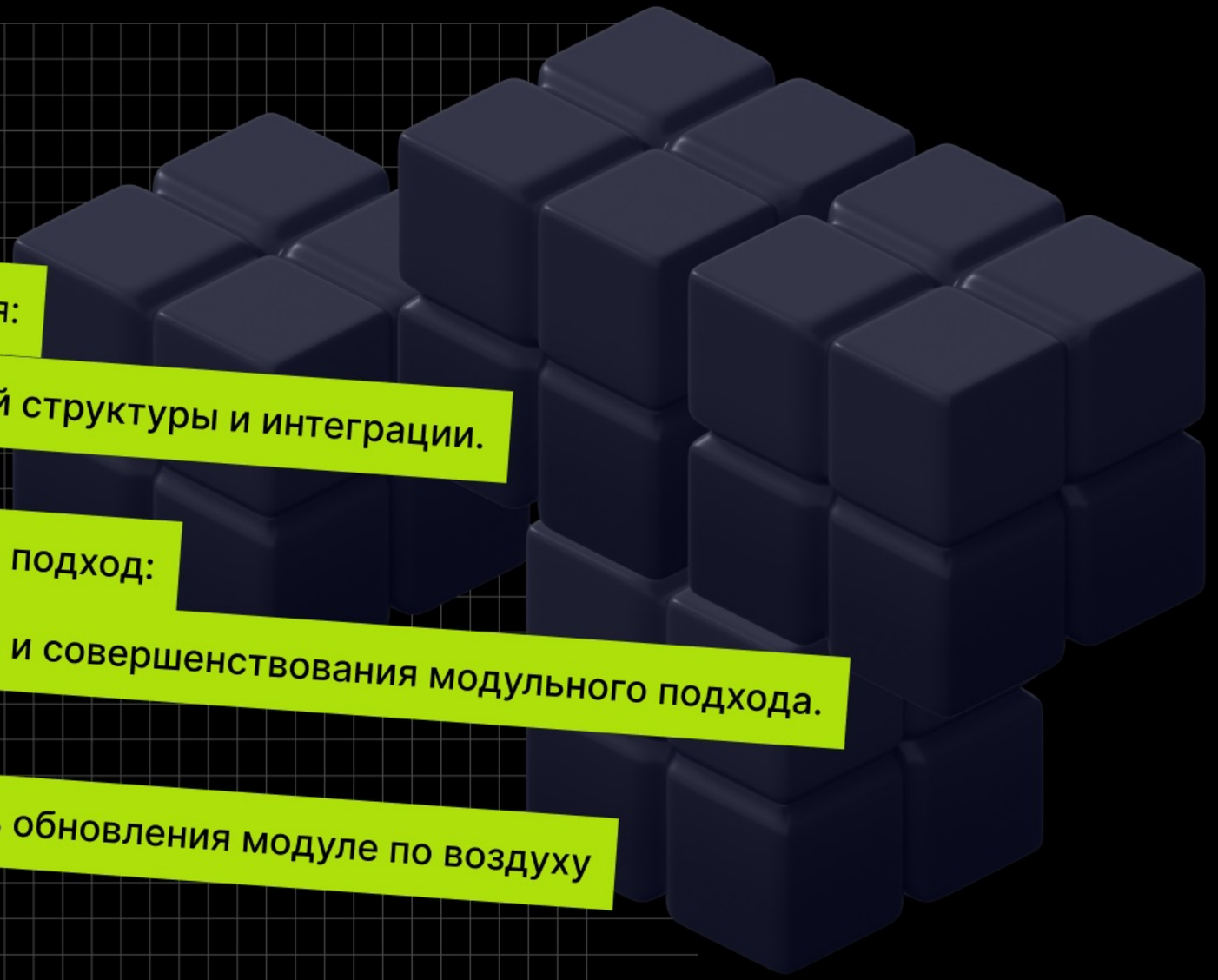
1. Модульная архитектура оправдала себя:

Решение проблем матричной структуры и интеграции.

2. Планируем продолжать развивать подход:

Продолжение развития и совершенствования модульного подхода.

3. Рассматриваем возможность обновления модуле по воздуху



Спасибо за внимание!

Кирилл Адещенко

Head of Mobile development в РСХБ

tg: @kaparray

kaparray@gmail.com