Мобильная архитектура для больших команд

Александр Михайлов, Sr Software Engineer

Uber

Несколько цифр

200+

Мобильных инженеров, работавших на новым приложением Uber

500+

Android инженеров, сделавших хотя бы один коммит в репозиторий

500+

iOS инженеров, сделавших хотя бы один коммит в репозиторий

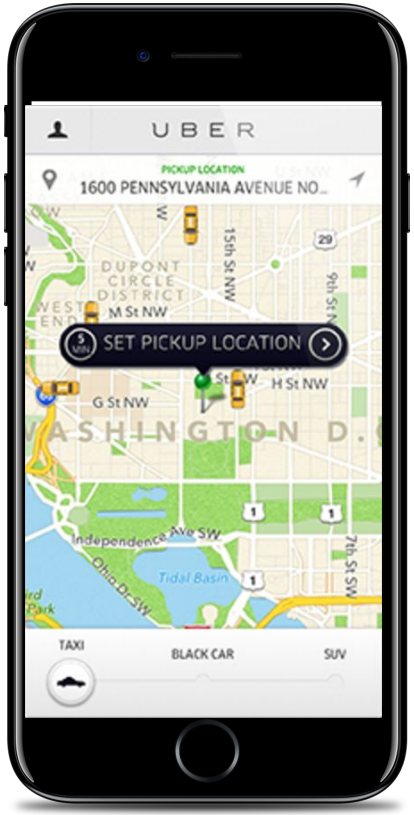
Agenda

01 Uber 3 года назад
02 От MVP до RIBs
03 Glue код
04 Как убер переезжал на новую архитектуру

Первая команда, разработавшая Uber

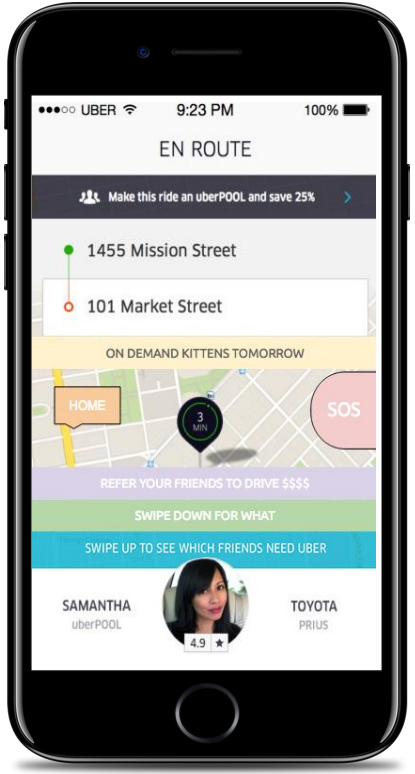
Иногда можно жить без архитектуры

- Небольшая команда, где изменения не происходят одновременно
- Небольшая кодовая база, позволяющая всем инженерам знать весь код



Проблемы роста

- Скорость разработки падает
- Количество багов растет, особенно в местах пересечения команд

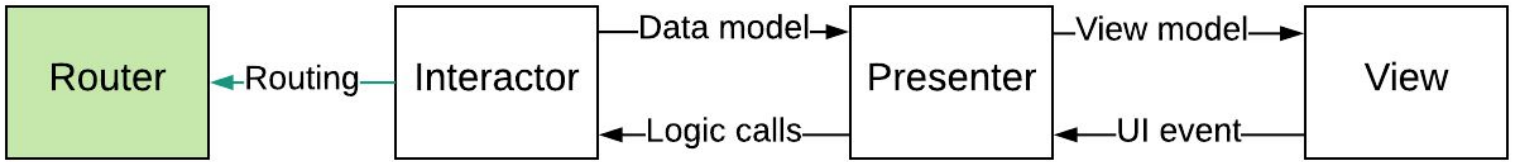


Что не так с MV*?

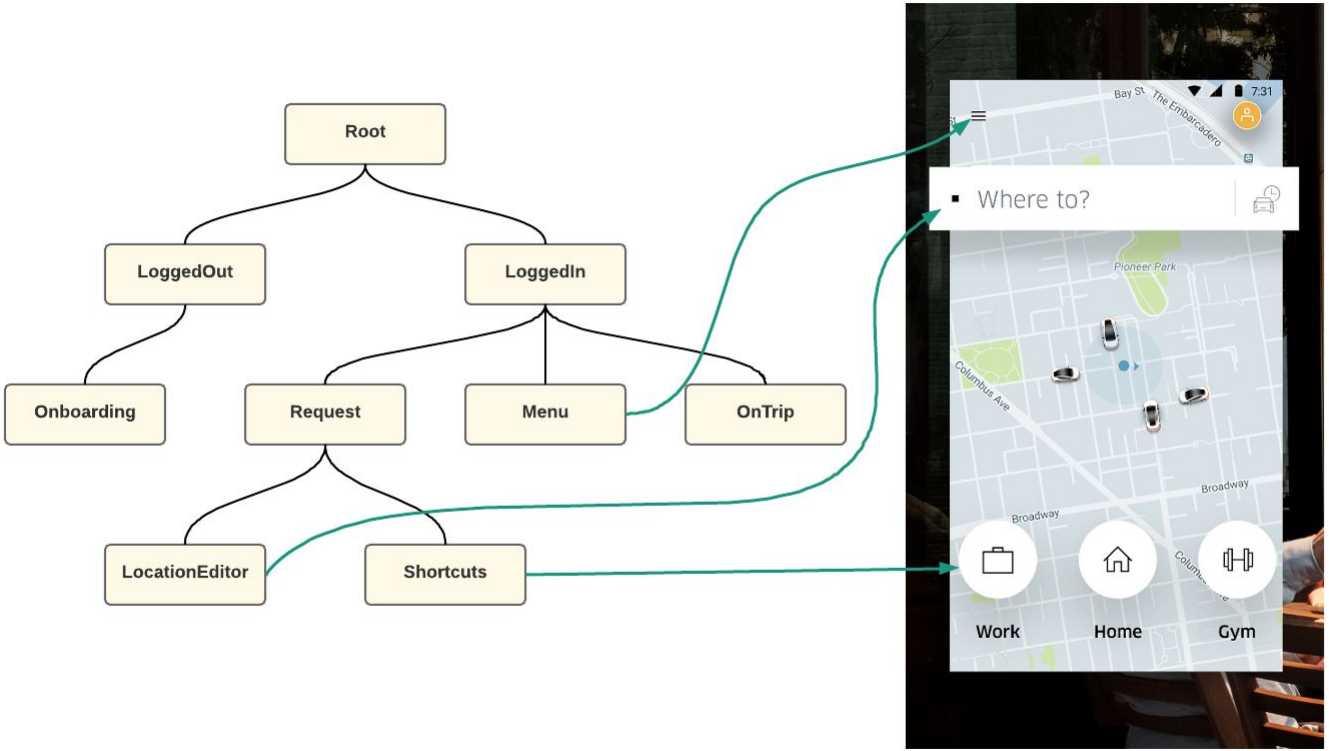
От MV* до RIB


Interactor
Data model
Logic calls
Presenter
View model
UI event
View

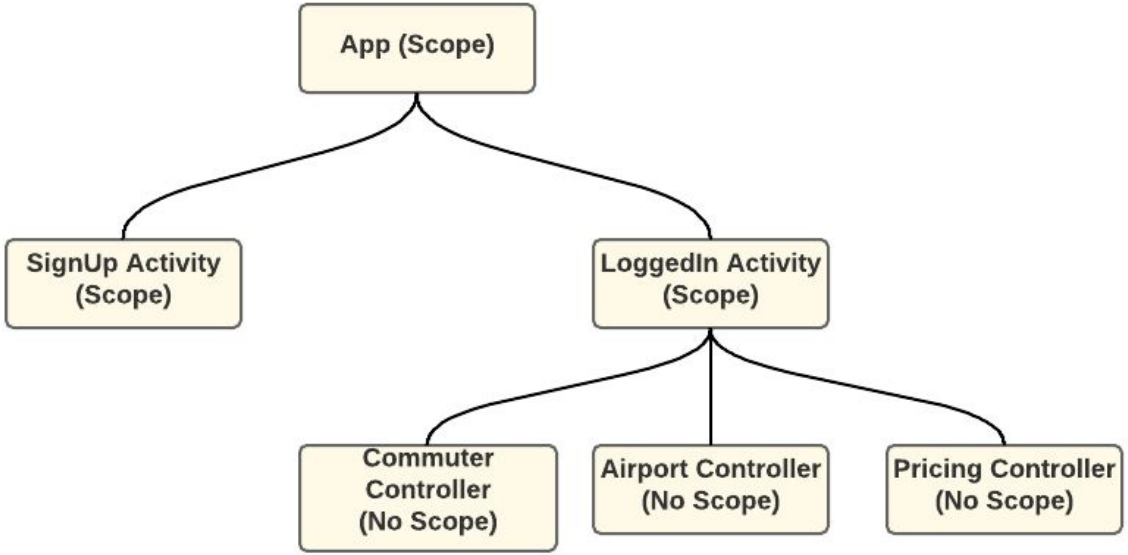
От MV* до RIB


Router
Routing
Interactor
Data model
Logic calls
Presenter
View model
UI event
View

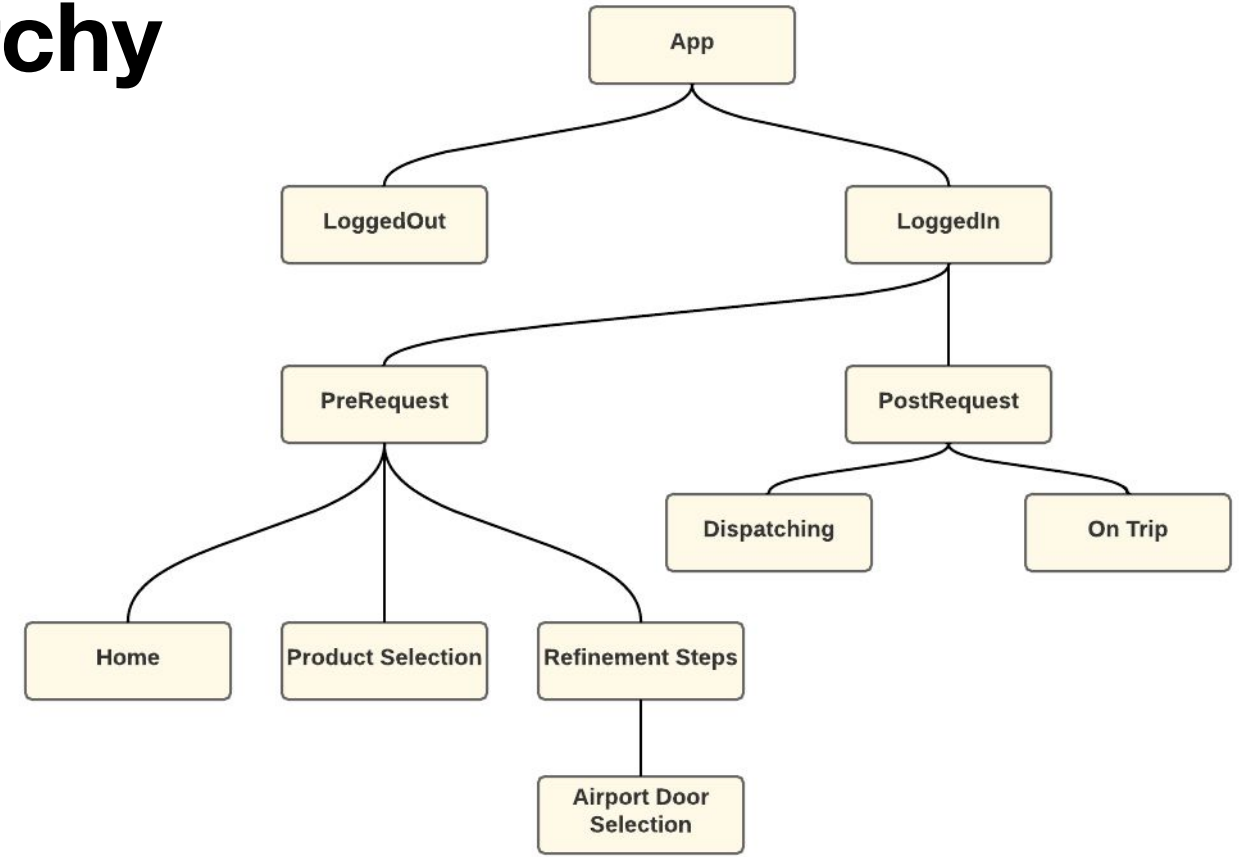
Высокая изоляция кода



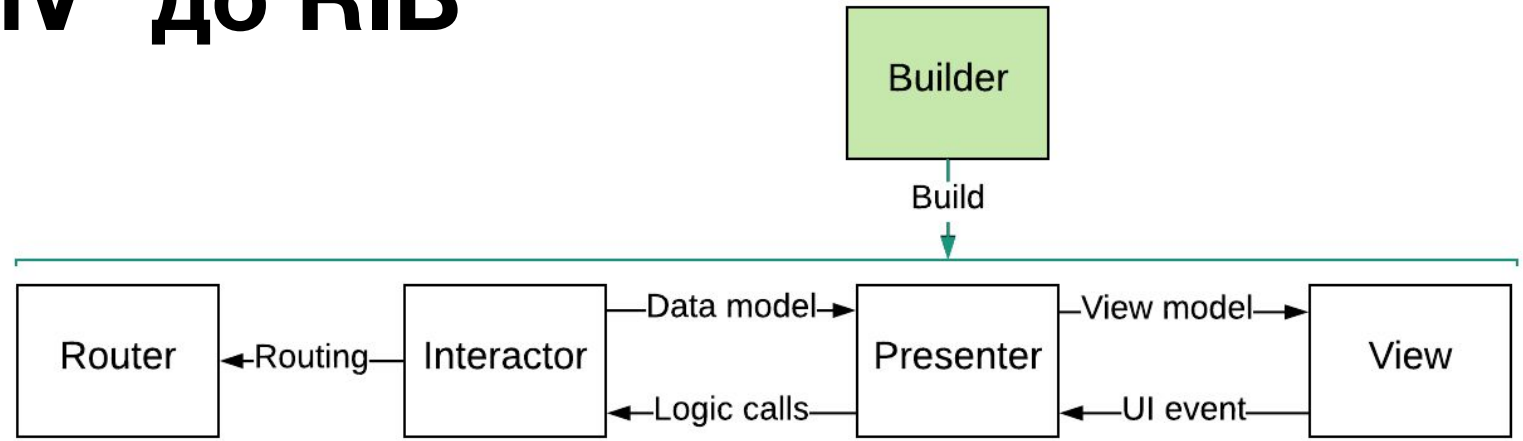
Старое приложение: 2 level scope hierarchy



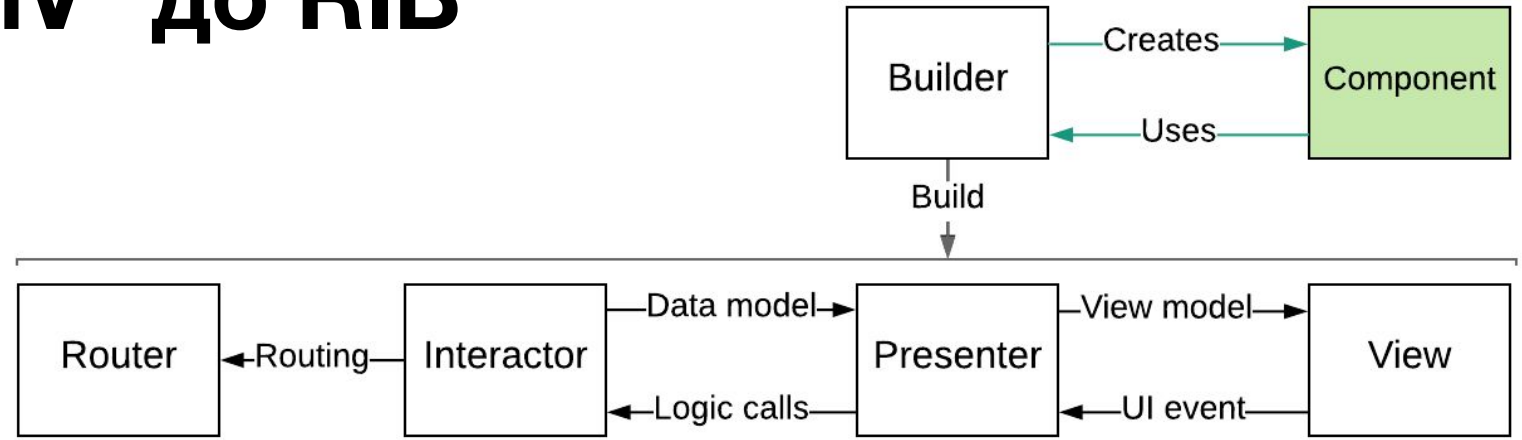
Новое приложение: Deep scope hierarchy



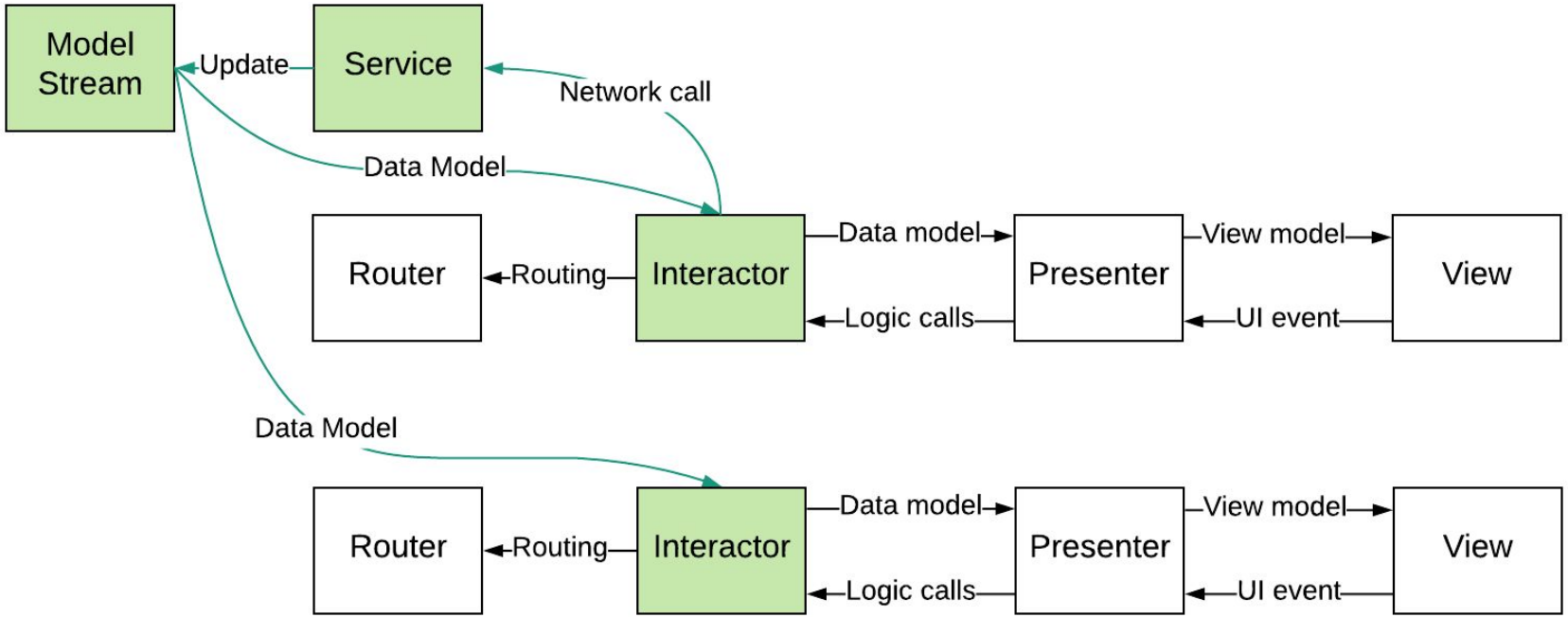
От MV* до RIB


Builder
Build
Router
Routing
Interactor
Data model
Logic calls
Presenter
View model
UI event
View

От MV* до RIB


Builder
Creates
Component
Uses
Build
Router
Routing
Interactor
Data model
Logic calls
Presenter
View model
UI event
View

От MV* до RIB


Model Stream
Update
Service
Network call
Data Model
Router
Routing
Interactor
Data model
Logic calls
Presenter
View model
UI event
View
Data Model
Router
Routing
Interactor
Data model
Logic calls
Presenter
View model
UI event
View

Переиспользуемость RIB

1. Любой RIB создается с помощью одного вызова Builder.build()

Переиспользуемость RIB

1. Любой RIB создается с помощью одного вызова Builder.build()
2. Каждый RIB имеет свой скоуп зависимостей. Все внешние зависимости описаны в его Builder классе.

Переиспользуемость RIB

1. Любой RIB создается с помощью одного вызова Builder.build()
2. Каждый RIB имеет свой скоуп зависимостей. Все внешние зависимости описаны в его Builder классе.
3. RIB не знает ничего про своего родителя. Это позволяет переиспользовать его в любом месте приложения.

Преимущества RIBs

1. Легкая навигация в коде.
   Бизнес логика всегда в Interactor, логика навигации в Router, а зависимости в Builder.

Преимущества RIBs

1. Легкая навигация в коде.
   Бизнес логика всегда в Interactor, логика навигации в Router, а зависимости в Builder.
2. Высокая переиспользуемость RIB.
   Один и тот же RIB может быть запущен в разные частях приложения и даже в разных приложениях.

Преимущества RIBs

1. Легкая навигация в коде.
   Бизнес логика всегда в Interactor, логика навигации в Router, а зависимости в Builder.
2. Высокая переиспользуемость RIB.
   Один и тот же RIB может быть запущен в разные частях приложения и даже в разных приложениях.
3. Высокая изоляция кода.
   Каждая команда играет со своими RIB и никак не взаимодействует с другими RIB.

Glue код. Проблемы.

Glue код. Проблемы.

- Сложно найти место в котором нужно подключить новую фичу

Glue код. Проблемы.

- Сложно найти место в котором нужно подключить новую фичу
- Чем больше фич - тем сложнее понять как работает приложение

Glue код. Проблемы.

- Сложно найти место в котором нужно подключить новую фичу
- Чем больше фич - тем сложнее понять как работает приложение
- Нестабильность. Очень легко сломать одну фичу, пока разработчик подключает другую

Glue код. Проблемы.

- Сложно найти место в котором нужно подключить новую фичу
- Чем больше фич - тем сложнее понять как работает приложение
- Нестабильность. Очень легко сломать одну фичу, пока разработчик подключает другую
- Добавление каждой следующей фичи требует все больше и больше времени

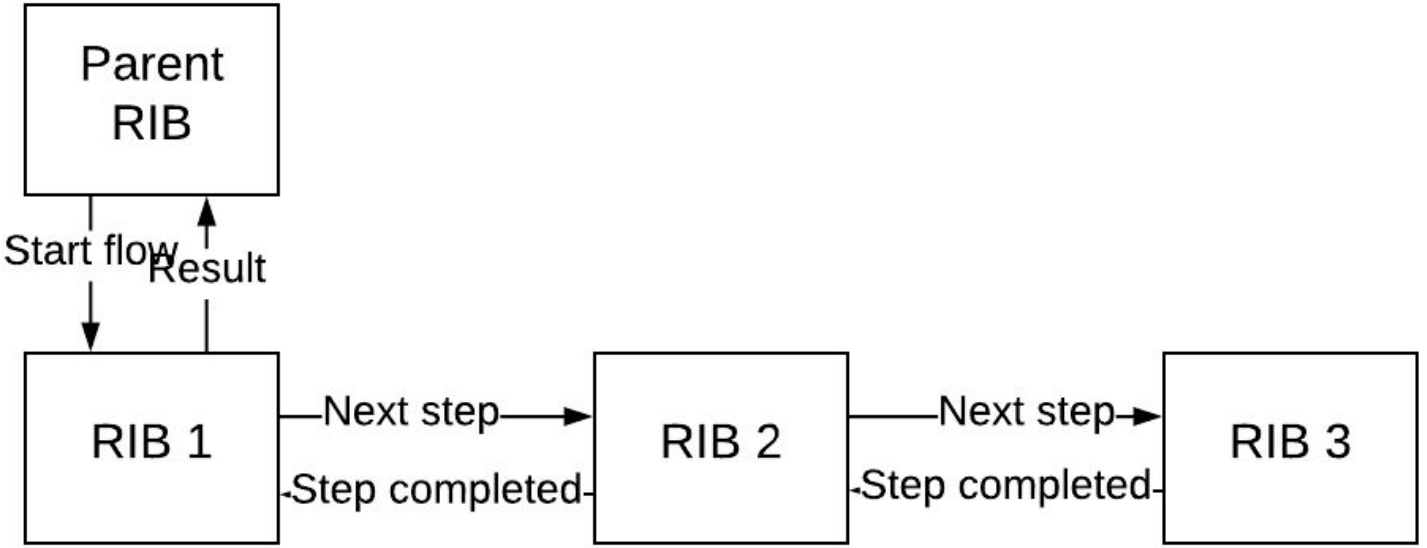
Glue код

- Что делать если добавление способа оплаты занимает больше одного шага

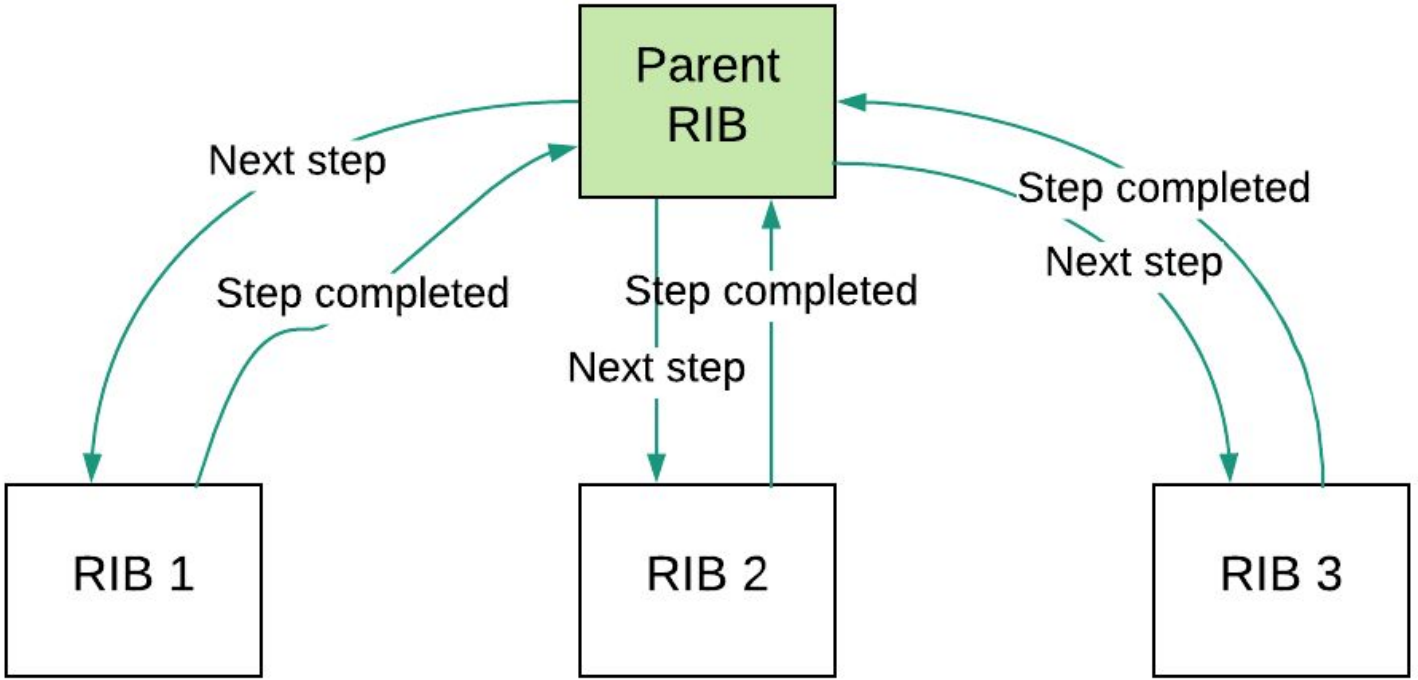
Операция за несколько шагов

1. Выбор банковской карточки, как метода оплаты
2. Экран ввода информации о банковской карте
3. Введение кода двухфакторной аутентификации

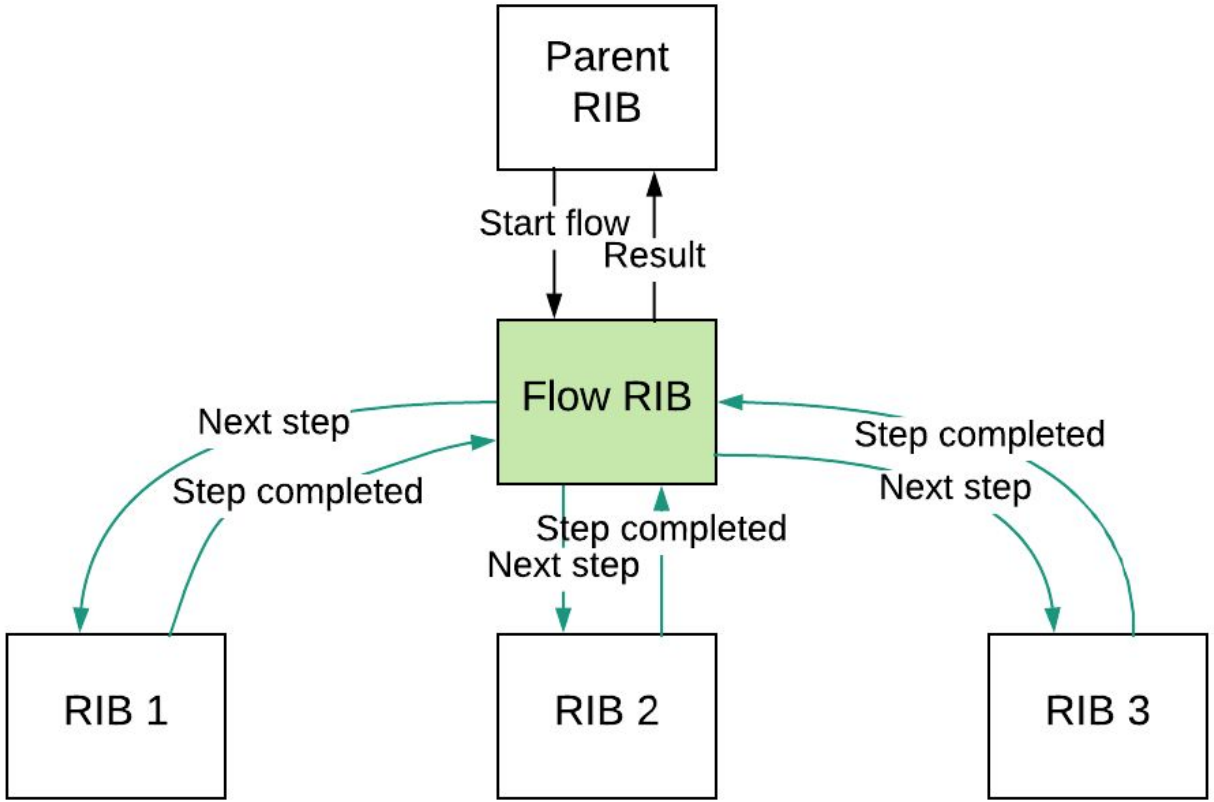
Flow


Parent
RIB
Start flow
Result
RIB 1
Next step
Step completed
RIB 2
Next step
Step completed
RIB 3

Flow

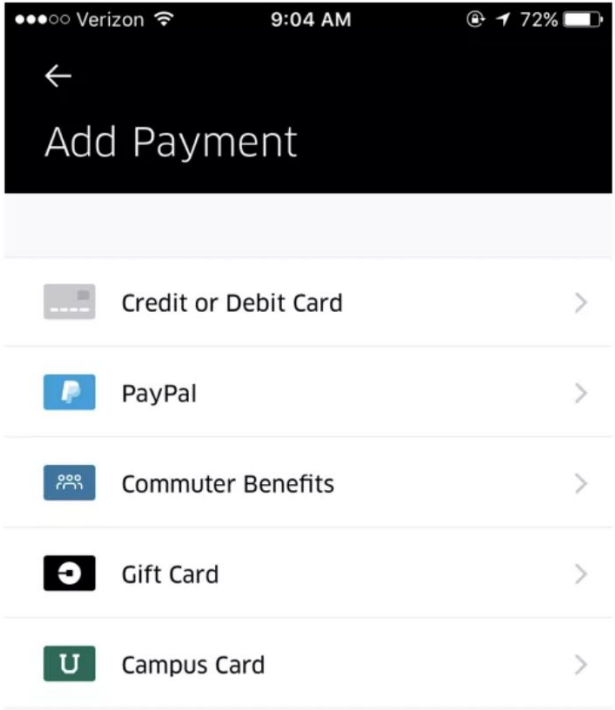

Parent
RIB
Next step
Step completed
Step completed
Next step
Step completed
Next step
RIB 1
RIB 2
RIB 3

Flow RIB

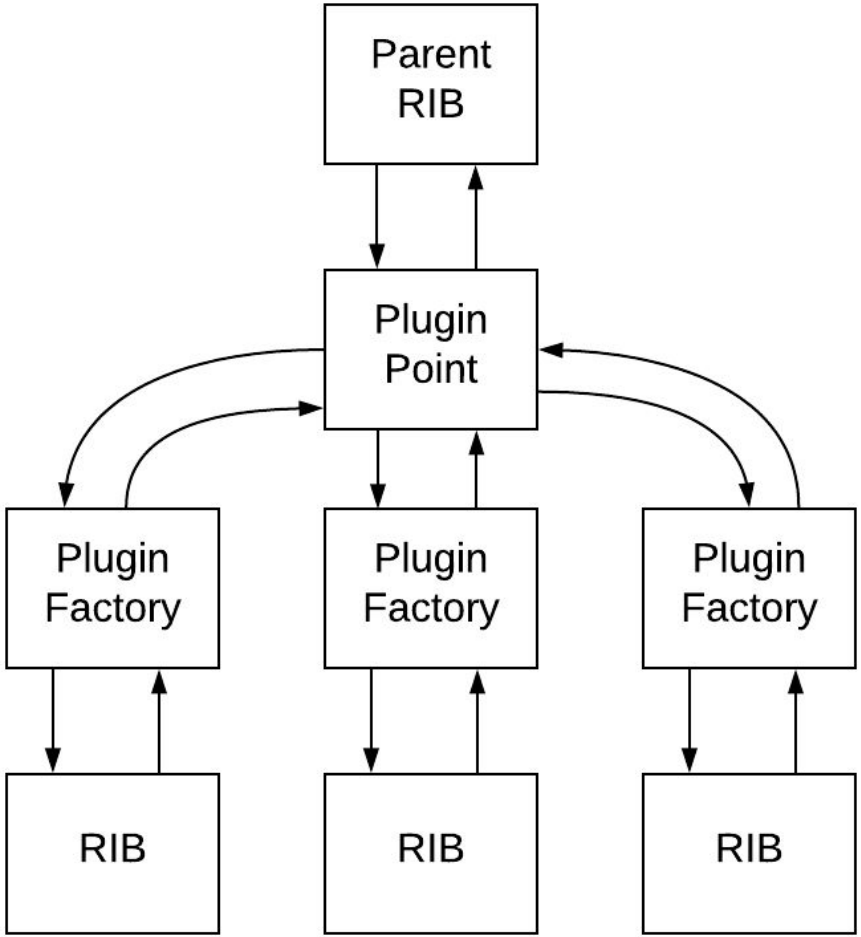

Parent RIB
Start flow
Result
Flow RIB
Next step
Step completed
Step completed
Next step
Step completed
Next step
RIB 1
RIB 2
RIB 3

Glue код

- Что делать если добавление способа оплаты занимает больше одного шага
- Как проверить доступность каждого элемента и не потонуть в этом



Plugin points



Plugin points

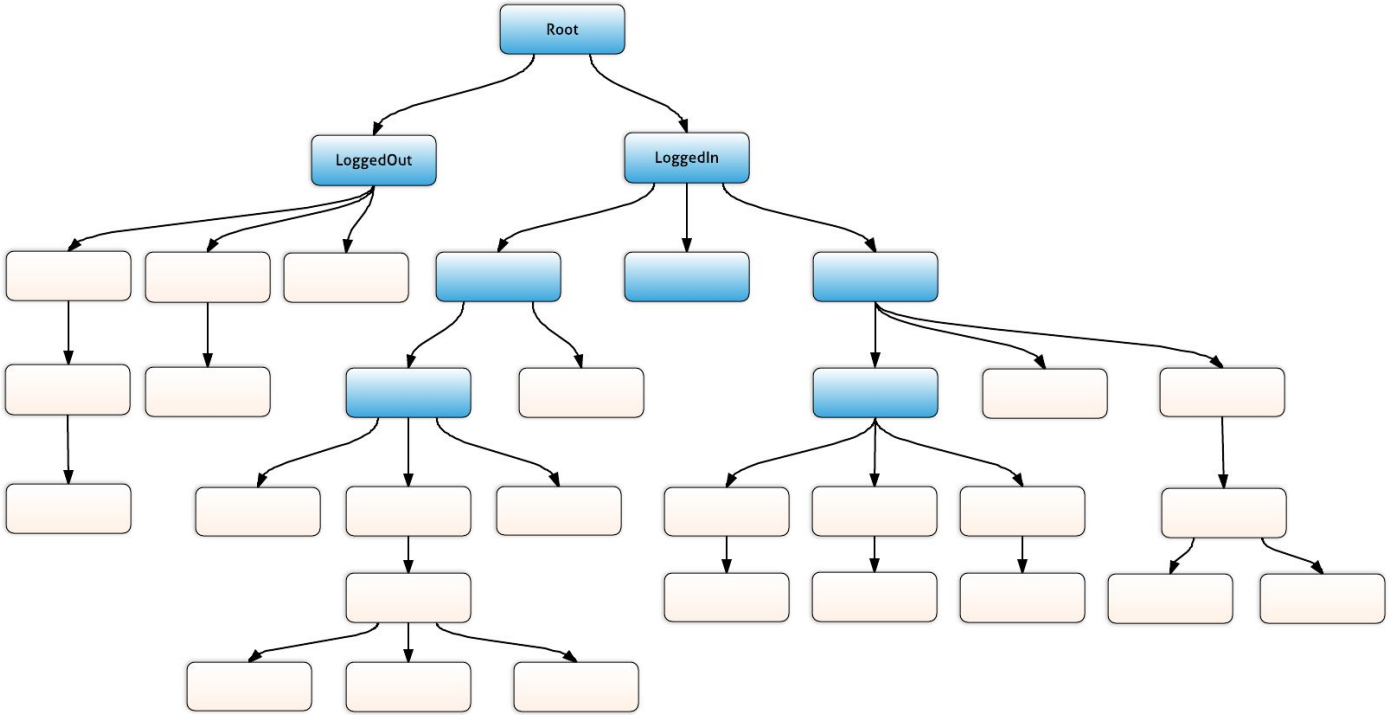
```
abstract class PluginPoint<T, P> {
  List<P> getApplicablePlugin(T context) {
    //Implementation
    { ... }
  }

  abstract List<P> getPlugins();
}

abstract class PluginFactory<T> {
  abstract String getKillswitchName();

  abstract boolean isApplicable(T Context);
}
```

Core vs non core



Преимущества plugin point

- Простая разработка новых фич. Для интеграции как правило уже есть Plugin Point.

Преимущества plugin point

- Простая разработка новых фич. Для интеграции как правило уже есть Plugin Point.
- Простота понимания. Для понимания архитектуры всего приложения нужно понять только 20% кода.

Преимущества plugin point

- Простая разработка новых фич. Для интеграции как правило уже есть Plugin Point.
- Простота понимания. Для понимания архитектуры всего приложения нужно понять только 20% кода.
- Надежность. Каждый плагин имеет KillSwitch, что бы выключить с бекенда

Преимущества plugin point

- Простая разработка новых фич. Для интеграции как правило уже есть Plugin Point.
- Простота понимания. Для понимания архитектуры всего приложения нужно понять только 20% кода.
- Надежность. Каждый плагин имеет KillSwitch, что бы выключить с бекенда
- Масштабируемость. Glue код не вытекает в фичи. Добавление 501й фичи нисколько не усложняет код на вызывающей стороне.

Gif source: Giphy

Переезд на новую архитектуру большой командой

- Генерация болейрплейт кода
  - Плагины к IDE
  - Annotation processing

Переезд на новую архитектуру большой командой

- Генерация болейрплейт кода
  - Плагины к IDE
  - Annotation processing
- Onboarding
  - Документация
  - Туториалы

Переезд на новую архитектуру большой командой

- Генерация болейрплейт кода
  - Плагины к IDE
  - Annotation processing
- Onboarding
  - Документация
  - Туториалы
- Enforcing (architecture) patterns
  - Lint rules
  - Blocking code reviews

Несколько цифр

5+

Приложений используют единую архитектуру и переиспользуют RIB

1000+

RIBs было написано за последние 2 года

1700+

Модулей в Android репозитории

01 github.com/uber/RIBs

02 eng.uber.com/tag/uber-mobile

Спасибо

Михайлов Александр