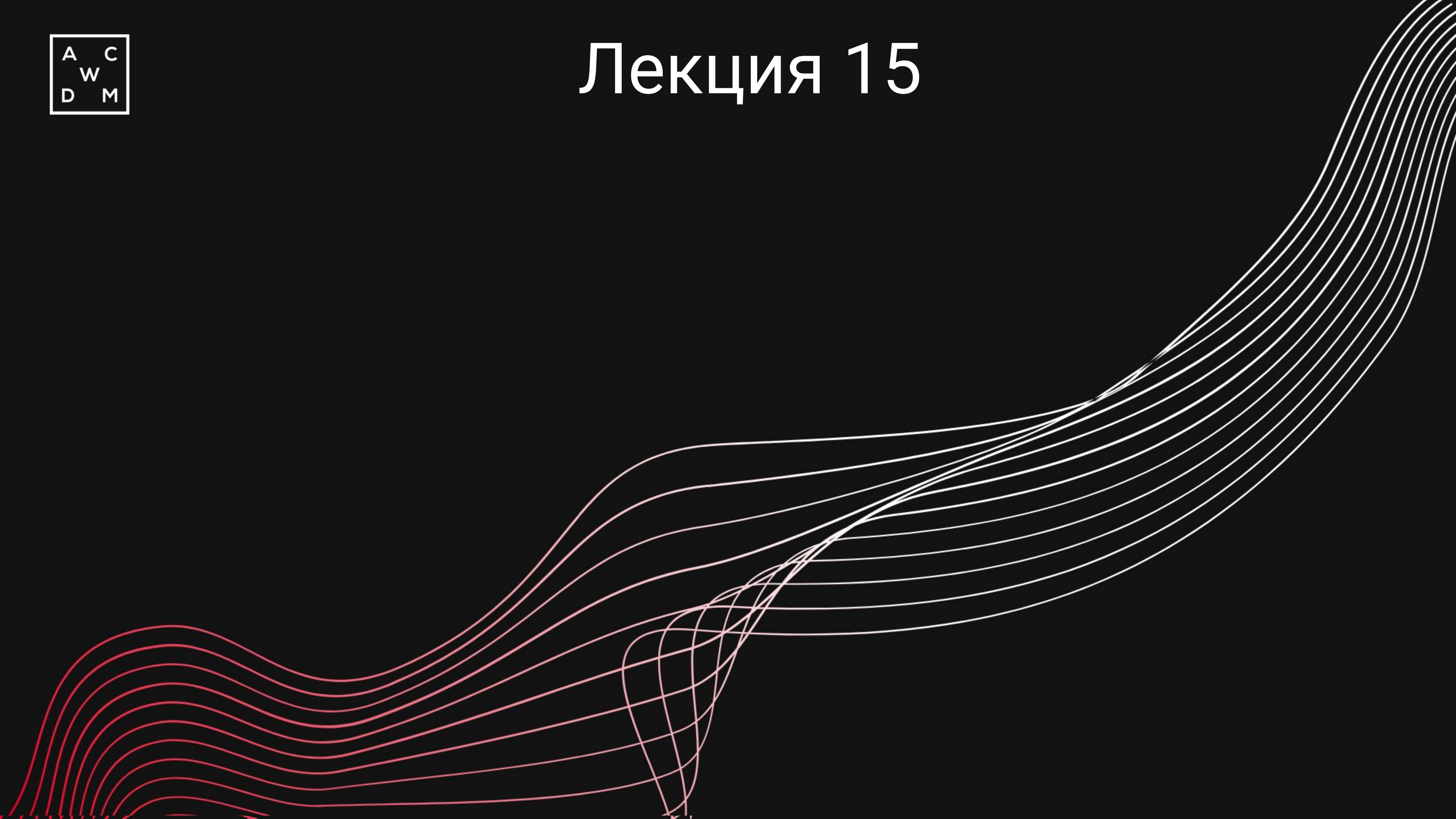


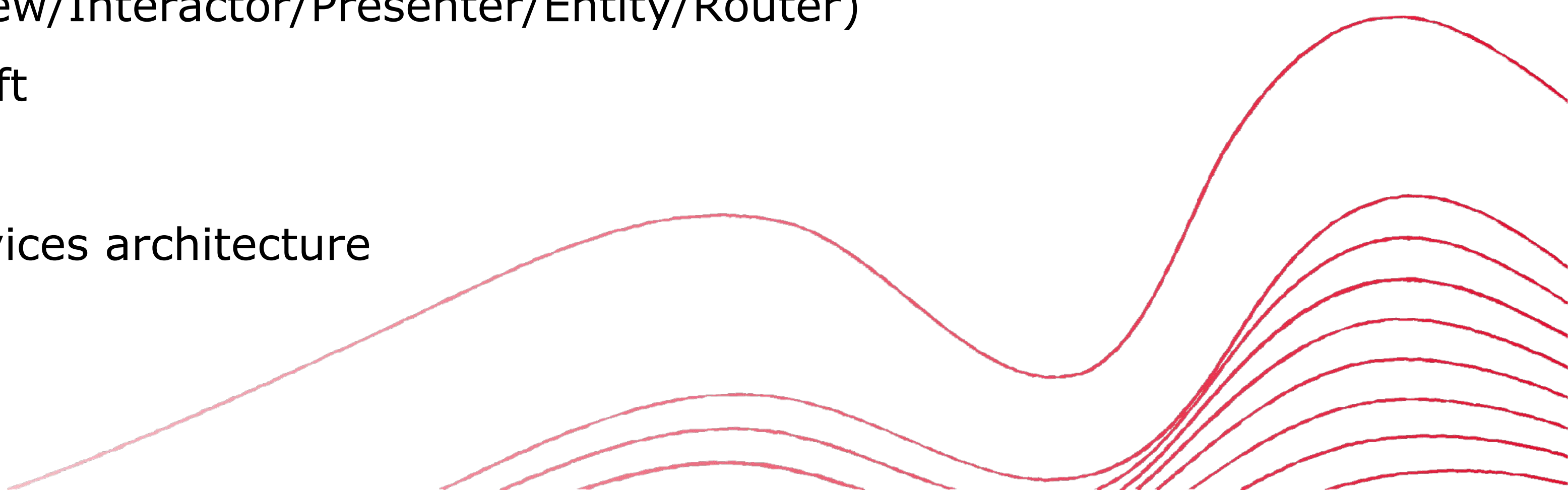


# Лекция 15



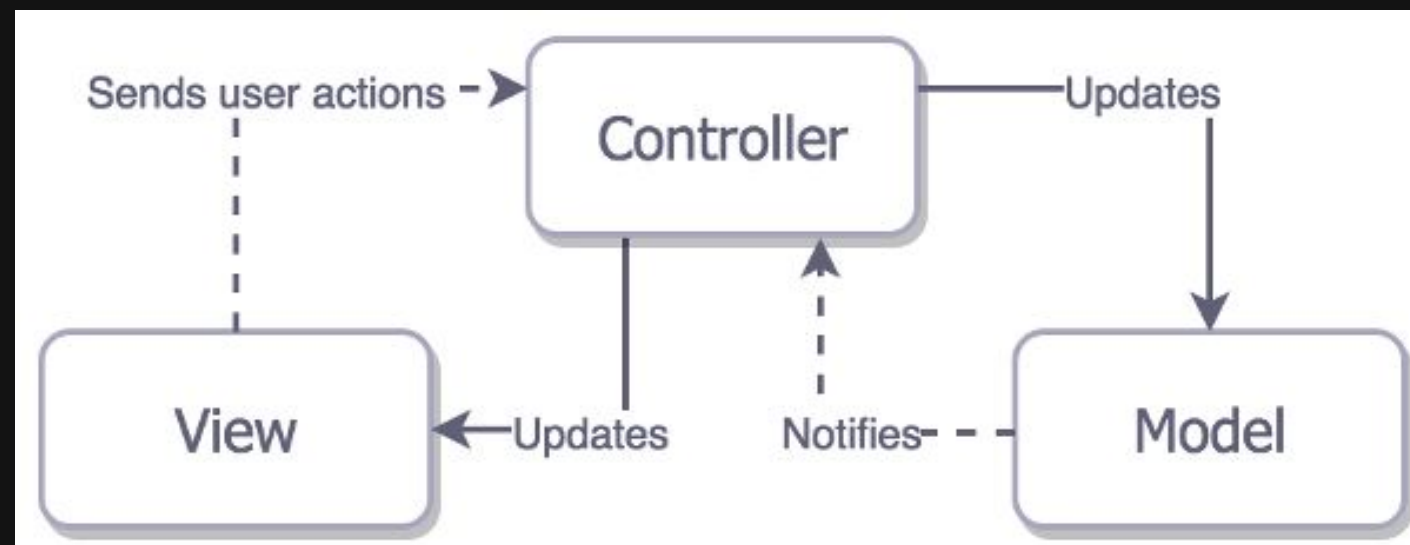


# Архитектура

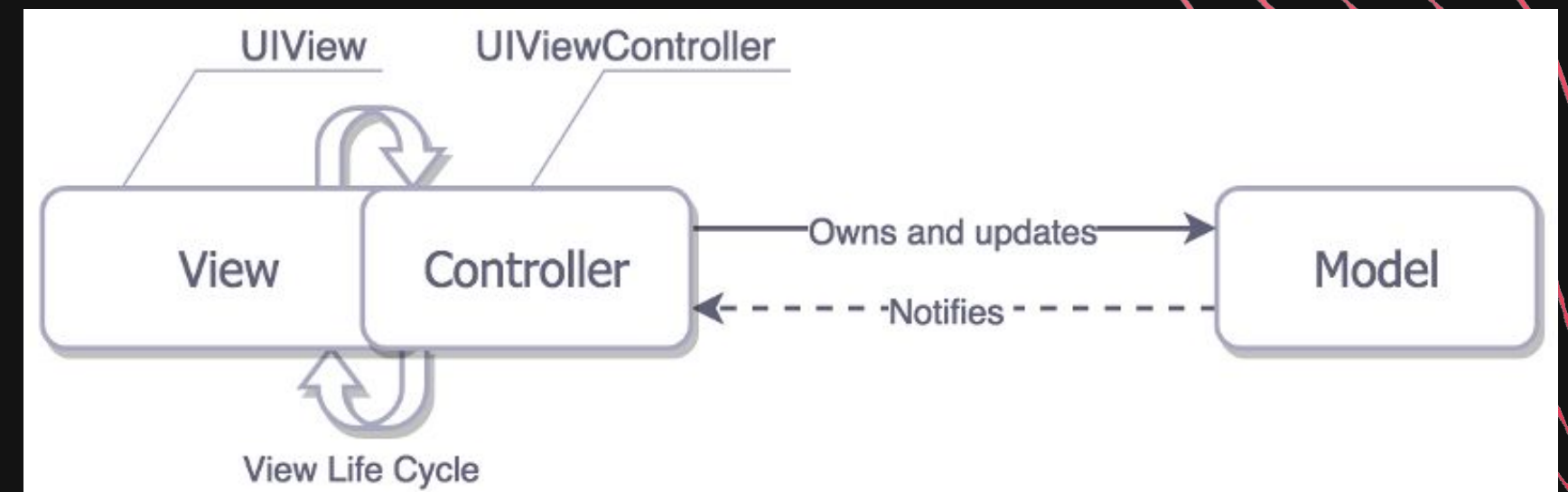
- Что такое архитектура приложения
  - MVC (Model View Controller)
  - MVVM (Model View ViewModel)
  - Viper (View/Interactor/Presenter/Entity/Router)
  - CleanSwift
  - Redux
  - Microservices architecture
- 



# MVC (Model View Controller)



Традиционный MVC



Используемый Apple MVC

Статья с разбором MVC у Apple и объяснением, что с ним не так:

<https://habr.com/ru/post/324414/>

(очень популярный вопрос на собеседованиях)

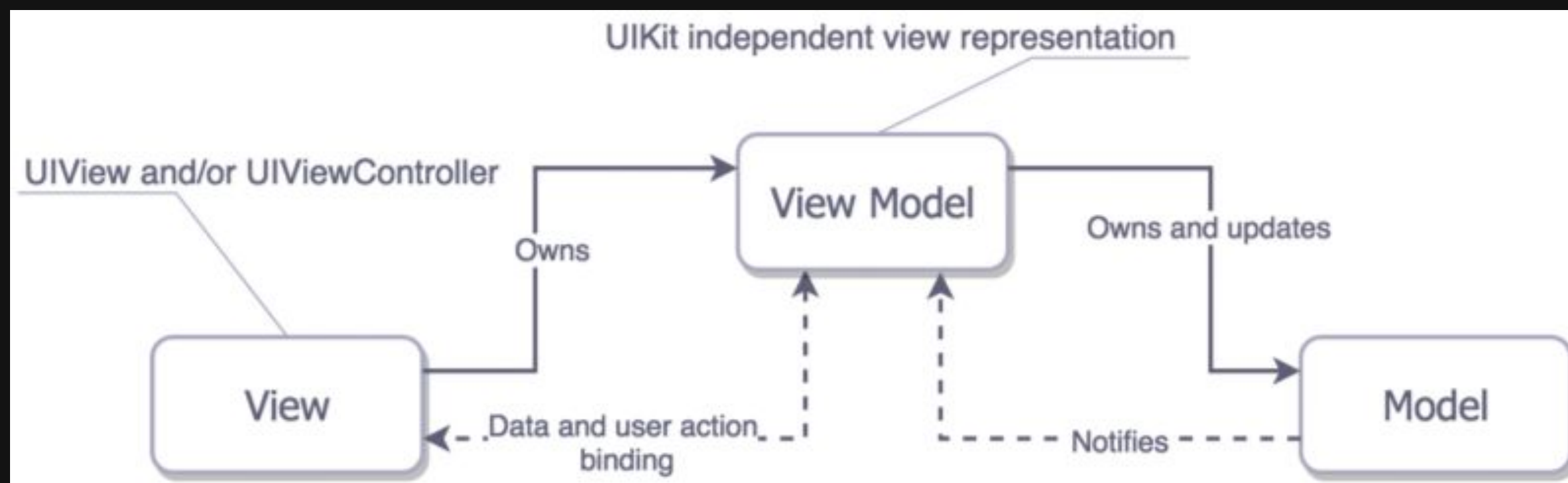




# MVVM (Model View ViewModel)

В качестве **View** - UIViewController, **ViewModel** - отдельный класс, **Model** - хранит только описание моделей.

ViewModel выступает посредником между View и Model, благодаря чему сами View и Model не знают друг о друге.



MVVM в большинстве случаев используют с биндингом. Биндинг слишком обширная тема чтобы вставлять ее в наш курс. Тем не менее самый простой вариант работы с биндингами - библиотека Bond (если интересует более глубокий подход - почитайте об RxSwift). С библиотекой Bond можно ознакомиться в этом tutorialе:

<https://www.raywenderlich.com/667-bond-tutorial-bindings-in-swift>



# VIPER

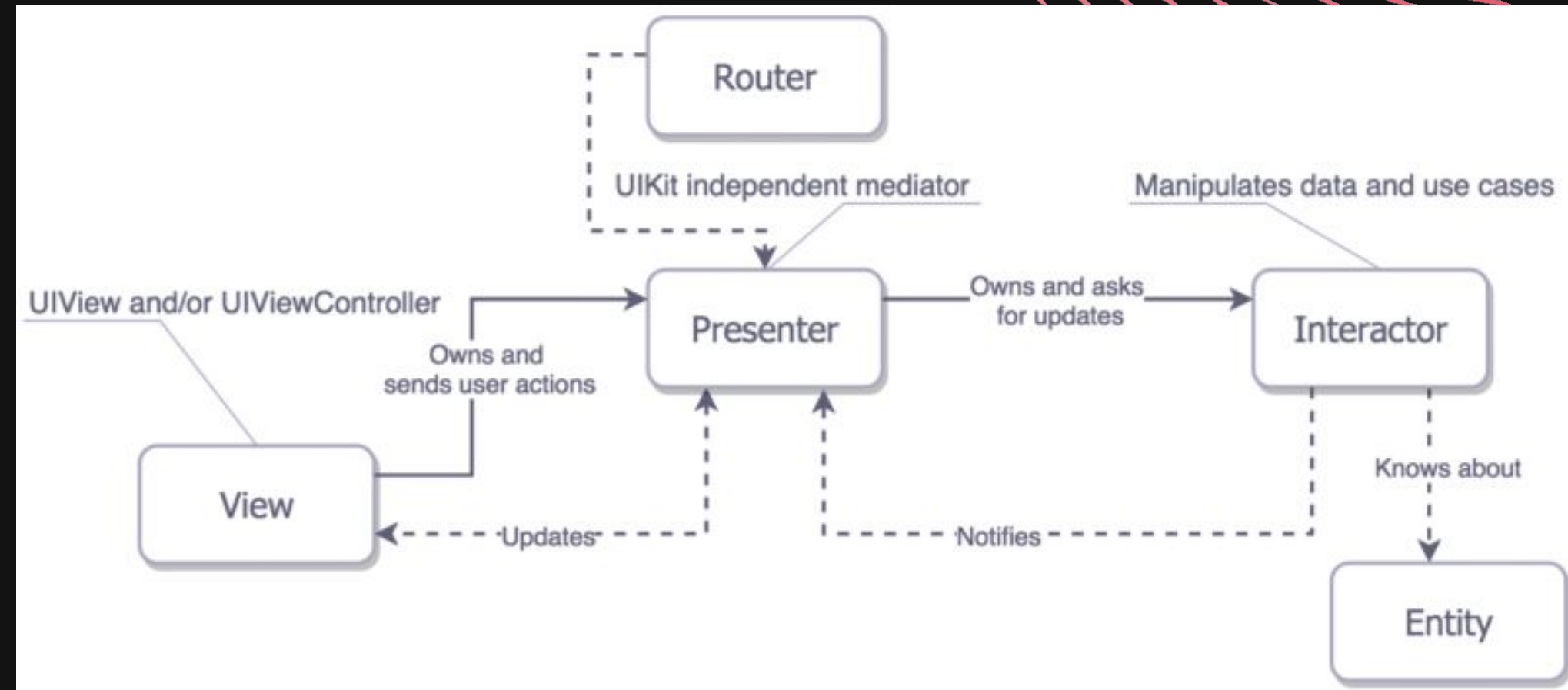
**V** - View, отвечает за UI (user interface)

**I** - Interactor, содержит бизнес логику связанную с данными

**P** - Presenter, содержит бизнес логику связанную с UI

**E** - отвечает только за описание моделей

**R** - Router, отвечает за переходы между VIPER модулями



Хороший и подробный разбор VIPER. Есть простой пример и сложный пример:

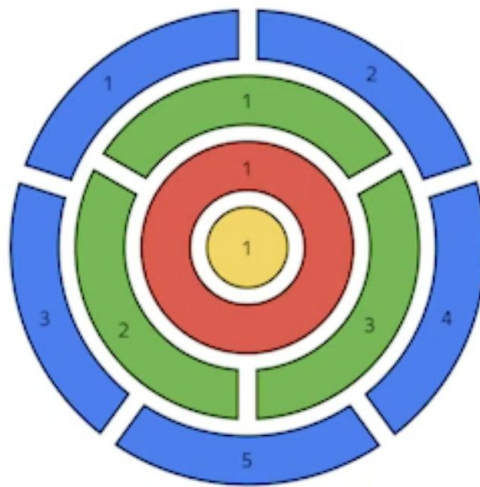
<https://habr.com/ru/post/358412/>



# Clean Swift

## Чистая архитектура

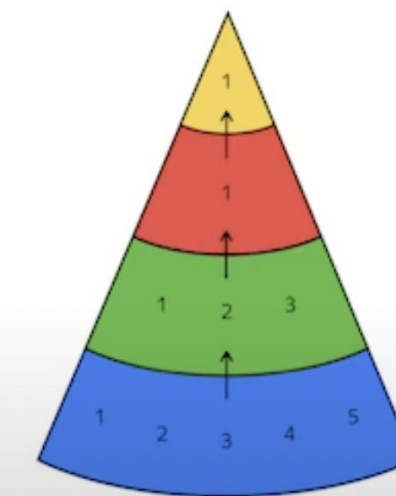
- 1 Entities
- 1 Use Cases
- 1 Controllers
- 2 Gateways
- 3 Presenters
- 1 Devices
- 2 Web
- 3 DB
- 4 UI
- 5 External Interfaces



- Enterprise Business Rules
- Application Business Rules
- Interface Adapters
- Framework & Drivers

## Чистая архитектура

- 1 Entities
- 1 Use Cases
- 1 Controllers
- 2 Gateways
- 3 Presenters
- 1 Devices
- 2 Web
- 3 DB
- 4 UI
- 5 External Interfaces



Abstract, General, Rarely Change

Level

Concrete, Specific, Change Frequently

- Enterprise Business Rules
- Application Business Rules
- Interface Adapters
- Framework & Drivers

Сравнение Clean swift с VIPER:

<https://habr.com/ru/post/415725/>

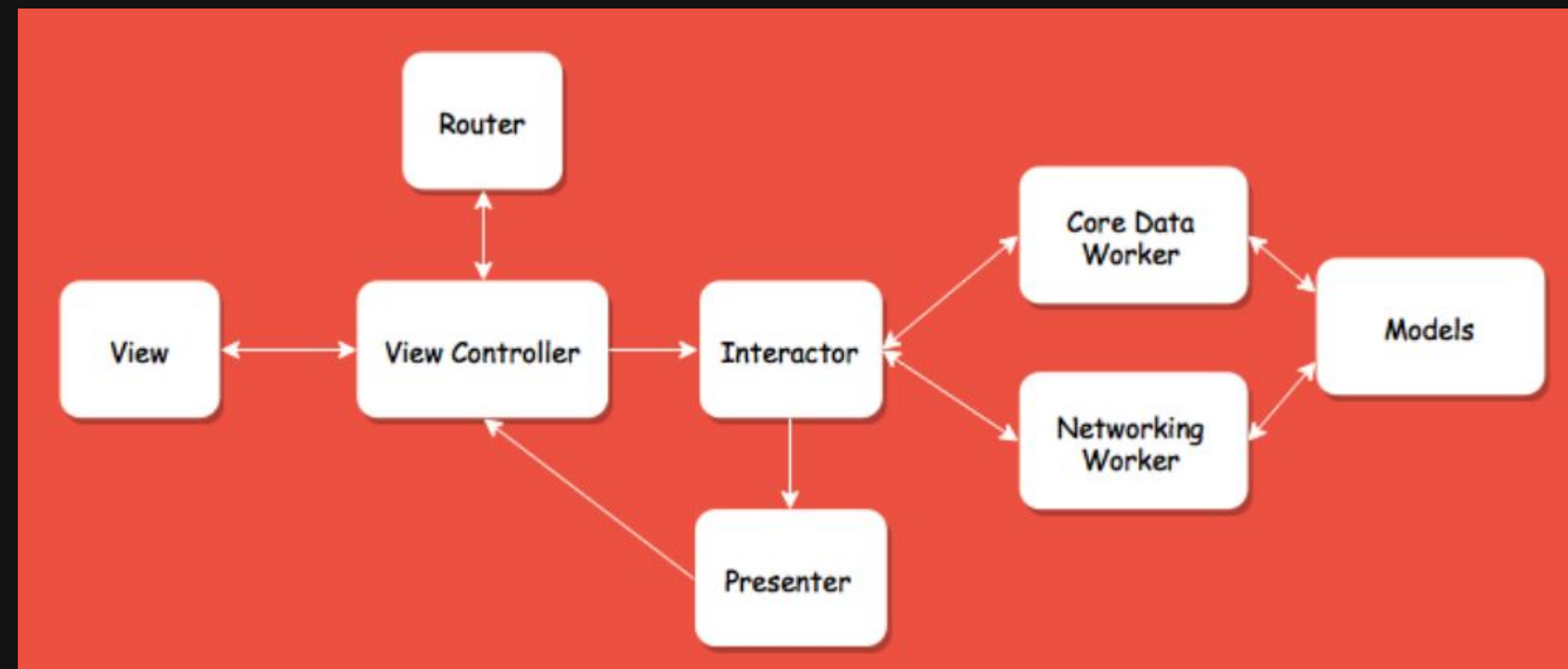




# Clean Swift

Каждый экран приложения - 1 сцена. Включает: Models, Router, Worker, Interactor, Presenter.

Их взаимодействие описано на схеме:



Что такое Clean swift архитектура:

<https://habr.com/ru/post/453986/>

Clean Swift на русском, хорошая статья с примерами:

<https://medium.com/@mr.wizman/забудьте-mvc-представляю-архитектуру-clean-swift-vip-a28797ef6f9b>



# Redux

Архитектура, пришедшая из JavaScript, применяется в SwiftUI. Ключевое понятие - **State**, источник правды, который хранит информацию. Изменяется только через вызов actions. Каждый раз при изменении состояния все наблюдатели получают уведомление.

Поток данных - **однонаправленный**, в каждый компонент данные приходят только одним способом.

Остальные понятия:

**Actions** (действия) - декларативный способ описания изменения состояния.

**Reducers** (обработчики) - **чистая функция**, принимает Action и State, и возвращает State.

**Store** (хранилище) - объект который содержит State и предоставляет объекты для его обновления.

**View** (представления) - **не имеют собственного состояния**, могут подписываться на изменения глобального состояния и описывать реакцию на эти изменения.

Redux в swift:

<https://habr.com/ru/post/500158/>

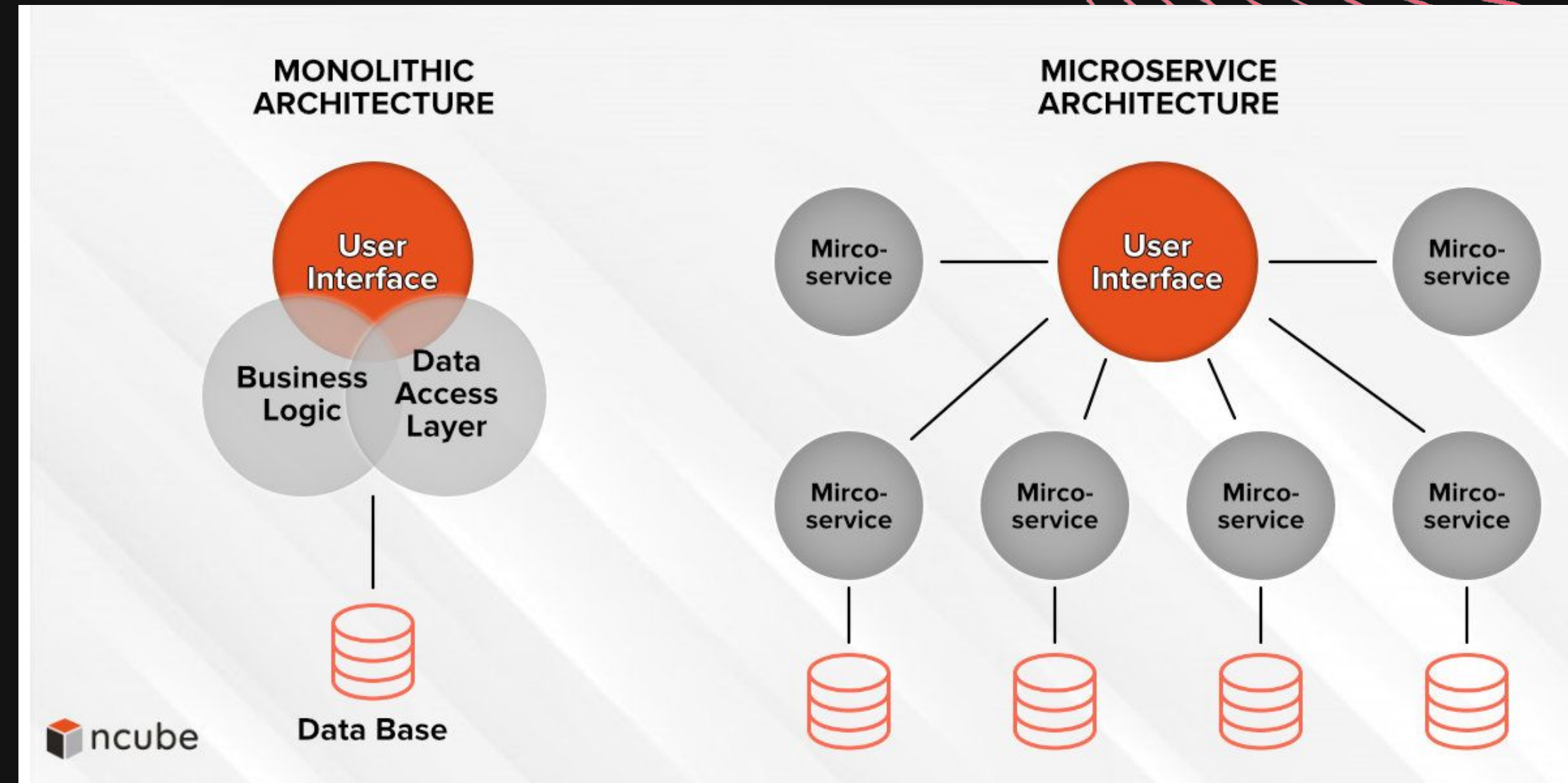
<http://ovchinnikov.cc/writing/redux-intro/>





# Microservices architecture in iOS

Каждый экран - микросервис (отдельное приложение), написанный на MVVM с биндингом. Каждый микросервис подтягивает микросервисы более низкого уровня. Например, Components (набор элементов интерфейса), который в свою очередь подтягивает более низкоуровневые микросервисы. Все взаимодействие осуществляется через CocoaPods, при этом публичные библиотеки имеют последнее место в цепи микросервисов.



В чем преимущества микросервисной архитектуры:

<https://habr.com/ru/company/mailru/blog/320962/>

Разбор, почему не стоит ее использовать:

<https://habr.com/ru/post/427215/>

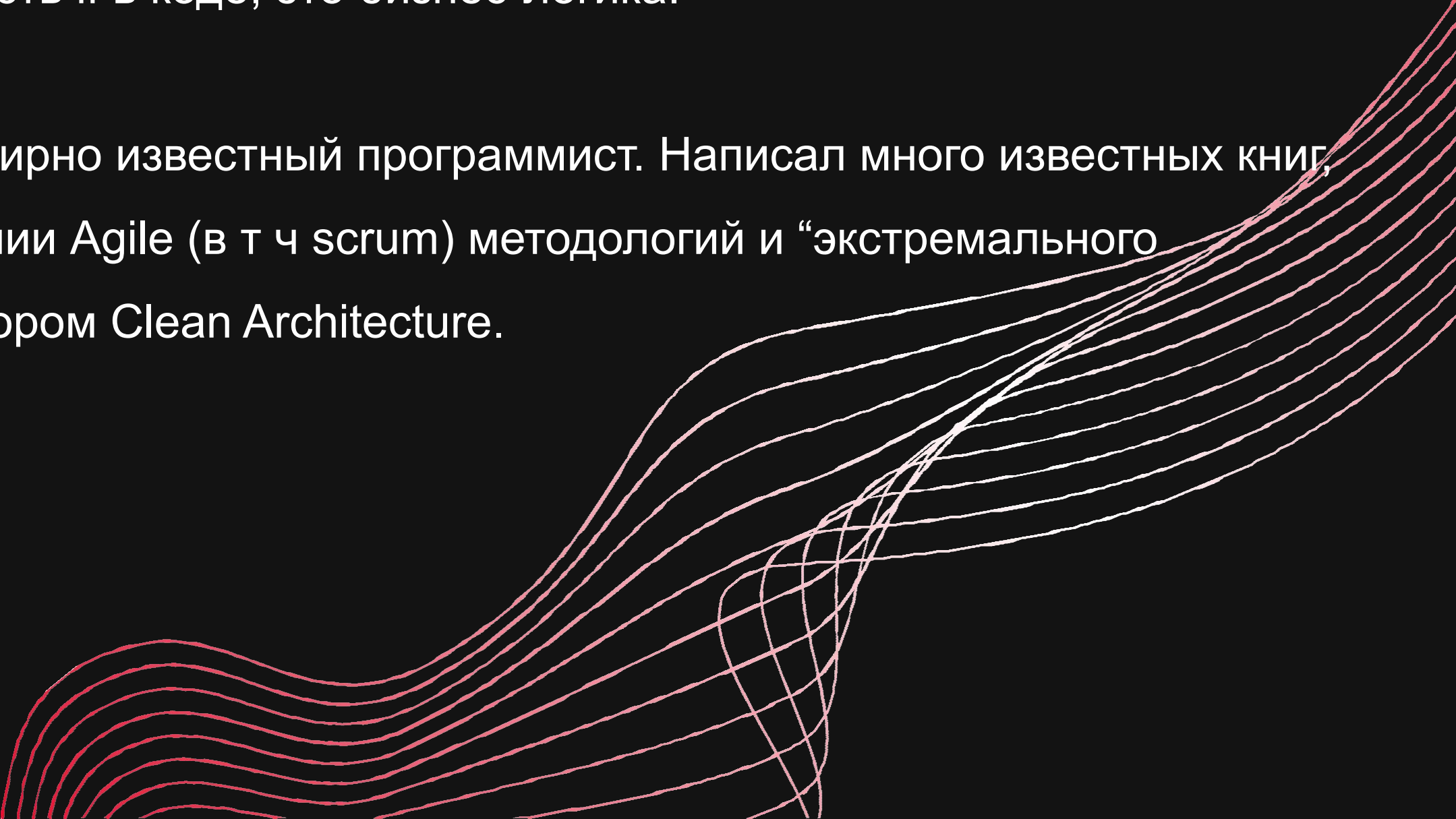


# Термины:

**Источник правды** - единственное место, в котором могут меняться данные в программе, которое является источником данных для всей программы.

**Бизнес логика** - все, что **происходит** в вашем приложении. Или логика, описывающая работу всех процессов что есть в программе. К примеру если у нас есть if в коде, это бизнес-логика.

**Uncle Bob (Дядя Боб)** - Роберт Мартин, всемирно известный программист. Написал много известных книг, описал принципы SOLID, участвовал в создании Agile (в т ч scrum) методологий и “экстремального программирования”. Кроме того является автором Clean Architecture.





# Все ссылки:

Статья с разбором MVC у Apple и объяснением, что с ним не так:

<https://habr.com/ru/post/324414/>

(очень популярный вопрос на собеседованиях)

MVVM в большинстве случаев используют с биндингом. Биндинг слишком обширная тема чтобы вставлять ее в наш курс. Тем не менее самый простой вариант работы с биндингами - библиотека Bond (если интересует более глубокий подход - почитайте об RxSwift). С библиотекой Bond можно ознакомиться в этом tutorialе:

<https://www.raywenderlich.com/667-bond-tutorial-bindings-in-swift>

Хороший и подробный разбор VIPER. Есть простой пример и сложный пример:

<https://habr.com/ru/post/358412/>

Что такое Clean swift архитектура:

<https://habr.com/ru/post/453986/>

Clean Swift на русском, хорошая статья с примерами:

<https://medium.com/@mr.wizman/забудьте-mvc-представляю-архитектуру-clean-swift-vip-a28797ef6f9b>

Redux в swift:

<https://habr.com/ru/post/500158/>

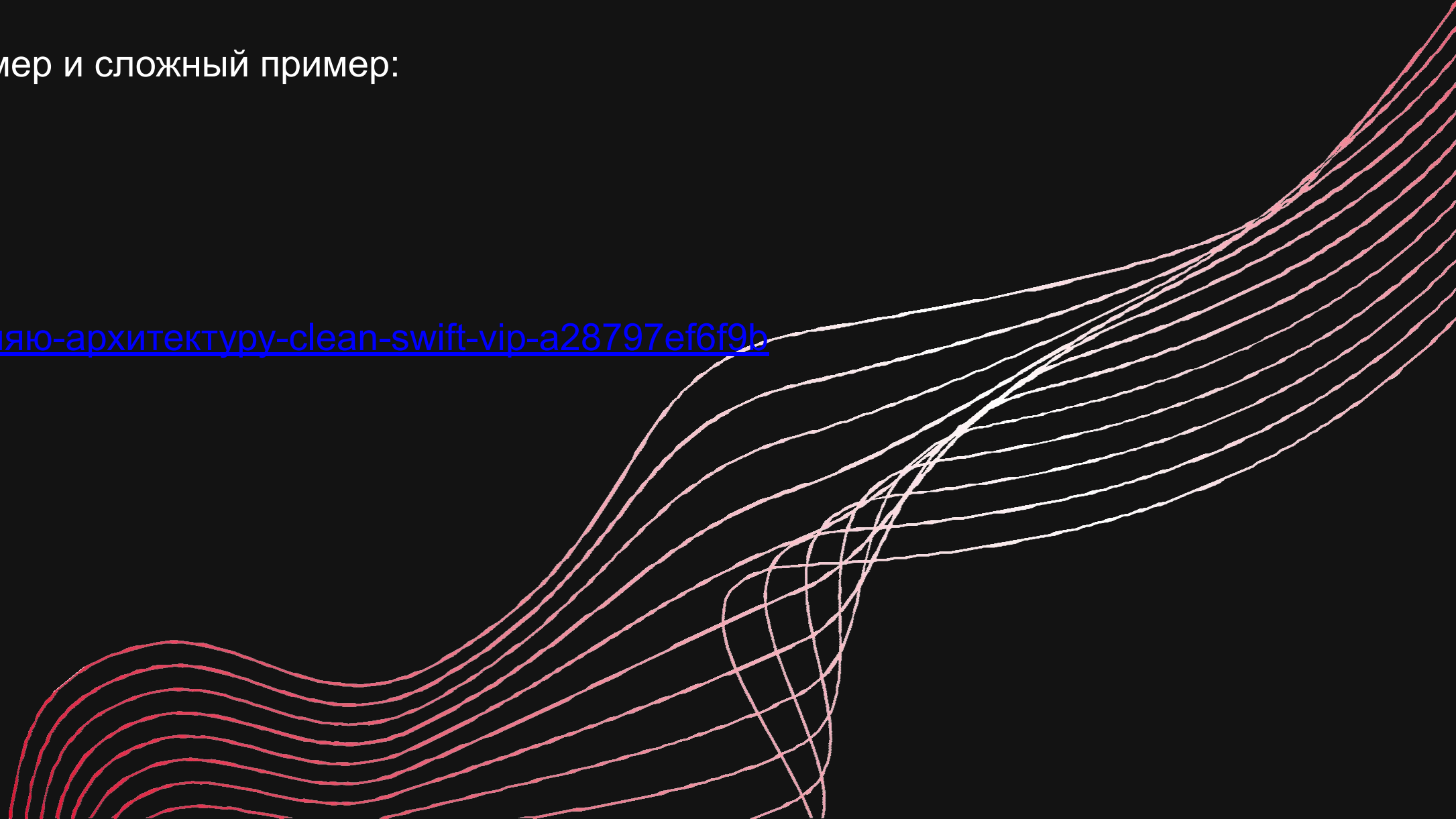
<http://ovchinnikov.cc/writing/redux-intro/>

В чем преимущества микросервисной архитектуры:

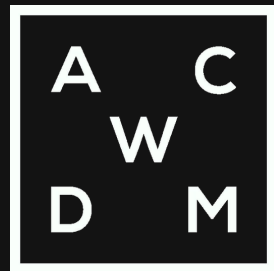
<https://habr.com/ru/company/mailru/blog/320962/>

Разбор, почему не стоит ее использовать:

<https://habr.com/ru/post/427215/>







Web  
Academy

# CREATE YOUR IT FUTURE.

WITH WEB ACADEMY

