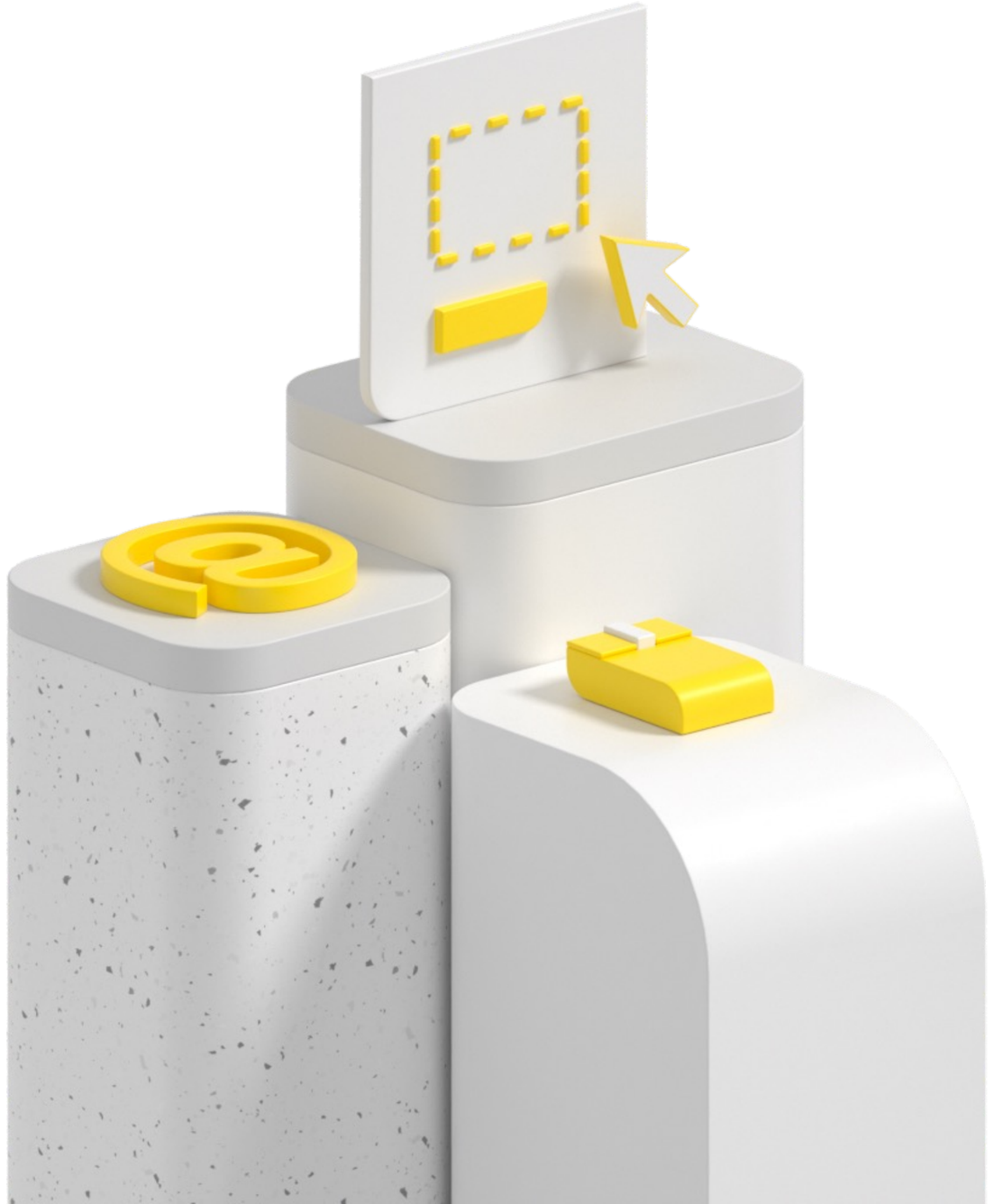
IT's TINKOFF

DI в многомодульном iOS приложении



IT's TINKOFF

Володин Кирилл

volodin.kirill.a@gmail.com

Telegram: @leoniknik

План доклада

- Модуляризация
- Выбор DI фреймворка
- Needle
- Многомодульное приложение
- Тестирование

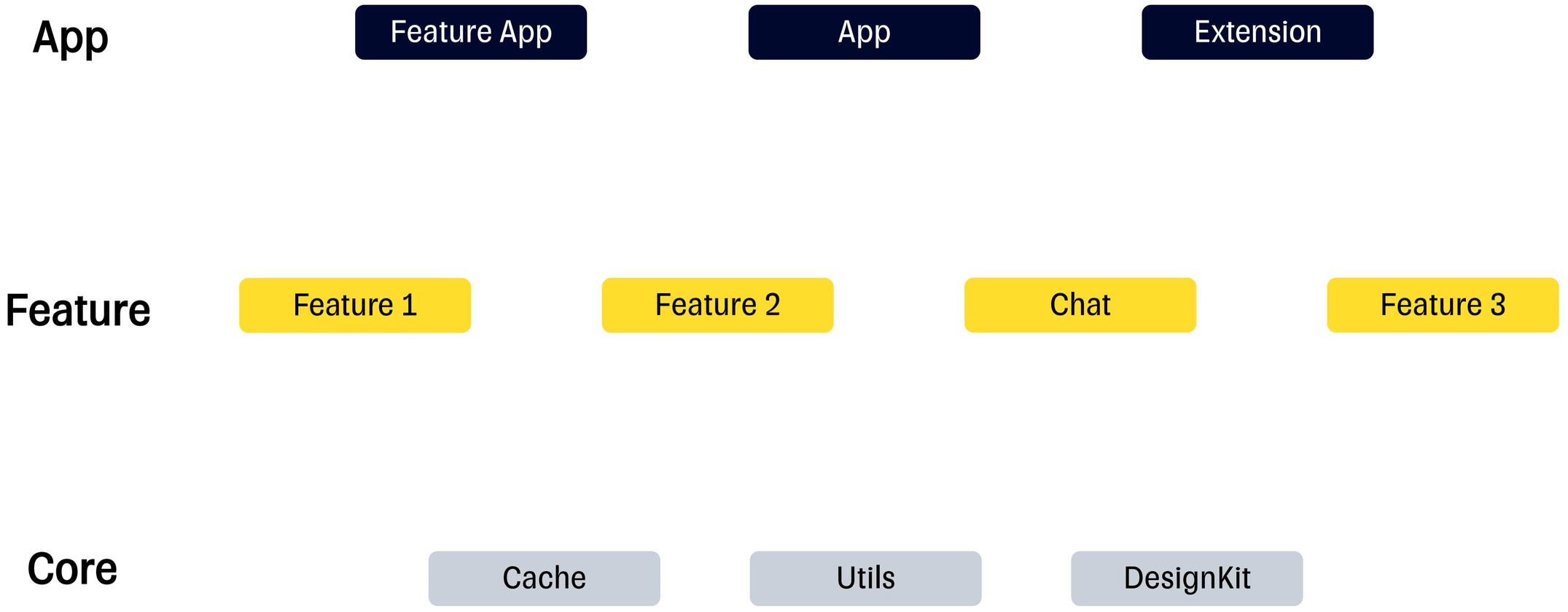
План доклада

- **Модуляризация**
- Выбор DI фреймворка
- Needle
- Многомодульное приложение
- Тестирование

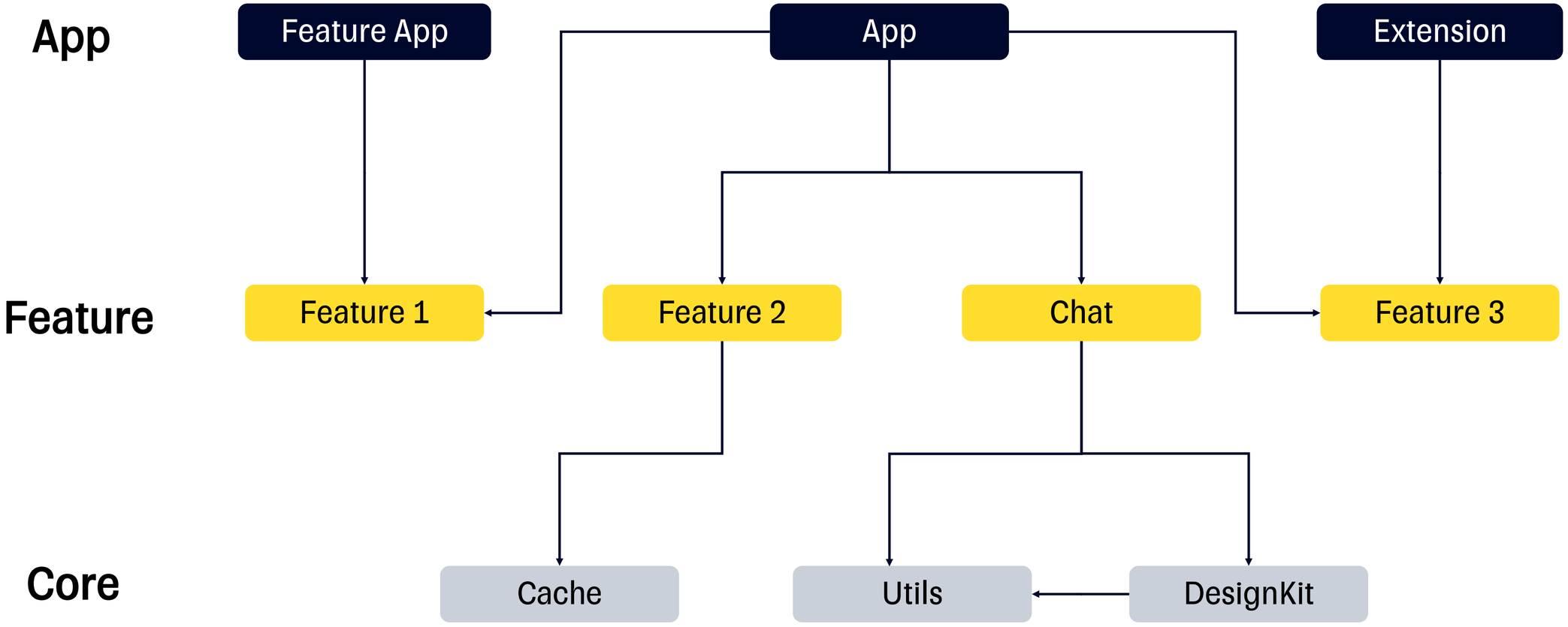
Зачем

- Ускорение сборки приложения
- Ускорение разработки
- Уменьшение связанности компонентов
- Разделение ответственности (Code Owners)
- Переиспользование модулей в приложениях
- Impact Analysis и т.д.

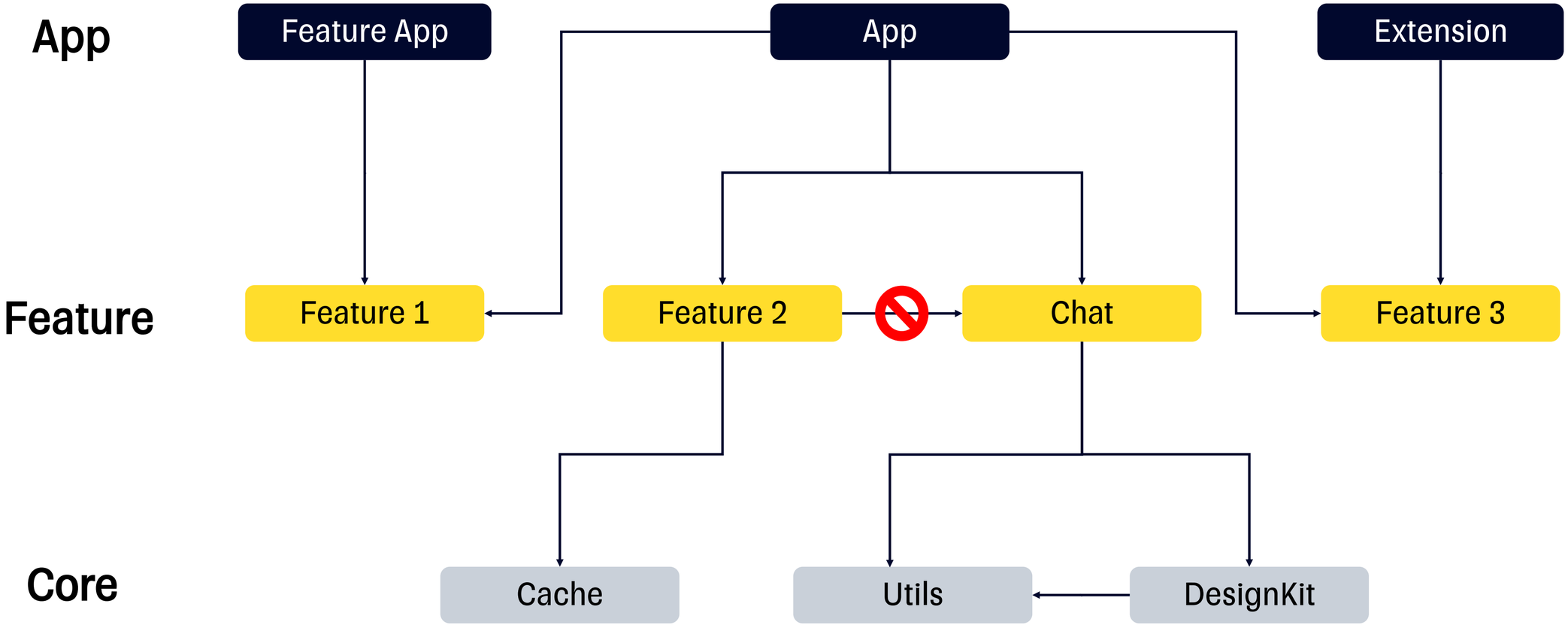
Типы модулей



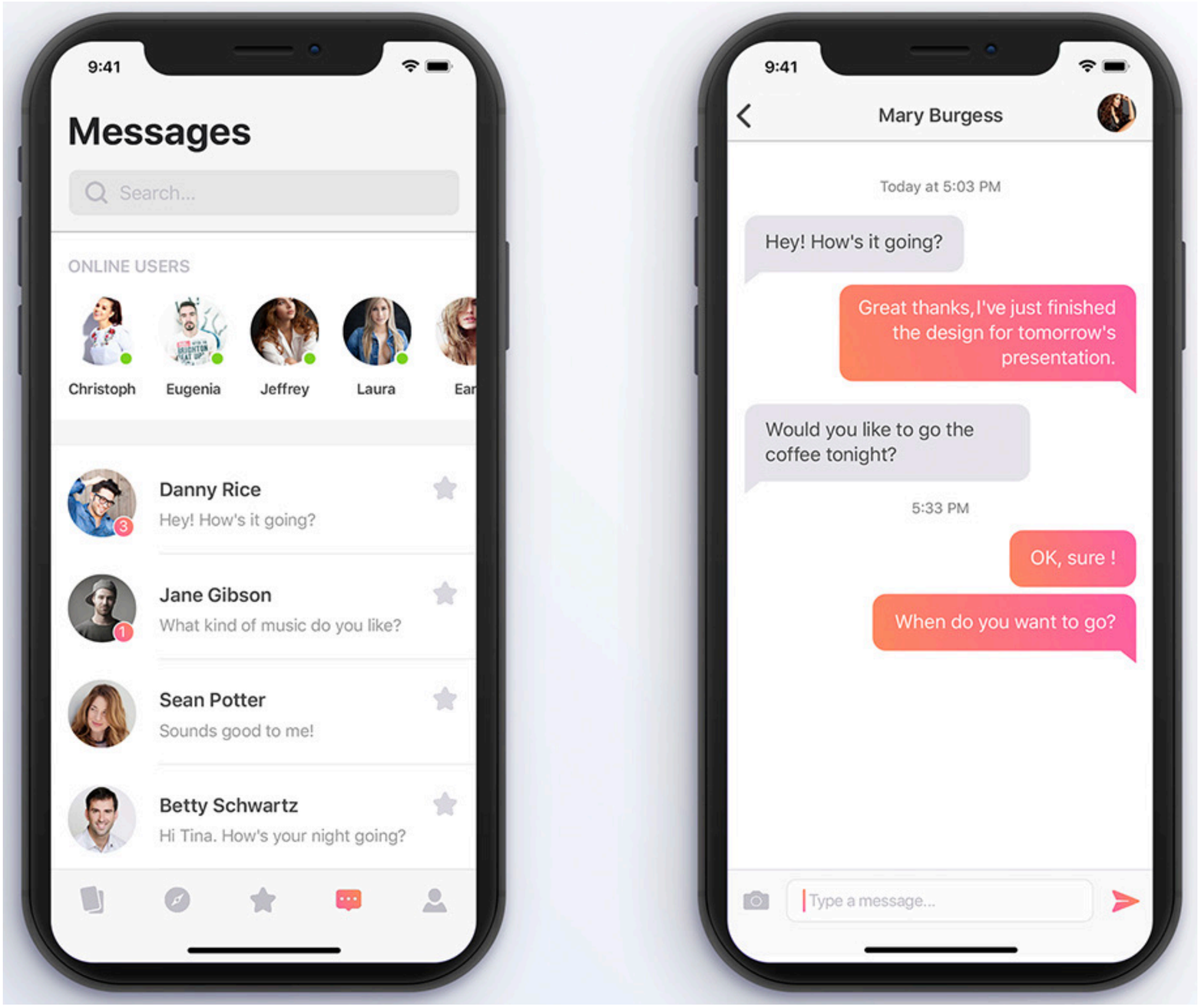
Многомодульное приложение



Многомодульное приложение



App


9:41
Messages
Search...
ONLINE USERS
Christoph
Eugenia
Jeffrey
Laura
Danny Rice
Hey! How's it going?
Jane Gibson
What kind of music do you like?
Sean Potter
Sounds good to me!
Betty Schwartz
Hi Tina. How's your night going?
9:41
Mary Burgess
Today at 5:03 PM
Hey! How's it going?
Great thanks, I've just finished the design for tomorrow's presentation.
Would you like to go the coffee tonight?
5:33 PM
OK, sure !
When do you want to go?
Type a message...

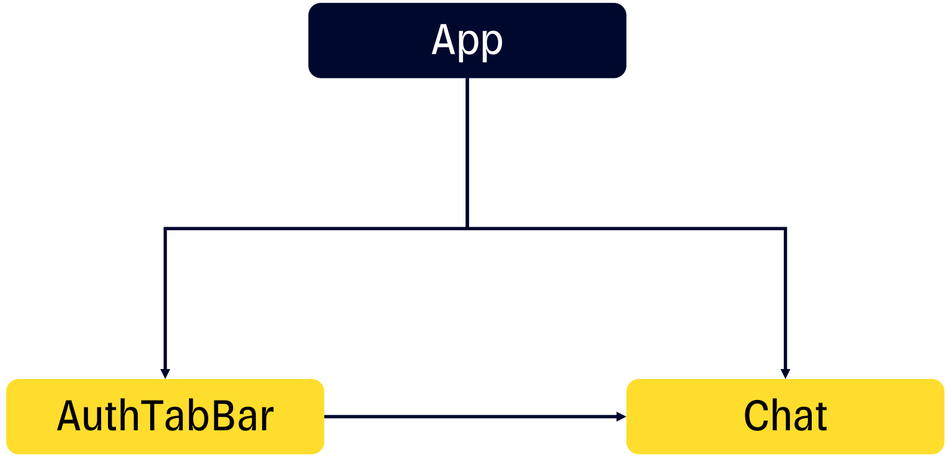
Межмодульные связи

```swift
import Chat

class AuthTabBarCoordinator {

    private let chatCoordinatorAssembly: IChatCoordinatorAssembly

    init(chatCoordinatorAssembly: IChatCoordinatorAssembly) {
        self.chatCoordinatorAssembly = chatCoordinatorAssembly
    }

    func setup() {
        let coordinator = chatCoordinatorAssembly.assemble()
        // ..
    }
}
```

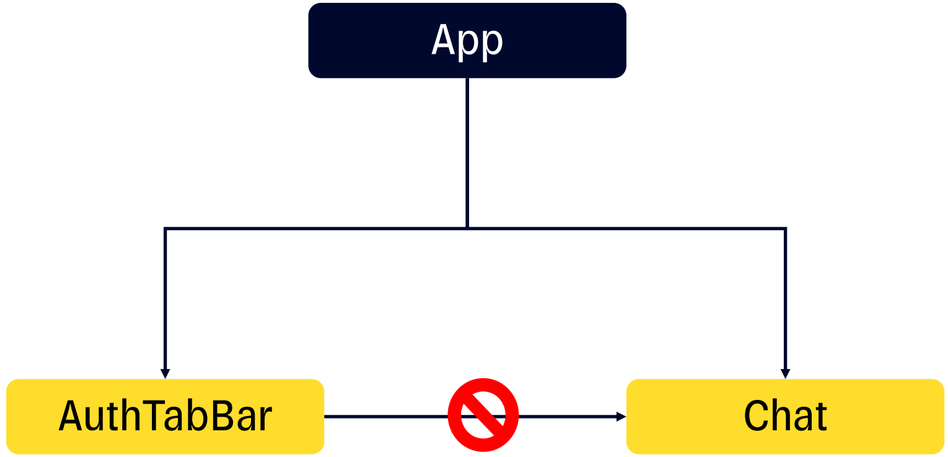
Межмодульные связи

```swift
import Chat

class AuthTabBarCoordinator {

    private let chatCoordinatorAssembly: IChatCoordinatorAssembly

    init(chatCoordinatorAssembly: IChatCoordinatorAssembly) {
        self.chatCoordinatorAssembly = chatCoordinatorAssembly
    }

    func setup() {
        let coordinator = chatCoordinatorAssembly.assemble()
        // ..
    }
}
```

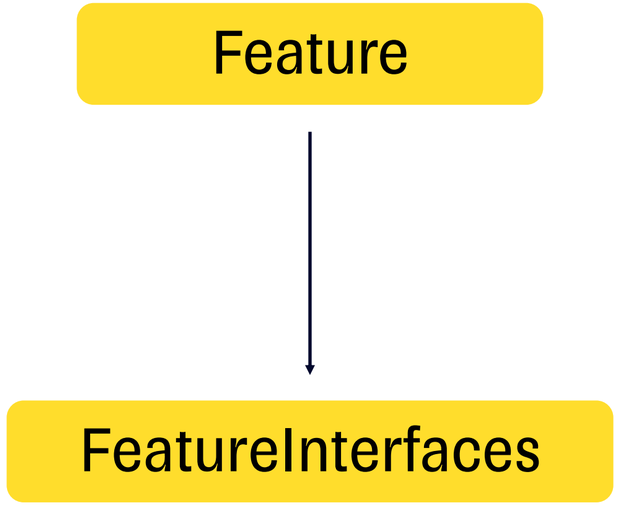
Межмодульные связи



Межмодульные связи



Модуль



Модуль Chat

IChatCoordinator

IChatCoordinatorAssembly

ISomeClass

ChatCoordinator

ChatCoordinatorAssembly

SomeClass

Модуль Chat

IChatCoordinator

IChatCoordinatorAssembly

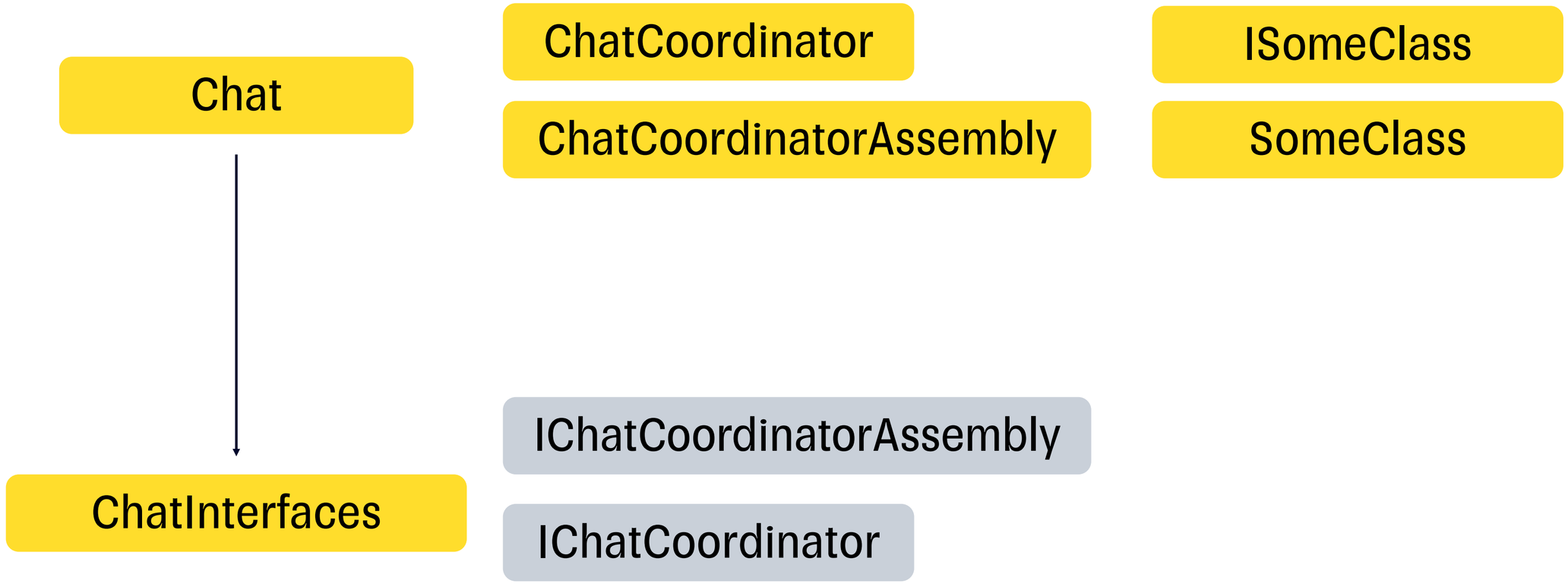
ISomeClass

ChatCoordinator

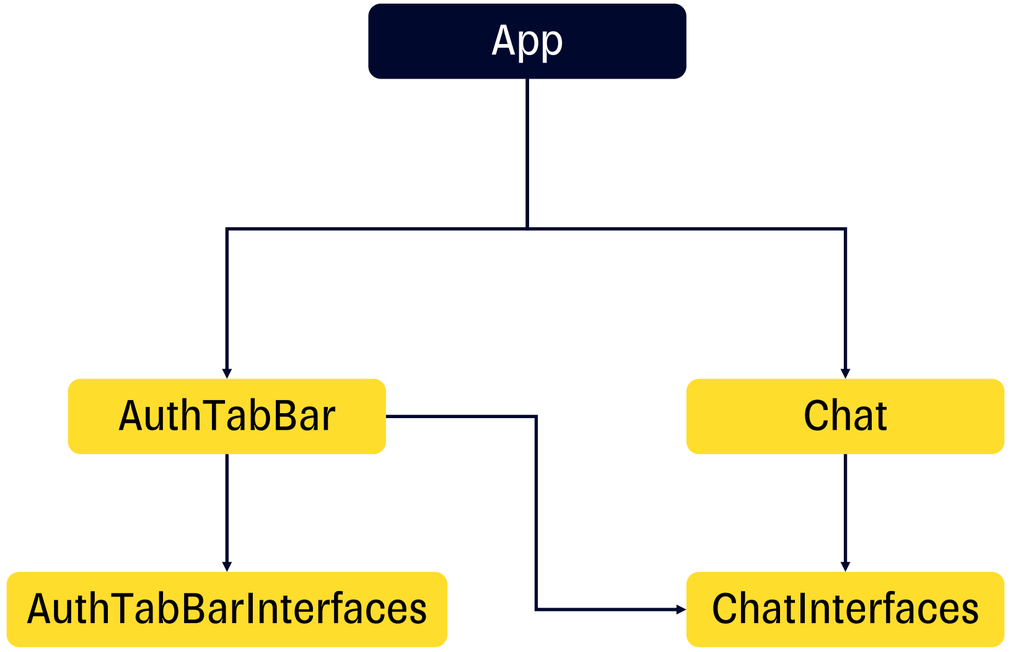
ChatCoordinatorAssembly

SomeClass

Модуль


Chat
ChatCoordinator
ChatCoordinatorAssembly
ISomeClass
SomeClass
ChatInterfaces
IChatCoordinatorAssembly
IChatCoordinator

Межмодульные связи



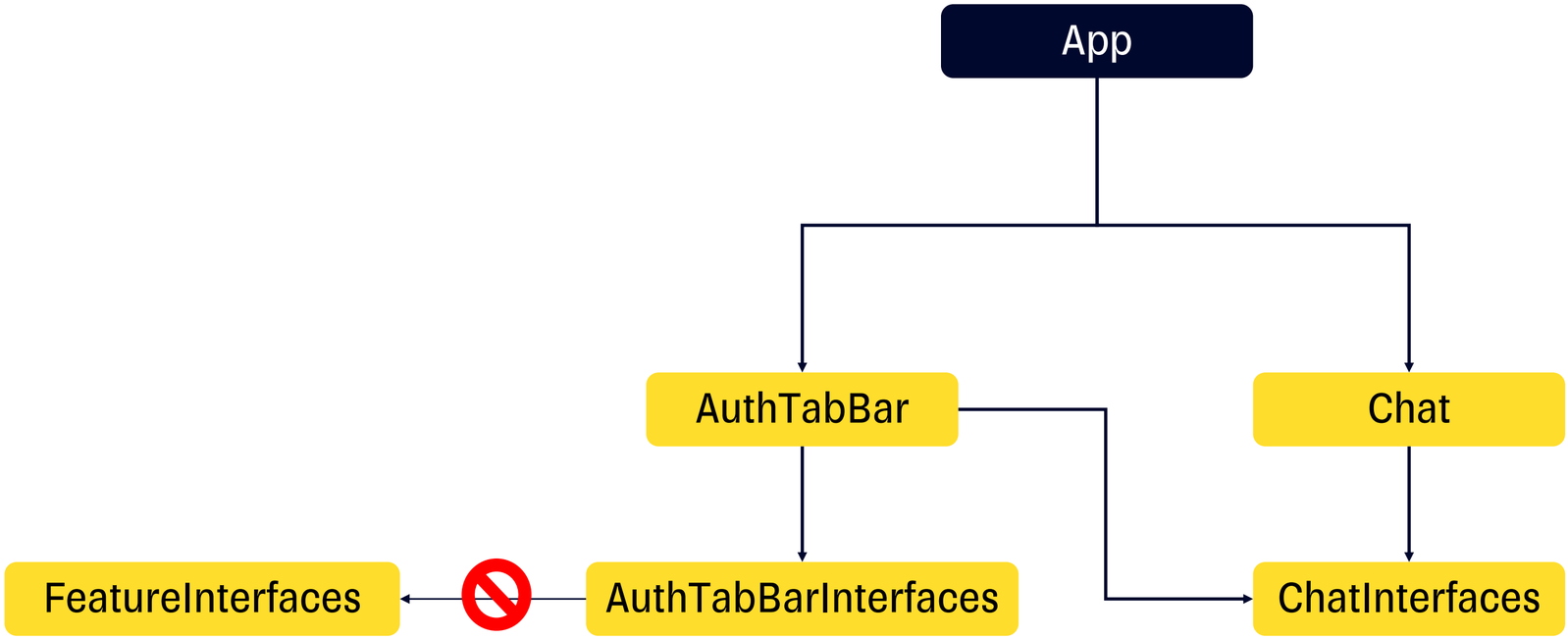
Межмодульные связи

```swift
import ChatInterfaces

class AuthTabBarCoordinator {

    private let chatCoordinatorAssembly: IChatCoordinatorAssembly

    init(chatCoordinatorAssembly: IChatCoordinatorAssembly) {
        self.chatCoordinatorAssembly = chatCoordinatorAssembly
    }

    func setup() {
        let coordinator = chatCoordinatorAssembly.assemble()
        // ..
    }
}
```

Межмодульные связи

```swift
import ChatInterfaces

class AuthTabBarCoordinator {

    private let chatCoordinatorAssembly: IChatCoordinatorAssembly

    init(chatCoordinatorAssembly: IChatCoordinatorAssembly) {
        self.chatCoordinatorAssembly = chatCoordinatorAssembly
    }

    func setup() {
        let coordinator = chatCoordinatorAssembly.assemble()
        // ..
    }
}
```

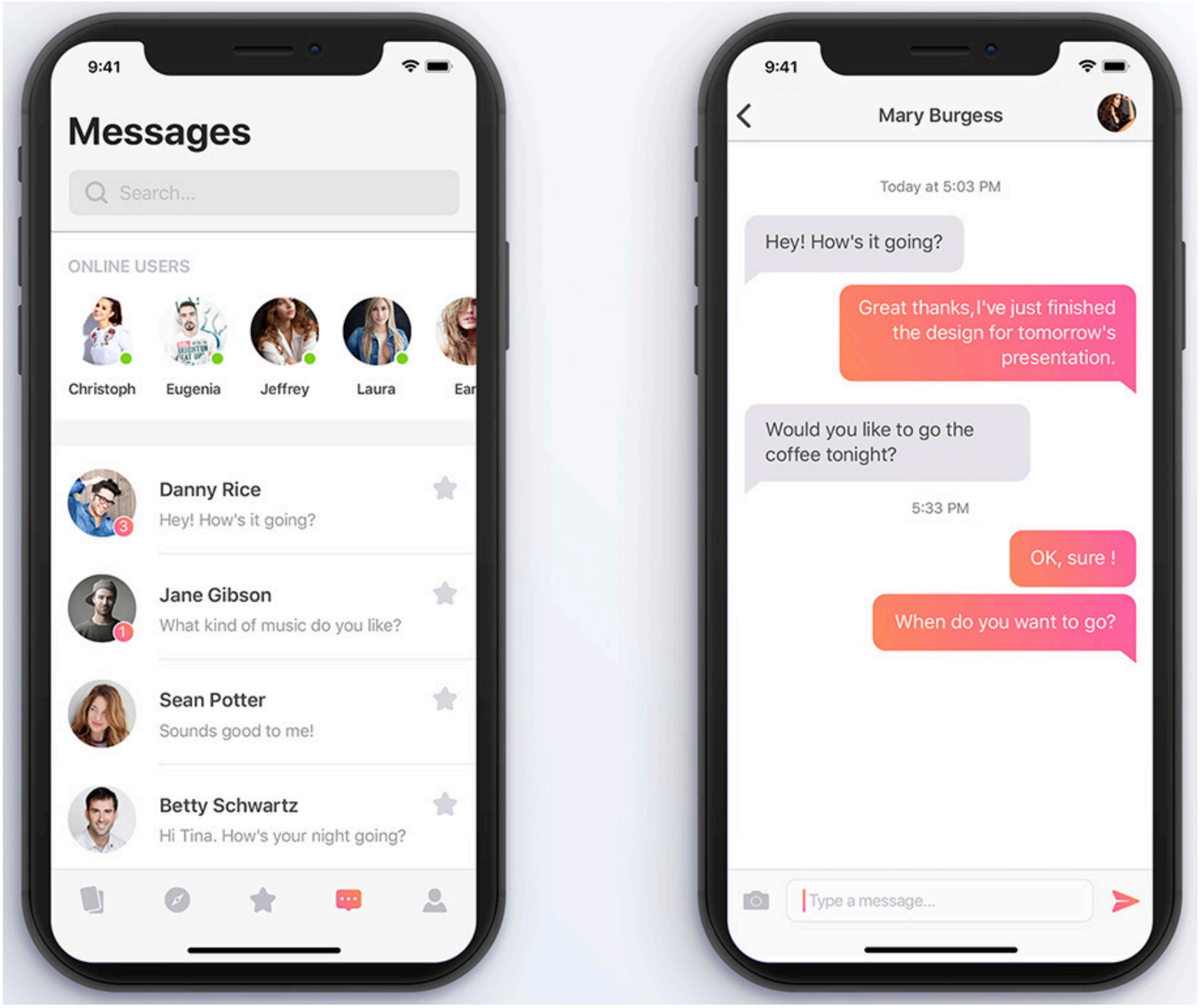
Многомодульное приложение



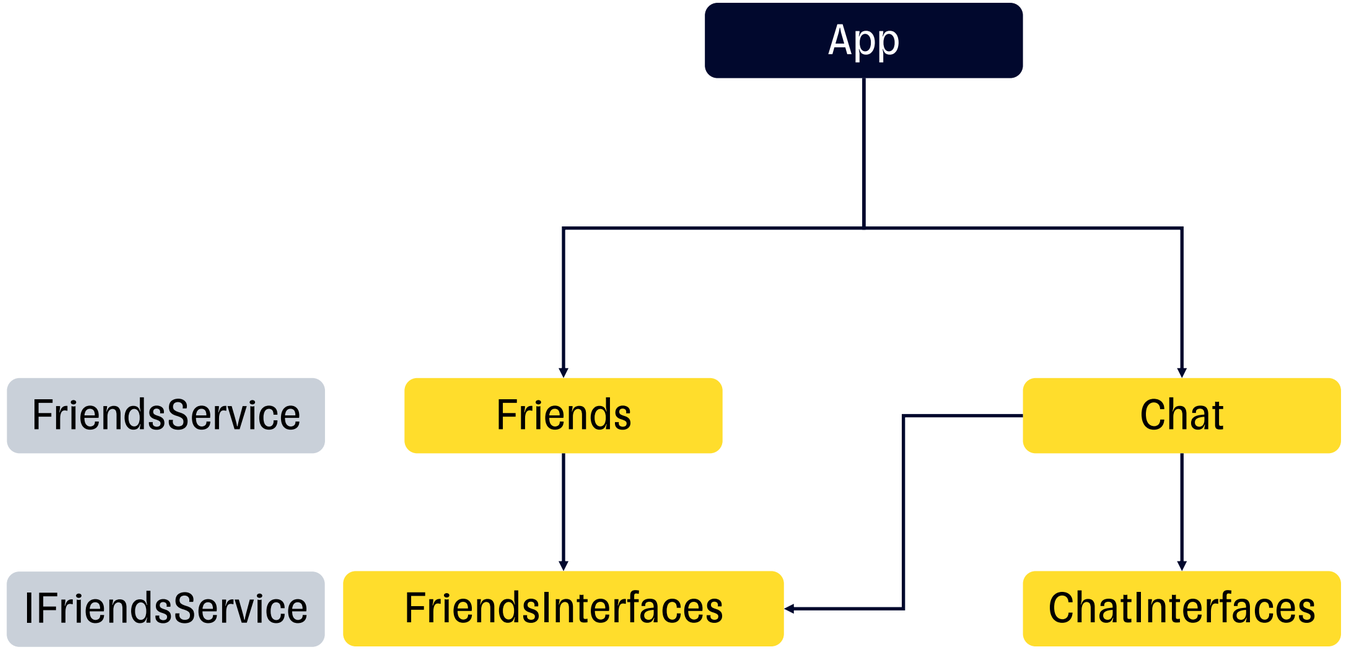
Задачи

- Инджектить в модули имплементации из других модулей
- Как и когда создавать эти сущности

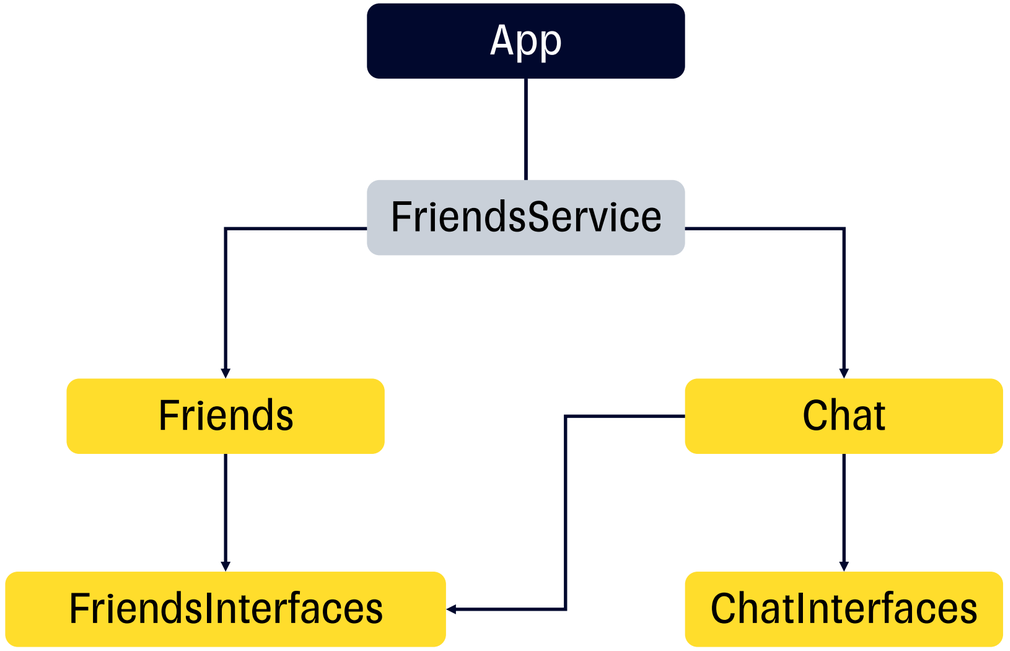
App


9:41
Messages
Search...
ONLINE USERS
Christoph
Eugenia
Jeffrey
Laura
Danny Rice
Hey! How's it going?
Jane Gibson
What kind of music do you like?
Sean Potter
Sounds good to me!
Betty Schwartz
Hi Tina. How's your night going?
9:41
Mary Burgess
Today at 5:03 PM
Hey! How's it going?
Great thanks, I've just finished the design for tomorrow's presentation.
Would you like to go the coffee tonight?
5:33 PM
OK, sure !
When do you want to go?
Type a message...

Многомодульное приложение



Многомодульное приложение



Задачи

- Инджектить в модули имплементации из других модулей
- Как и когда создавать эти сущности
- Управлять жизненным циклом

Dependency Injection

DI Container

- Регистрирует
- Создает
- Предоставляет
- Управляет жизненным циклом

План доклада

- Модуляризация
- **Выбор DI фреймворка**
- Needle
- Многомодульное приложение
- Тестирование

Критерии выбора

- Compile time
- Производительность (скорость генерации compile time кода)
- Поддержка тестирования
- Поддержка многомодульности
- Community и поддержка

DI фреймворки

- Swinject
- DIP
- DITranquillity
- Cleanse
- Resolver
- Weaver
- Needle
- Internal*

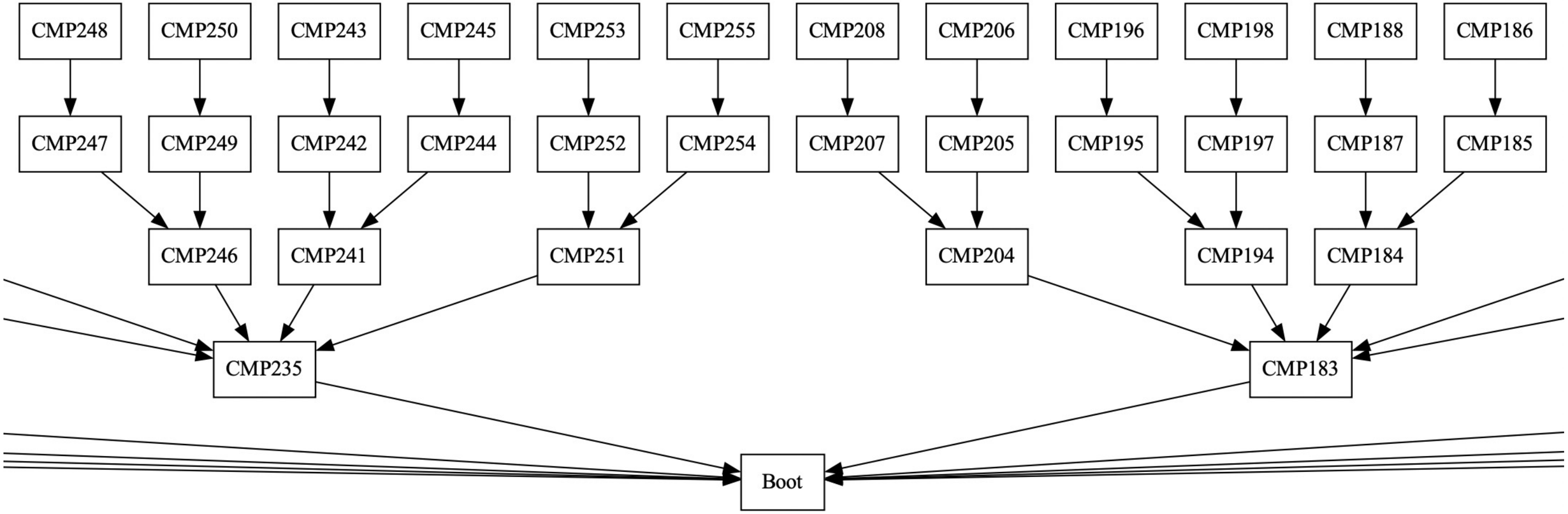
	Project	Stars
1	Swinject	5,376
2	Resolver	1,840
3	Cleanse	1,707
4	Needle	1,347
5	Weaver	665

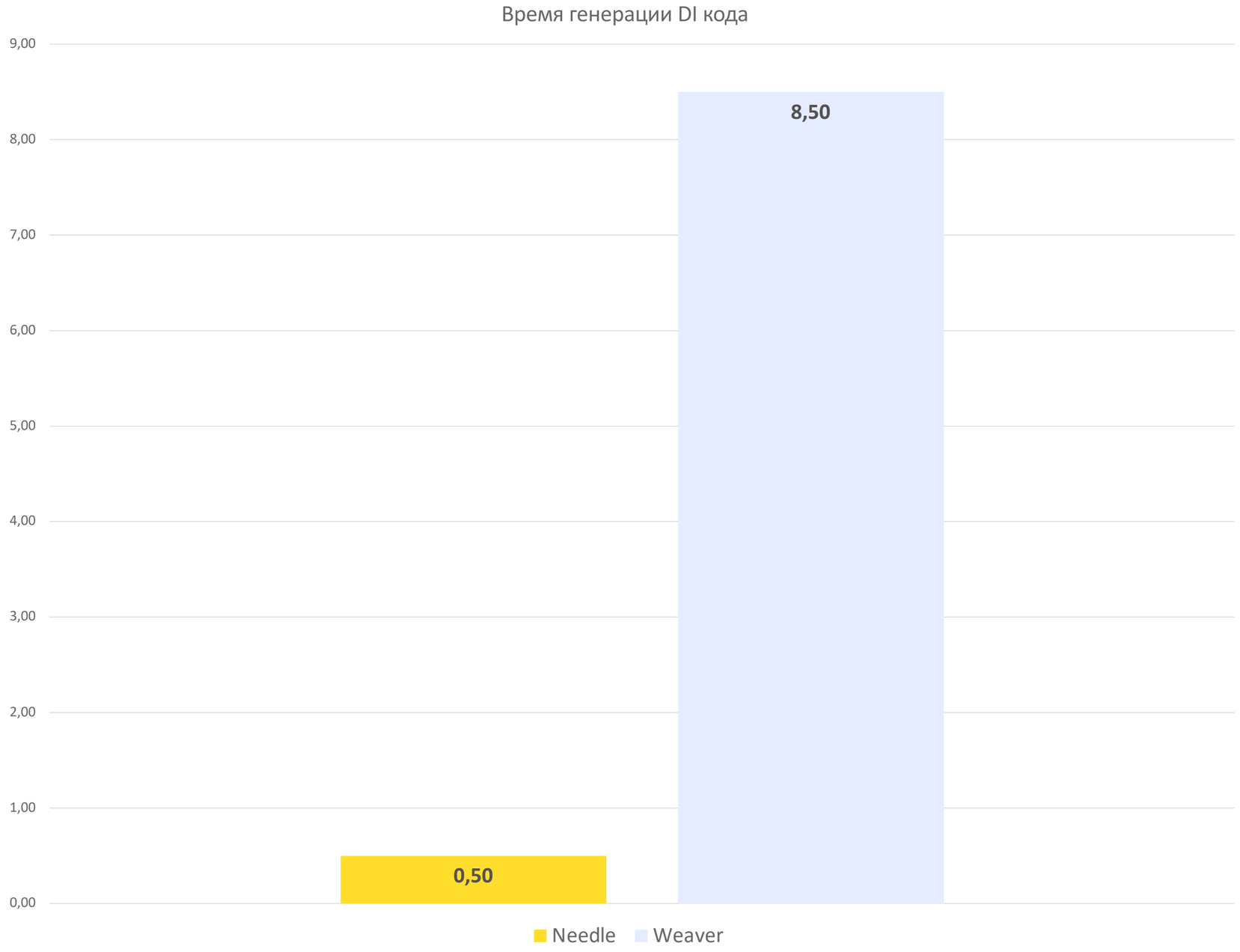
Compile time DI

Needle

Weaver

Производительность




Время генерации DI кода
9,00
8,00
7,00
6,00
5,00
4,00
3,00
2,00
1,00
0,00
8,50
0,50
Needle
Weaver

Сравнение

	Compile time	Производительность	Community
Swinject	❌	⚠️	✅
DIP	❌	⚠️	❌
DITranquillity	❌	⚠️	✅
Cleanse	⚠️	⚠️	❌
Resolver	❌	⚠️	⚠️
Weaver	✅	❌	✅
Needle	✅	✅	⚠️

План доклада

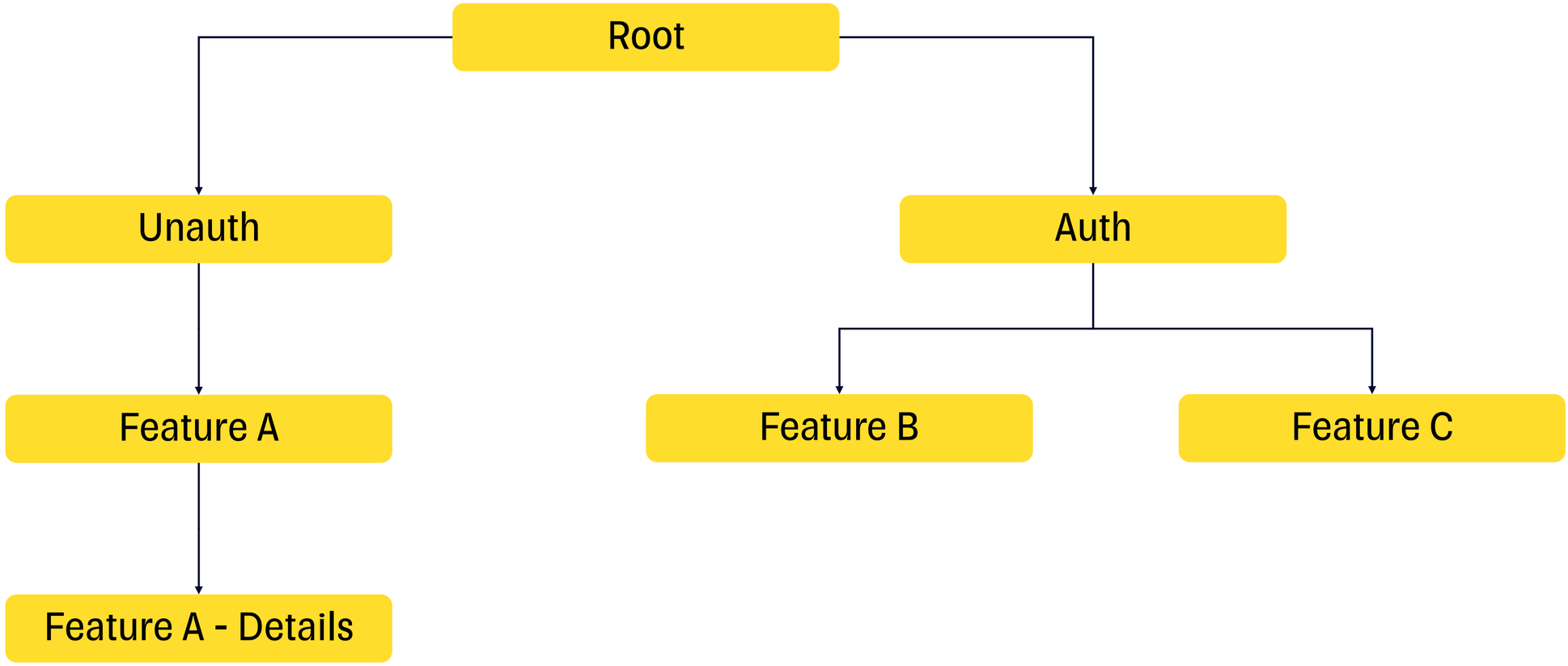
- Модуляризация
- Выбор DI фреймворка
- **Needle**
- Многомодульное приложение
- Тестирование

Needle

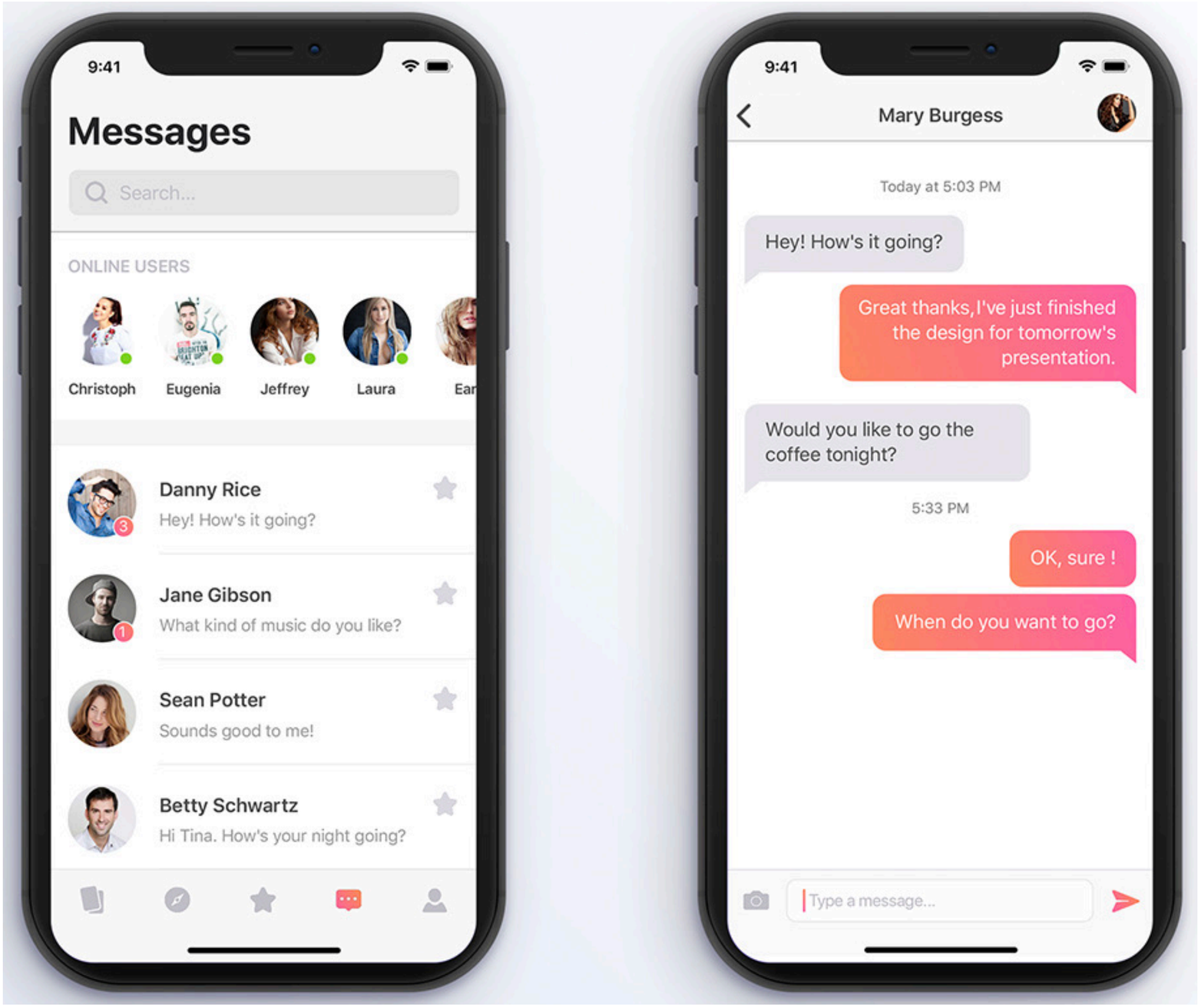
Framework

Generator

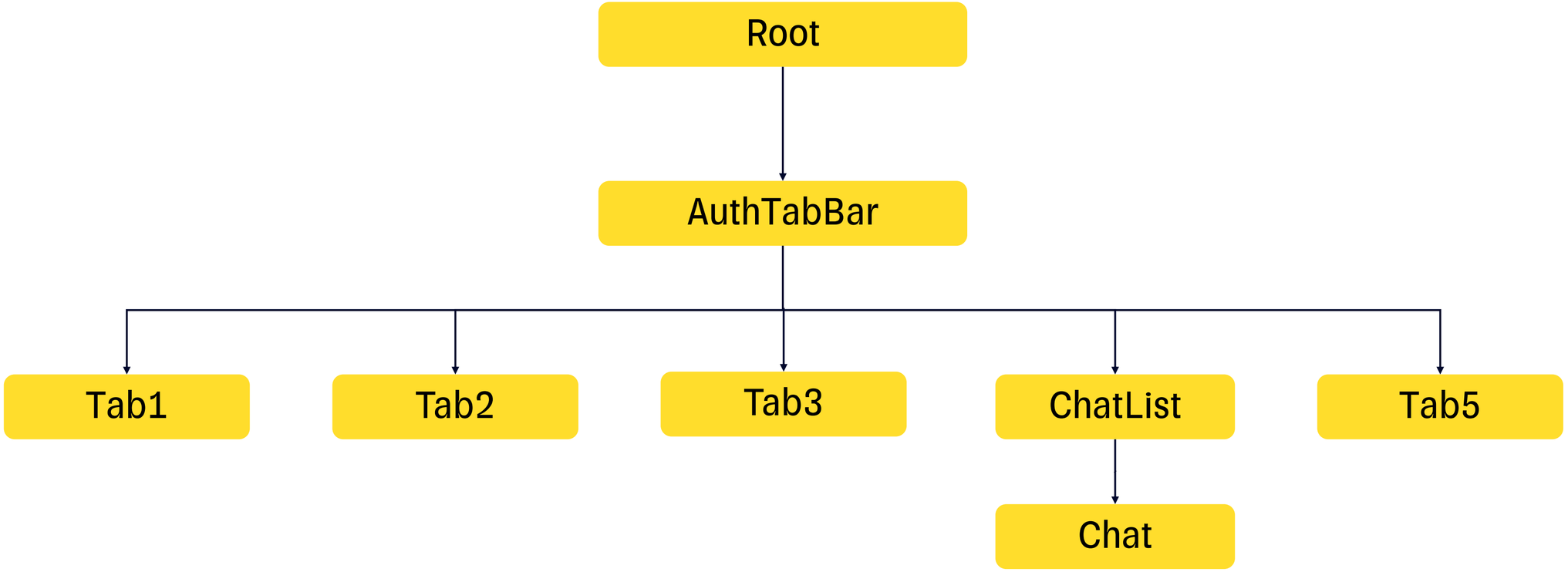
DI иерархичен - Scopes



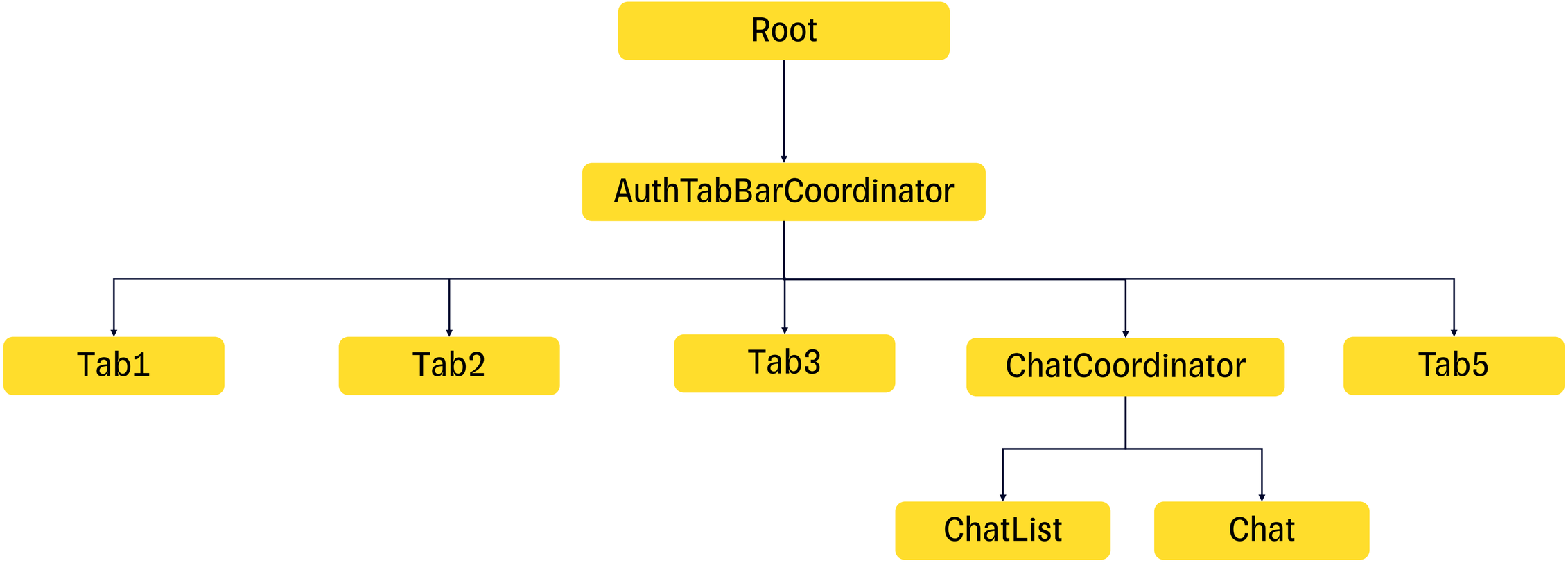
Scopes


9:41
Messages
Search...
ONLINE USERS
Christoph
Eugenia
Jeffrey
Laura
Danny Rice
Hey! How's it going?
Jane Gibson
What kind of music do you like?
Sean Potter
Sounds good to me!
Betty Schwartz
Hi Tina. How's your night going?
9:41
Mary Burgess
Today at 5:03 PM
Hey! How's it going?
Great thanks, I've just finished the design for tomorrow's presentation.
Would you like to go the coffee tonight?
5:33 PM
OK, sure !
When do you want to go?
Type a message...

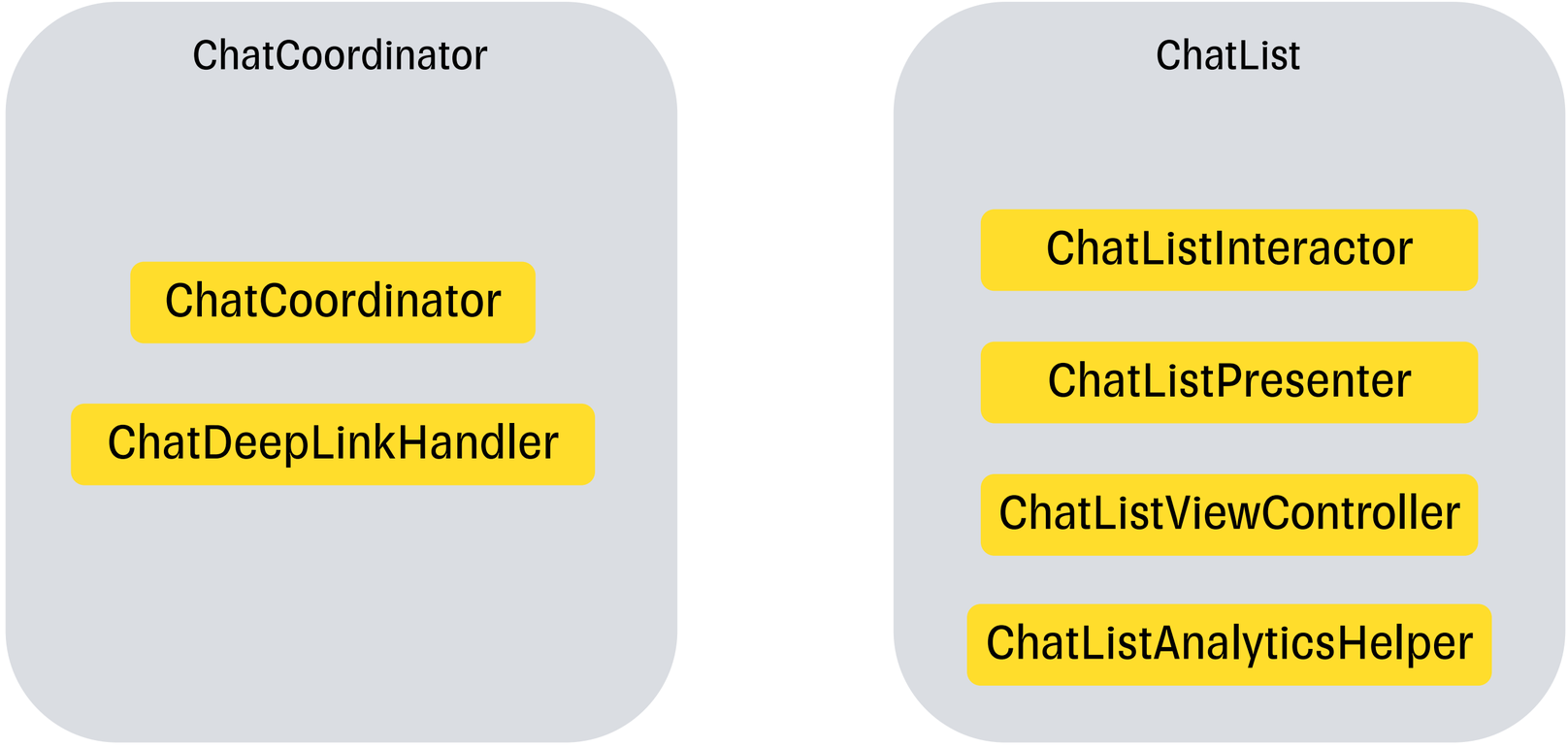
Scopes



Scopes


Root
AuthTabBarCoordinator
Tab1
Tab2
Tab3
ChatCoordinator
Tab5
ChatList
Chat

Scopes


ChatCoordinator
ChatCoordinator
ChatDeepLinkHandler
ChatList
ChatListInteractor
ChatListPresenter
ChatListViewController
ChatListAnalyticsHelper

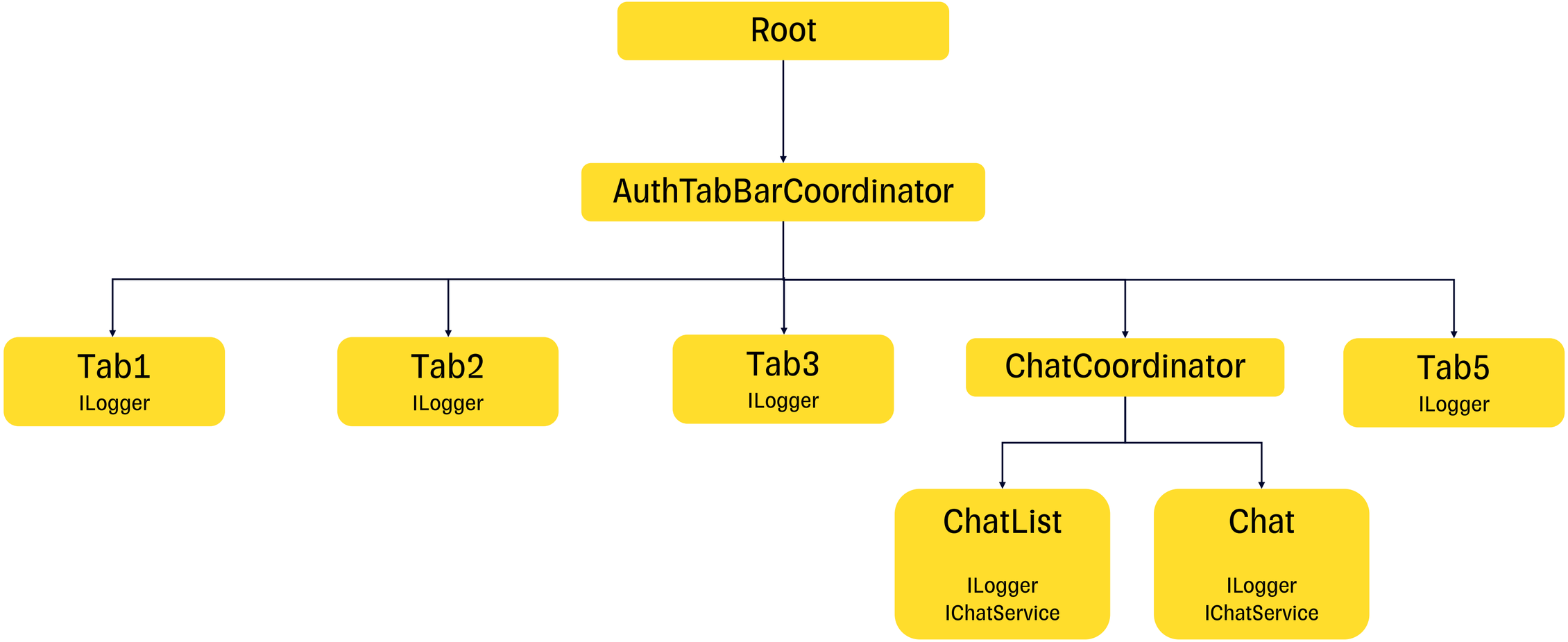
Scopes

```
class ChatListPresenter {
    private let logger: ILogger
    private let service: IChatService

    init(logger: ILogger, service: IChatService) {
        self.logger = logger
        self.service = service
    }
}
```

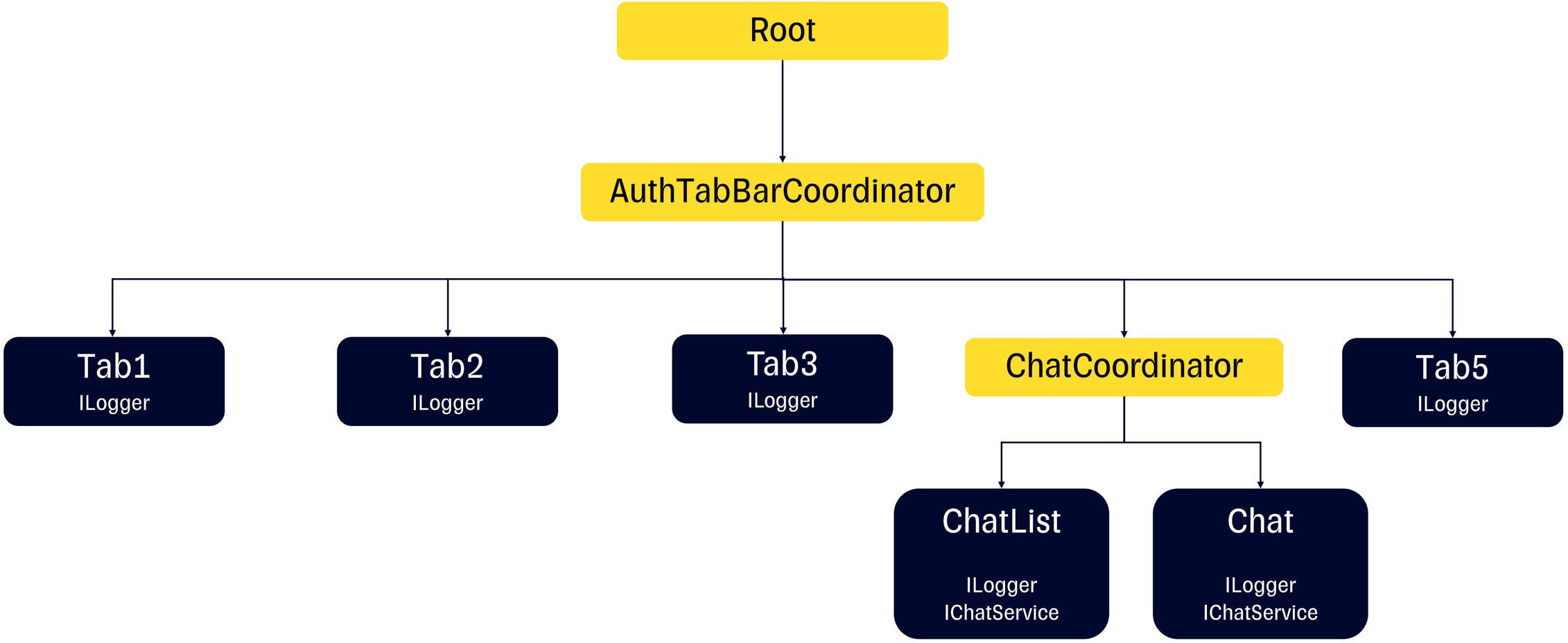
Scopes

```
class ChatPresenter {
    private let logger: ILogger
    private let service: IChatService

    init(logger: ILogger, service: IChatService) {
        self.logger = logger
        self.service = service
    }
}
```

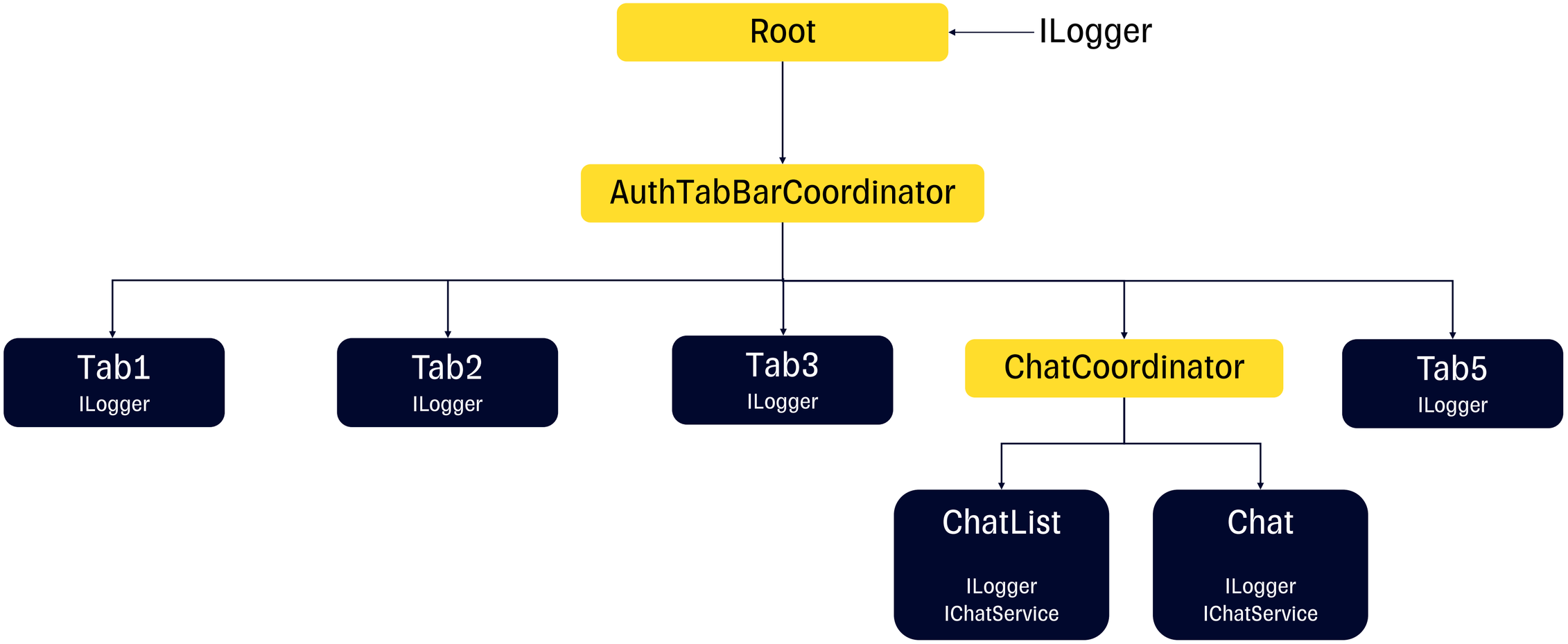
Scopes


Root
AuthTabBarCoordinator
Tab1
ILogger
Tab2
ILogger
Tab3
ILogger
ChatCoordinator
Tab5
ILogger
ChatList
ILogger
IChatService
Chat
ILogger
IChatService

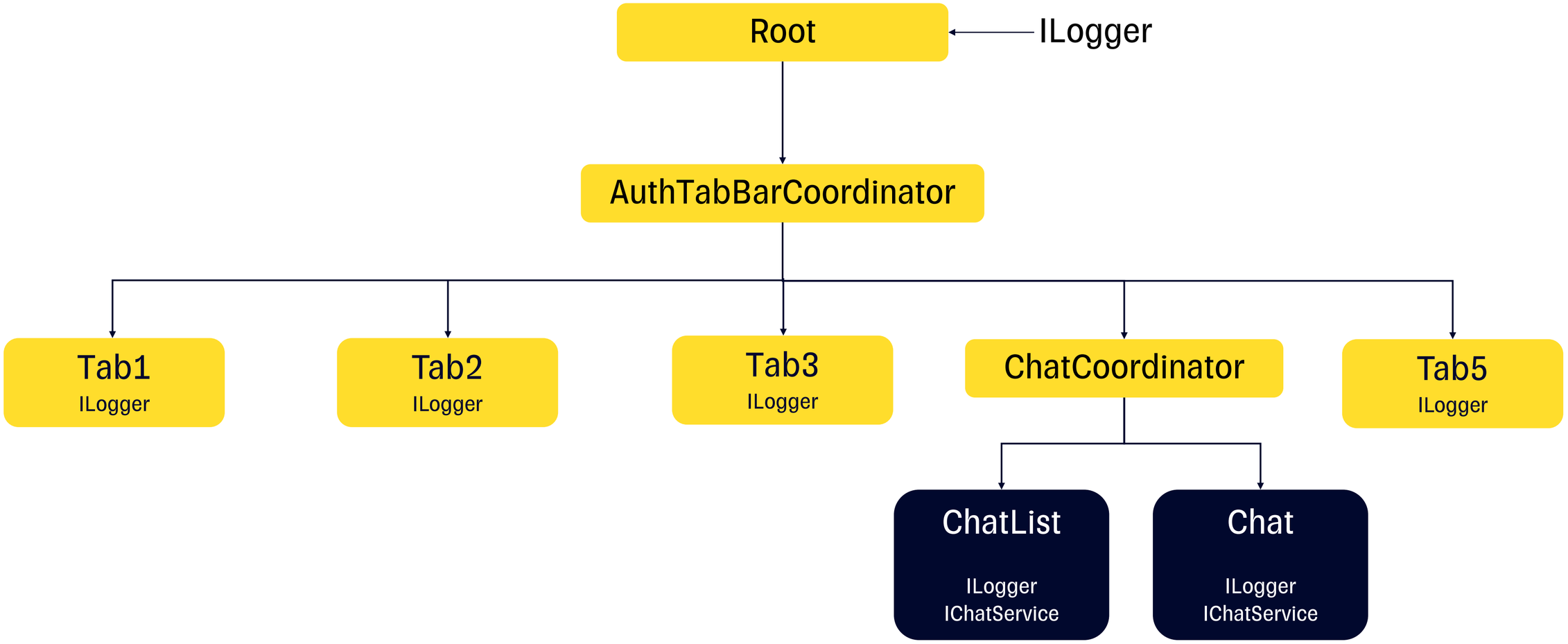
ILogger


Root
AuthTabBarCoordinator
Tab1
ILogger
Tab2
ILogger
Tab3
ILogger
ChatCoordinator
Tab5
ILogger
ChatList
ILogger
IChatService
Chat
ILogger
IChatService

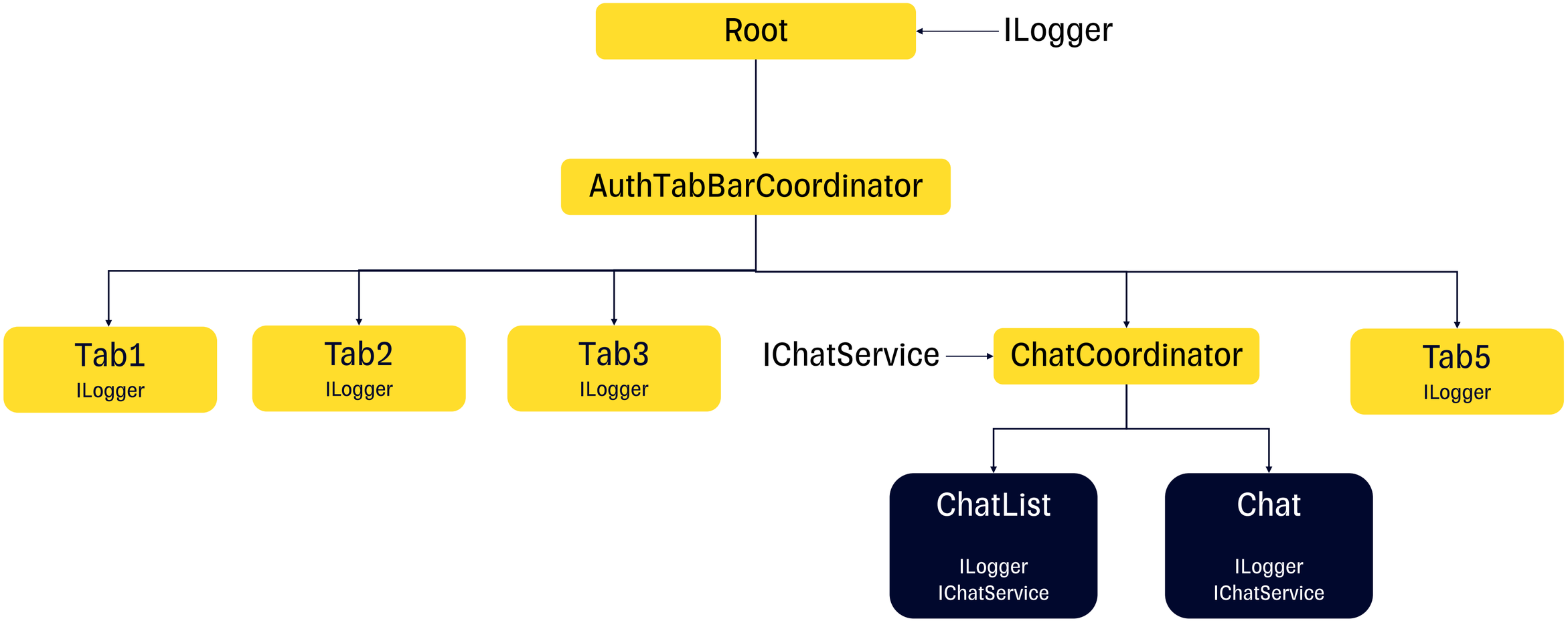
ILogger


Root
ILogger
AuthTabBarCoordinator
Tab1
ILogger
Tab2
ILogger
Tab3
ILogger
ChatCoordinator
Tab5
ILogger
ChatList
ILogger
IChatService
Chat
ILogger
IChatService

IChatService



IChatService



ChatList

```
class ChatListAssembly {
    init() { }

    func assemble() -> UIViewController {
        // ...
        let presenter = ChatListPresenter(
          logger: ILogger,
          service: IChatService
        )
        let viewController = ChatListViewController(presenter: presenter)
        // ...
        return presenter
    }
}
```

ChatList

```
1 class ChatListAssembly {
2     init() { }
3
4     func assemble() -> UIViewController {
5         // ...
6         let presenter = ChatListPresenter(
7           logger: ILogger,
8           service: IChatService
9         )
10        let viewController = ChatListViewController(presenter: presenter)
11        // ...
12        return presenter
13    }
14 }
```

Component и Dependency

```
protocol ChatListDependency: Dependency {
    // ...
}

class ChatListComponent: Component<ChatListDependency> {
    // ...
}
```

Component и Dependency

```
1 protocol ChatListDependency: Dependency {
2     var logger: ILogger { get }
3     var chatService: IChatService { get }
4 }
5
6 class ChatListComponent: Component<ChatListDependency> {
7
8 }
```

Component и Dependency

```
class ChatListAssembly {
    private let component: ChatListComponent

    init(component: ChatListComponent) {
        self.component = component
    }

    func assemble() -> UIViewController {
        let presenter = ChatListPresenter(
            logger: component.dependency.logger,
            service: component.dependency.chatService
        )
        let viewController = ChatListViewController(presenter: presenter)
        return presenter
    }
}
```

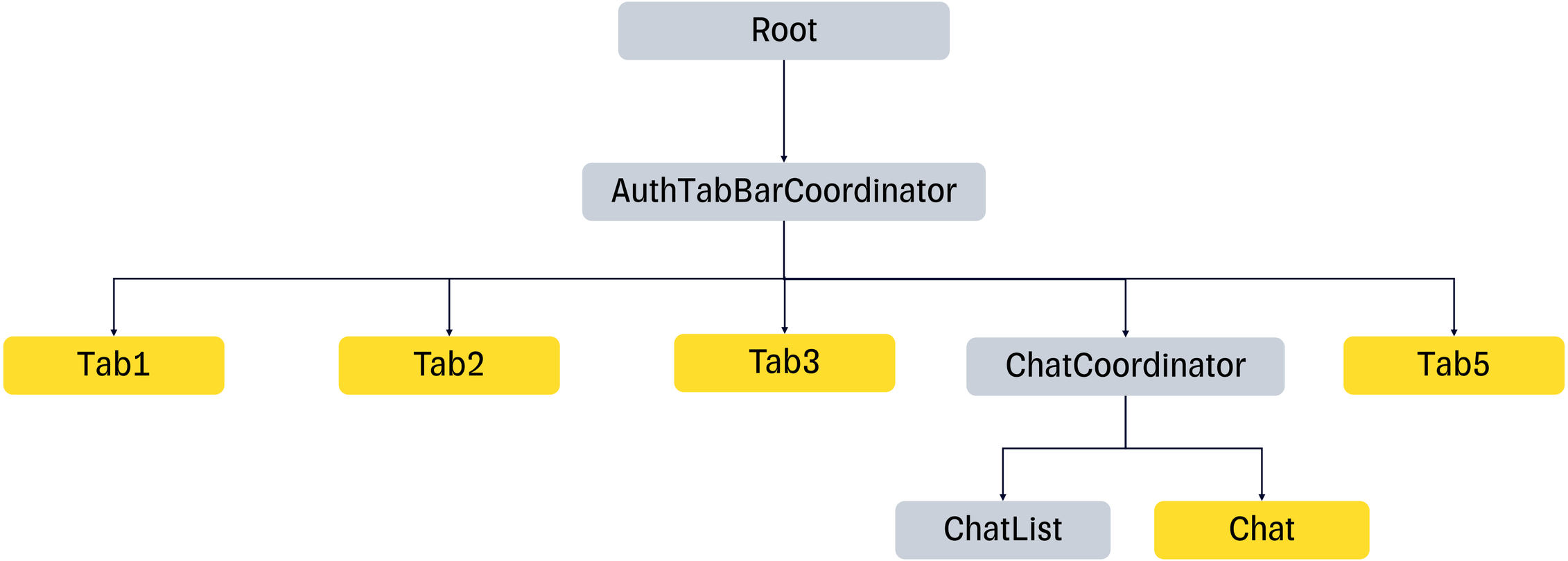
Component и Dependency

```
1 class ChatListAssembly {
2     private let component: ChatListComponent
3
4     init(component: ChatListComponent) {
5         self.component = component
6     }
7
8     func assemble() -> UIViewController {
9         let presenter = ChatListPresenter(
10            logger: component.dependency.logger,
11            service: component.dependency.chatService
12        )
13        let viewController = ChatListViewController(presenter: presenter)
14        return presenter
15    }
16 }
```

Component и Dependency

```swift
class ChatListAssembly {
    private let component: ChatListComponent

    init(component: ChatListComponent) {
        self.component = component
    }

    func assemble() -> UIViewController {
        let presenter = ChatListPresenter(
            logger: component.dependency.logger,
            service: component.dependency.chatService
        )
        let viewController = ChatListViewController(presenter: presenter)
        return presenter
    }
}
```

Scopes


Root
AuthTabBarCoordinator
Tab1
Tab2
Tab3
ChatCoordinator
Tab5
ChatList
Chat

Component и Dependency

```
protocol ChatListDependency: Dependency {
    var logger: ILogger { get }
    var chatService: IChatService { get }
}

class ChatListComponent: Component<ChatListDependency> {
    var chatListHelper: IChatListHelper {
        ChatListHelper()
    }
}
```

Component и Dependency

```swift
protocol ChatListDependency: Dependency {
    var logger: ILogger { get }
    var chatService: IChatService { get }
}

protocol ChatListComponentProtocol {
    var logger: ILogger { get }
    var chatService: IChatService { get }
    var chatListHelper: IChatListHelper { get }
}

class ChatListComponent: Component<ChatListDependency> {}

extension ChatListComponent: ChatListComponentProtocol {
    var logger: ILogger { dependency.logger }
    var chatService: IChatService { dependency.chatService }

    var chatListHelper: IChatListHelper {
        ChatListHelper()
    }
}
```

Component и Dependency

```swift
protocol ChatListDependency: Dependency {
    var logger: ILogger { get }
    var chatService: IChatService { get }
}

protocol ChatListComponentProtocol {
    var logger: ILogger { get }
    var chatService: IChatService { get }
    var chatListHelper: IChatListHelper { get }
}

class ChatListComponent: Component<ChatListDependency> {}

extension ChatListComponent: ChatListComponentProtocol {
    var logger: ILogger { dependency.logger }
    var chatService: IChatService { dependency.chatService }

    var chatListHelper: IChatListHelper {
        ChatListHelper()
    }
}
```

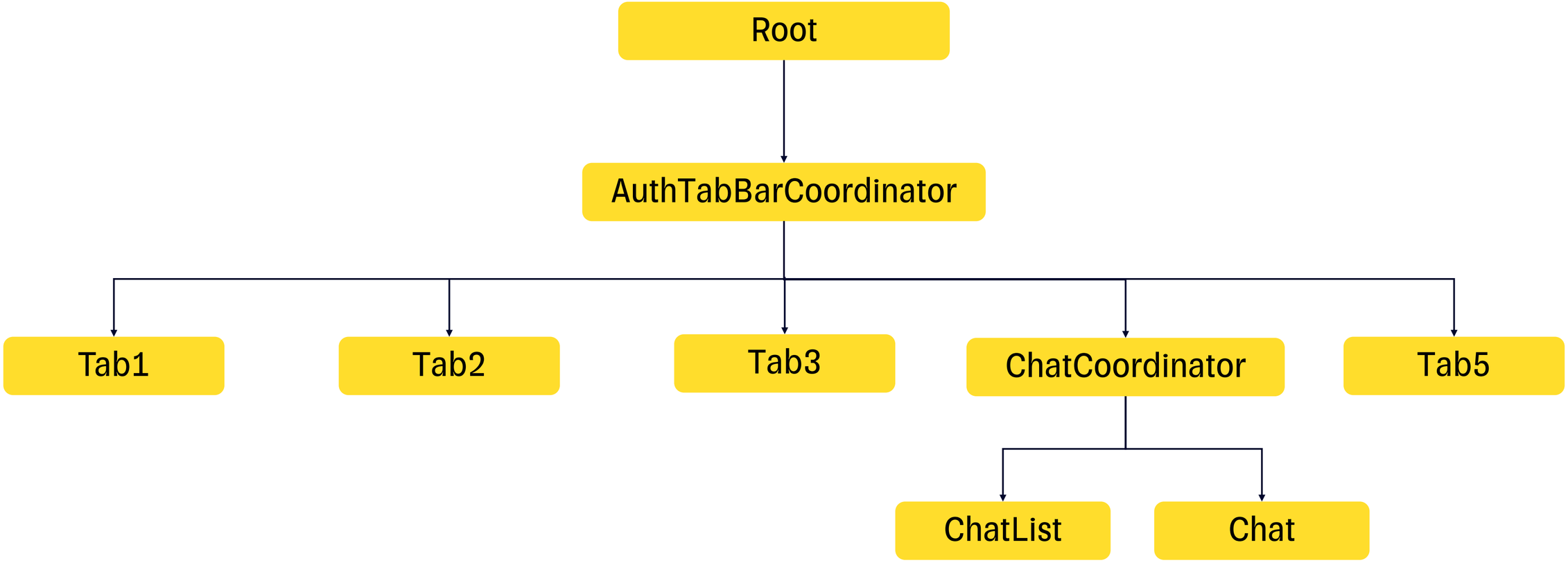
Component и Dependency

```swift
protocol ChatListDependency: Dependency {
    var logger: ILogger { get }
    var chatService: IChatService { get }
}

protocol ChatListComponentProtocol {
    var logger: ILogger { get }
    var chatService: IChatService { get }
    var chatListHelper: IChatListHelper { get }
}

class ChatListComponent: Component<ChatListDependency> {}

extension ChatListComponent: ChatListComponentProtocol {
    var logger: ILogger { dependency.logger }
    var chatService: IChatService { dependency.chatService }

    var chatListHelper: IChatListHelper {
        ChatListHelper()
    }
}
```

Иерархия Component

```swift
protocol ChatCoordinatorDependency: Dependency {
    // ...
}

class ChatCoordinatorComponent: Component<ChatCoordinatorDependency> {
    var chatListComponent: ChatListComponent {
        ChatListComponent(parent: self)
    }

    var chatComponent: ChatComponent {
        ChatComponent(parent: self)
    }
}
```

Scopes


Root
AuthTabBarCoordinator
Tab1
Tab2
Tab3
ChatCoordinator
Tab5
ChatList
Chat

BootstrapComponent

```
class ApplicationComponent: BootstrapComponent {
    var authTabBarCoordinatorComponent: AuthTabBarCoordinatorComponent {
        AuthTabBarCoordinatorComponent(parent: self)
    }

    var logger: ILogger {
        shared { Logger() }
    }
}
```

IChatService

```
protocol ChatCoordinatorDependency: Dependency {
    // ...
}

class ChatCoordinatorComponent: Component<ChatCoordinatorDependency> {
    var chatListComponent: ChatListComponent {
        ChatListComponent(parent: self)
    }

    var chatComponent: ChatComponent {
        ChatComponent(parent: self)
    }

    lazy var chatService: IChatService = {
        ChatService()
    }()
}
```

Ошибки в DI графе

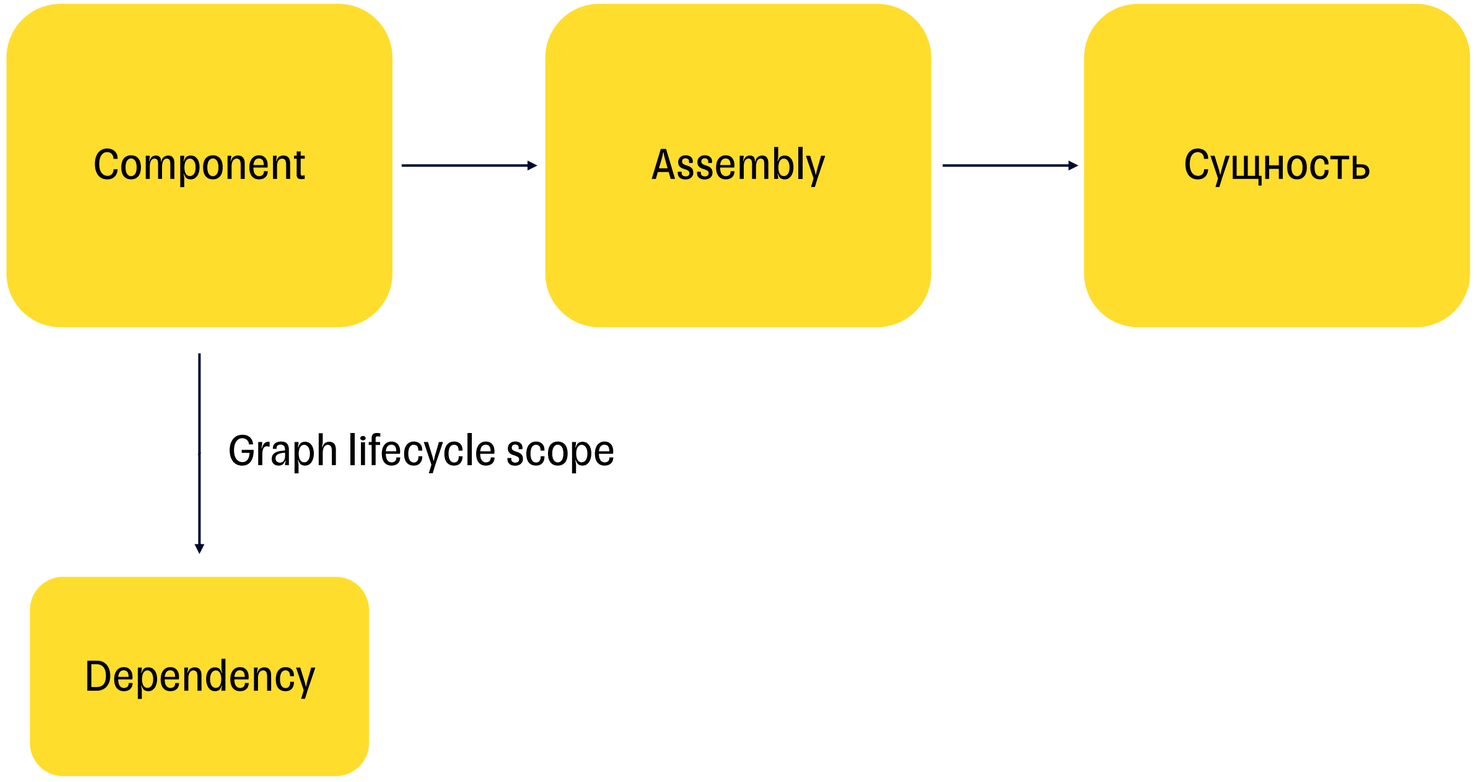
```
1 class ApplicationComponent: BootstrapComponent {
2     var authTabBarCoordinatorComponent: AuthTabBarCoordinatorComponent {
3         AuthTabBarCoordinatorComponent(parent: self)
4     }
5 }
6
7 class AuthTabBarCoordinatorComponent: Component<AuthTabBarCoordinatorDependency> {
8     var chatCoordinatorComponent: ChatCoordinatorComponent {
9         ChatCoordinatorComponent(parent: self)
10    }
11 }
12
13 class ChatCoordinatorComponent: Component<ChatCoordinatorDependency> {
14    var chatListComponent: ChatListComponent {
15        ChatListComponent(parent: self)
16    }
17 }
```

Ошибки в DI графе

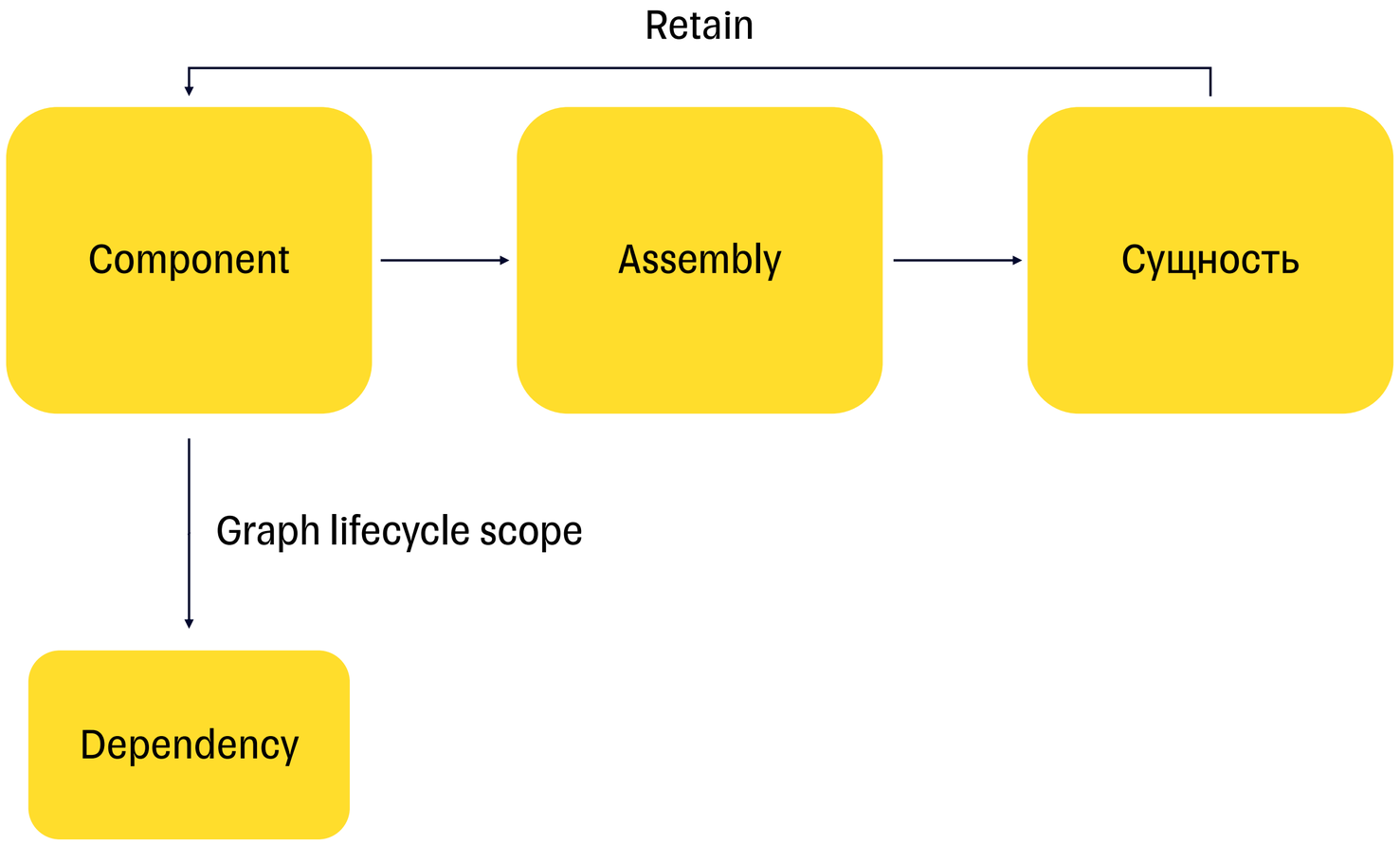
```
1 SOURCEKIT_LOGGING=0 && needle generate NeedleGenerated.swift $SRCROOT --exclude-paths Pods
```

```
warning: ❗ Could not find a provider for (logger: ILogger) which was required by ChatListDependency, along the DI branch of
  ^->ApplicationComponent->AuthTabBarCoordinatorComponent->ChatCoordinatorComponent->ChatListComponent.
warning: ❗ Could not find a provider for (chatService: IChatService) which was required by ChatListDependency, along the DI
  branch of ^->ApplicationComponent->AuthTabBarCoordinatorComponent->ChatCoordinatorComponent->ChatListComponent.
warning: ❗ Missing one or more dependencies at scope.
error: 💩 Some dependencies are missing, please look at the warnings above for the list.
```

Время жизни Component



Время жизни Component



Время жизни Component

```
class BaseCoordinator {
    private let scope: Any

    init(scope: Any) {
        self.scope = scope
    }
}

class BasePresenter {
    private let scope: Any

    init(scope: Any) {
        self.scope = scope
    }
}

class Presenter: BasePresenter {
    private let logger: ILogger

    init(scope: Any, logger: ILogger) {
        self.logger = logger
        super.init(scope: scope)
    }
}
```

Время жизни Component

```
1 class BaseCoordinator {
2     private let scope: Any
3
4     init(scope: Any) {
5         self.scope = scope
6     }
7 }
8
9 class BasePresenter {
10    private let scope: Any
11
12    init(scope: Any) {
13        self.scope = scope
14    }
15 }
16
17 class Presenter: BasePresenter {
18    private let logger: ILogger
19
20    init(scope: Any, logger: ILogger) {
21        self.logger = logger
22        super.init(scope: scope)
23    }
24 }
```

Время жизни Component

```swift
class ChatListAssembly {
    private let componentProvider: () -> ChatListComponent

    init(componentProvider: @escaping () -> ChatListComponent) {
        self.componentProvider = componentProvider
    }

    func assemble() -> UIViewController {
        let component = componentProvider()
        // ..
    }
}
```

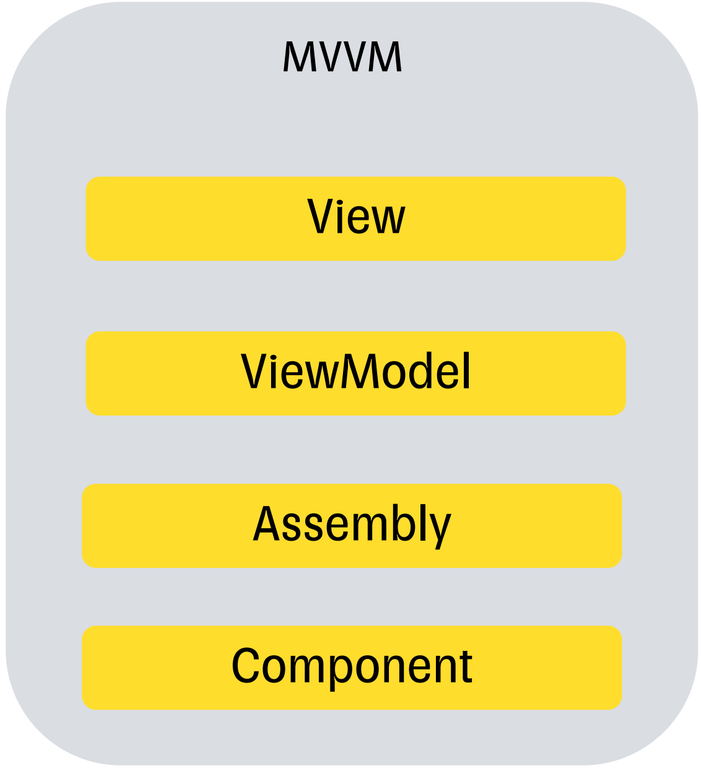
Время жизни Component

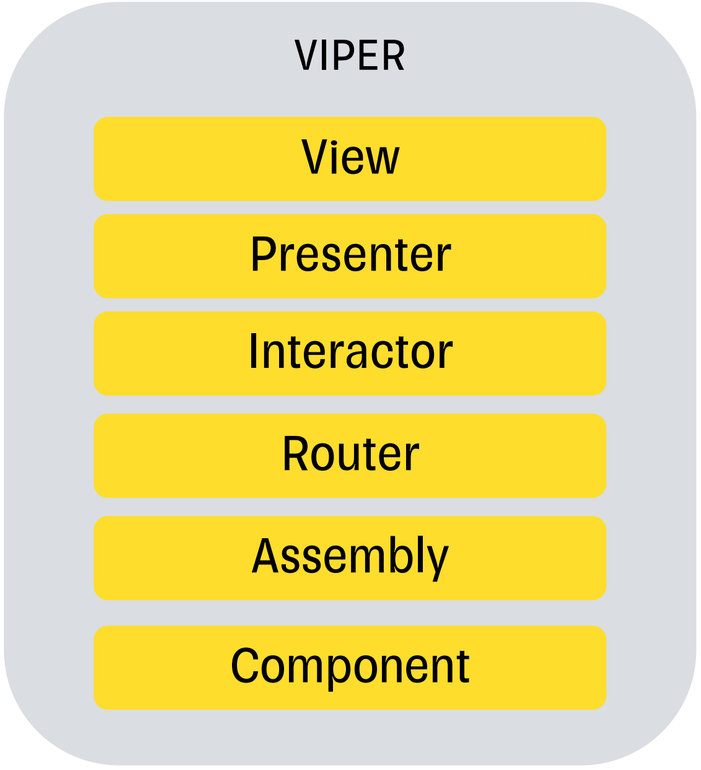
```swift
class ChatListAssembly {
    private let componentProvider: () -> ChatListComponent

    init(componentProvider: @escaping () -> ChatListComponent) {
        self.componentProvider = componentProvider
    }

    func assemble() -> UIViewController {
        let component = componentProvider()
        // ..
    }
}
```

Время жизни Component

```
1 class ChatCoordinatorAssembly {
2     private let componentProvider: () -> ChatCoordinatorComponent
3
4     init(componentProvider: @escaping () -> ChatCoordinatorComponent) {
5         self.componentProvider = componentProvider
6     }
7
8     func assemble() -> IChatCoordinator {
9         let component = componentProvider()
10        return ChatCoordinator(
11            scope: component,
12            chatListAssembly: ChatListAssembly { component.chatListComponent },
13            chatAssembly: ChatAssembly { component.chatComponent }
14        )
15    }
16 }
```

DI Component – часть архитектуры





Как работает генератор

Схема работы сгенерированного кода

Scopes

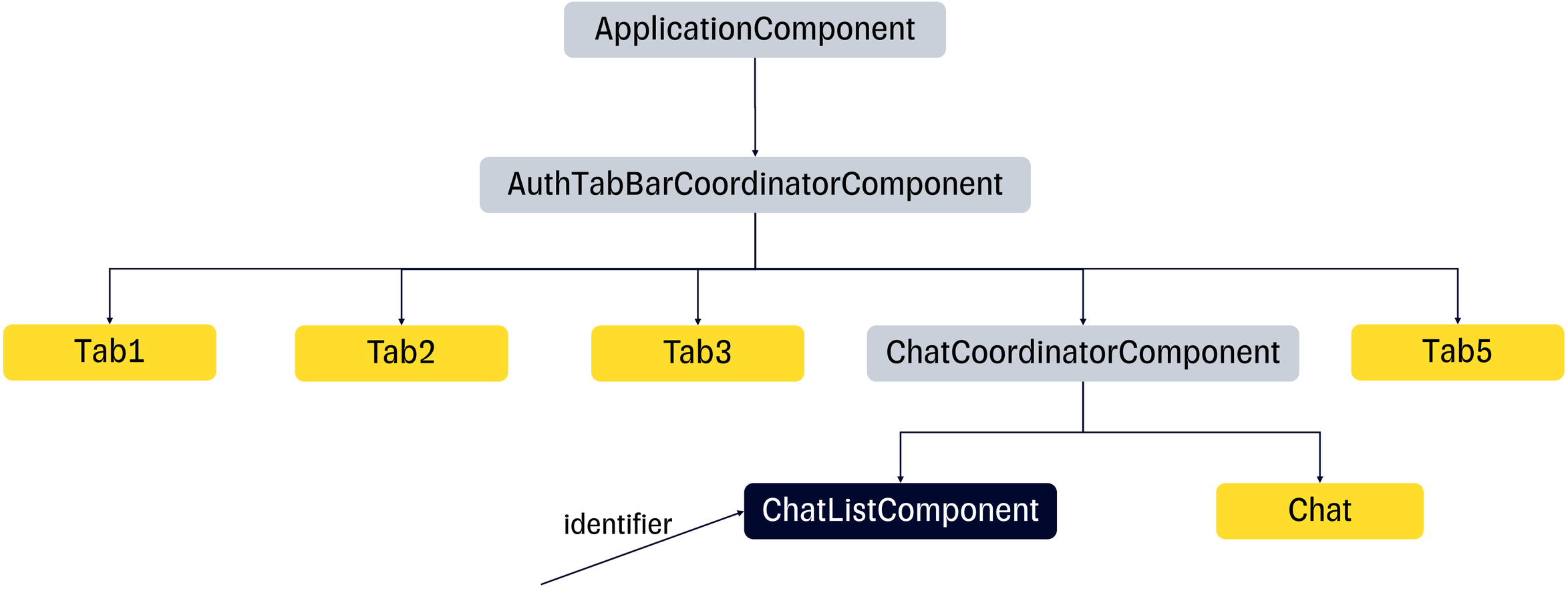
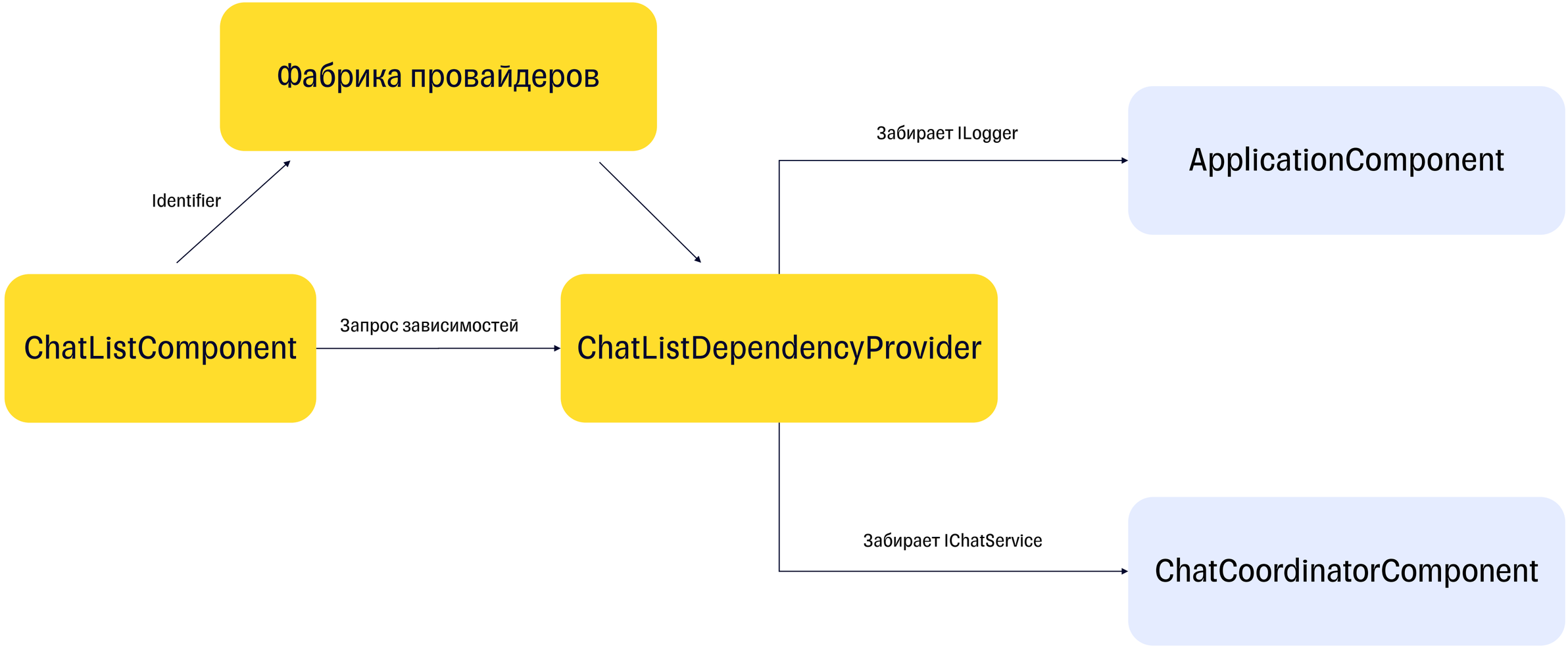

ApplicationComponent
AuthTabBarCoordinatorComponent
Tab1
Tab2
Tab3
ChatCoordinatorComponent
Tab5
ChatListComponent
Chat
identifier
“ ^->ApplicationComponent->AuthTabBarCoordinatorComponent->ChatCoordinatorComponent->ChatListComponent “

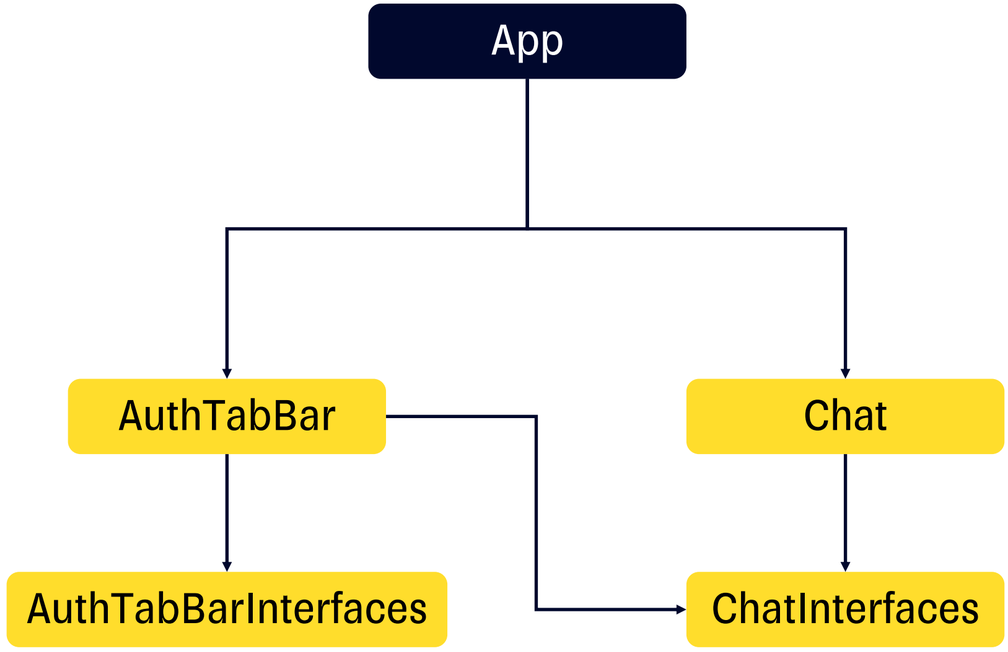
Схема работы сгенерированного кода



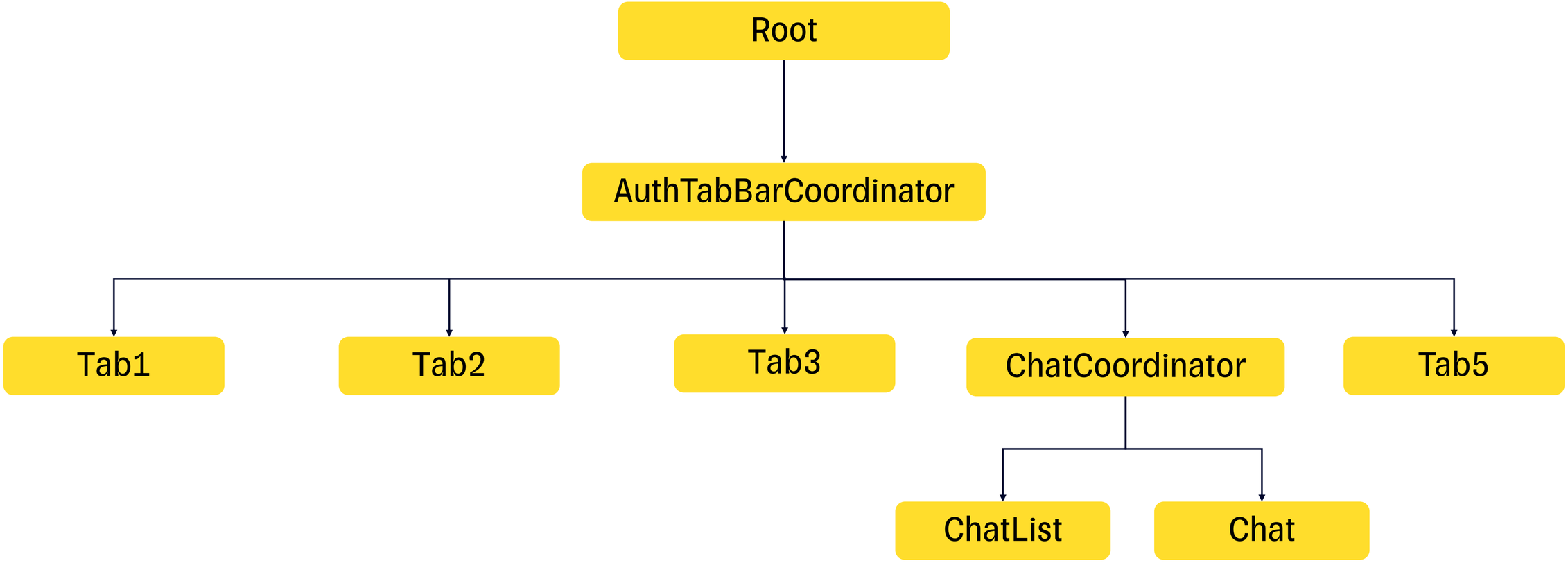
План доклада

- Модуляризация
- Выбор DI фреймворка
- Needle
- **Многомодульный DI**
- Тестирование

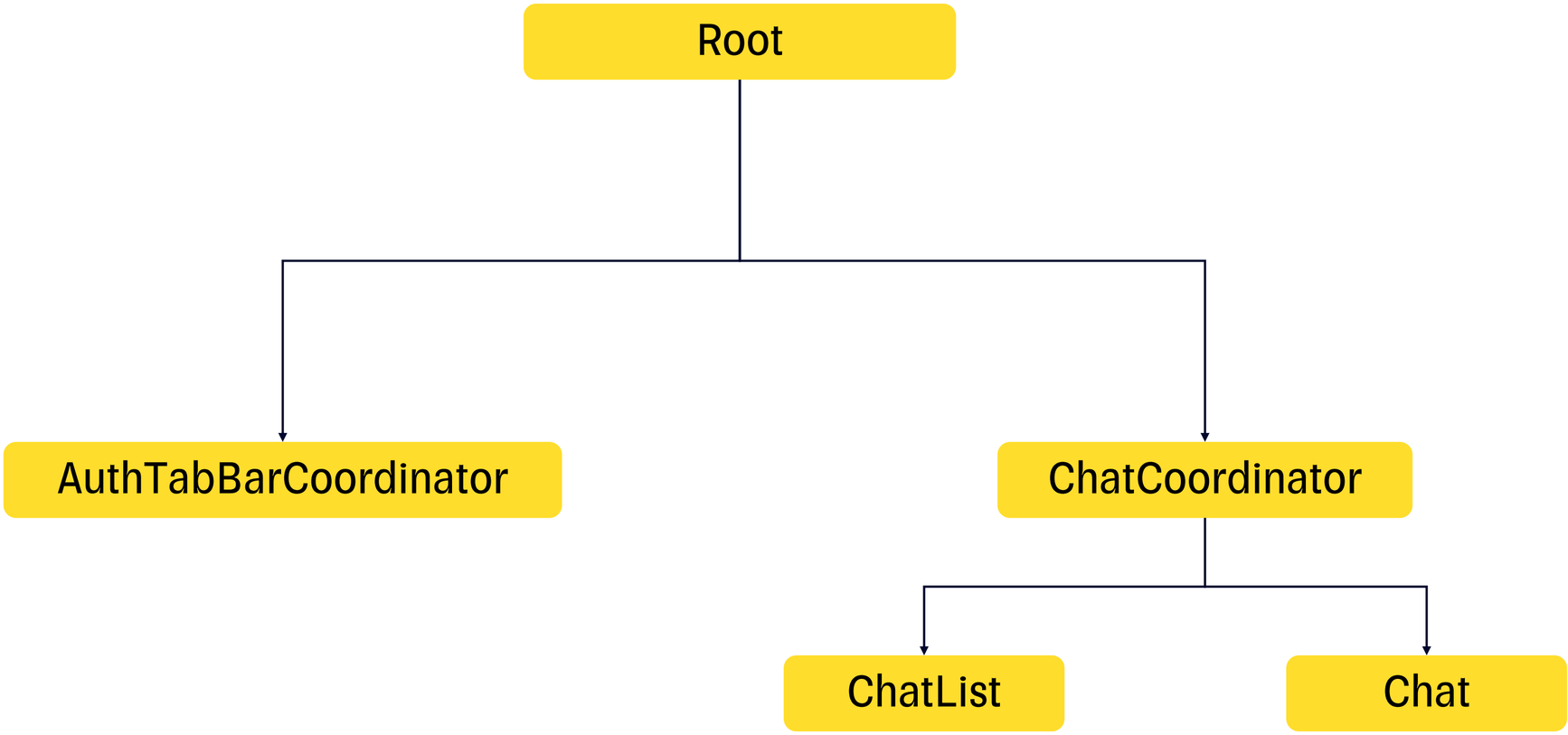
Межмодульные связи



Scopes


Root
AuthTabBarCoordinator
Tab1
Tab2
Tab3
ChatCoordinator
Tab5
ChatList
Chat

Scopes


Root
AuthTabBarCoordinator
ChatCoordinator
ChatList
Chat

Многомодульное приложение

```
import ChatInterfaces

class AuthTabBarCoordinatorComponent: Component<AuthTabBarCoordinatorDependency> {
//    var chatCoordinatorComponent: ChatCoordinatorComponent {
//        ChatCoordinatorComponent(parent: self)
//    }

    var chatCoordinatorAssembly: IChatCoordinatorAssembly {
        dependency.chatCoordinatorAssembly
    }
}
```

Многомодульное приложение

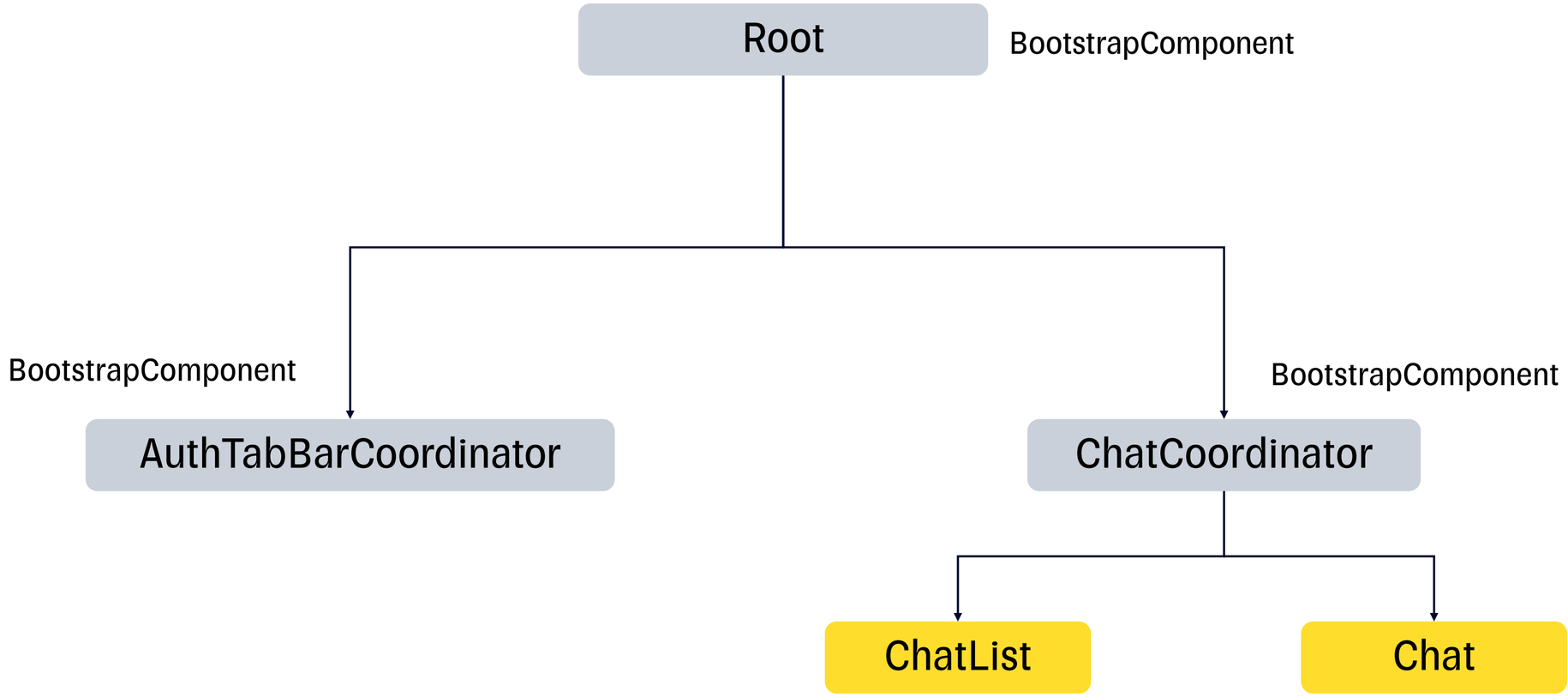
```
import Chat
import ChatInterfaces

class ApplicationComponent: BootstrapComponent {
    var chatCoordinatorAssembly: IChatCoordinatorAssembly {
        ChatCoordinatorAssembly {
            ChatCoordinatorComponent(parent: self)
        }
    }
}
```

Кажется, все хорошо, но есть одно но...

Многомодульный Needle

```swift
public class ChatCoordinatorComponent: Component<EmptyDependency>, IChatCoordinatorComponent {
    public var chatListComponent: ChatListComponent {
        ChatListComponent(parent: self)
    }

    public var chatComponent: ChatComponent {
        ChatComponent(parent: self)
    }

    public lazy var chatService: IChatService = {
        ChatService()
    }()
}
```

Многомодульный Needle



Многомодульный Needle

```
1 public class ChatCoordinatorComponent: BootstrapComponent, IChatCoordinatorComponent {
2     var chatListComponent: ChatListComponent {
3         ChatListComponent(parent: self)
4     }
5
6     var chatComponent: ChatComponent {
7         ChatComponent(parent: self)
8     }
9
10    lazy var chatService: IChatService = {
11        ChatService()
12    }()
13 }
```

Многомодульный Needle

```swift
public protocol ChatCoordinatorDependency {
    var logger: ILogger { get }
}

public class ChatCoordinatorComponent: BootstrapComponent, IChatCoordinatorComponent {
    private let dependencies: ChatCoordinatorDependency

    public init(dependencies: ChatCoordinatorDependency) {
        self.dependencies = dependencies
    }

    var logger: ILogger {
        dependencies.logger
    }

    // ...
}
```

Многомодульный Needle

```swift
public protocol ChatCoordinatorDependency {
    var logger: ILogger { get }
}

public class ChatCoordinatorComponent: BootstrapComponent, IChatCoordinatorComponent {
    private let dependencies: ChatCoordinatorDependency

    public init(dependencies: ChatCoordinatorDependency) {
        self.dependencies = dependencies
    }

    var logger: ILogger {
        dependencies.logger
    }

    // ...
}
```

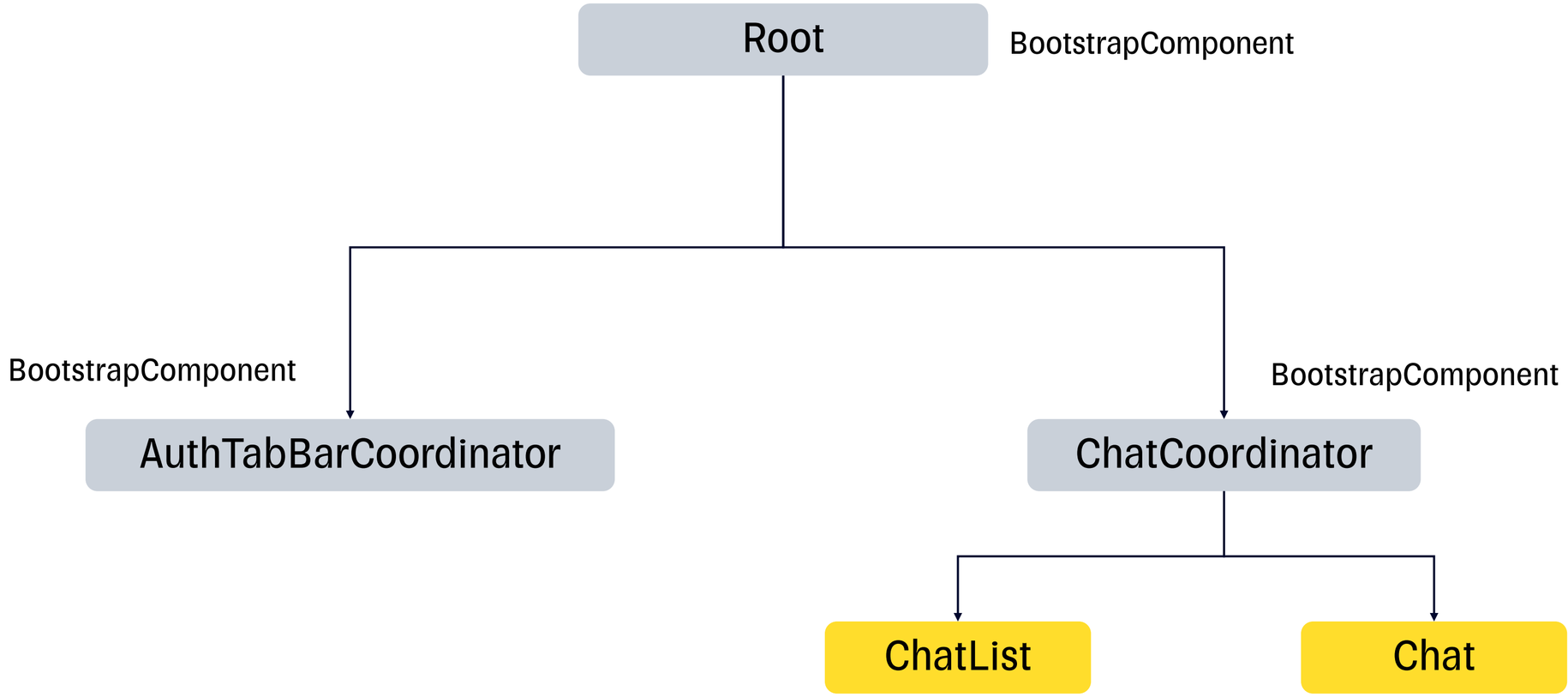
Многомодульный Needle

```swift
public protocol ChatCoordinatorDependency {
    var logger: ILogger { get }
}

public class ChatCoordinatorComponent: BootstrapComponent, IChatCoordinatorComponent {
    private let dependencies: ChatCoordinatorDependency

    public init(dependencies: ChatCoordinatorDependency) {
        self.dependencies = dependencies
    }

    var logger: ILogger {
        dependencies.logger
    }

    // ...
}
```

Многомодульный Needle

```
1  class ApplicationComponent: BootstrapComponent, ChatCoordinatorDependency {
2      var chatCoordinatorAssembly: IChatCoordinatorAssembly {
3          ChatCoordinatorAssembly {
4              ChatCoordinatorComponent(dependencies: self)
5          }
6      }
7
8      var logger: ILogger {
9          shared { Logger() }
10     }
11 }
```

Многомодульный Needle



Многомодульный Needle

```
SOURCEKIT_LOGGING=0 && needle generate $SRCROOT/ChatFeature/NeedleGenerated.swift $SRCROOT/ChatFeature --exclude-paths Pods
```

```
    func application(
      _ application: UIApplication,
      didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        registerProviderFactories()
        Chat.registerProviderFactories()

        return true
    }
```

План доклада

- Модуляризация
- Выбор DI фреймворка
- Needle
- Многомодульный DI
- **Тестирование**

MockingPath

```
1 func testSome() {
2     let testDependencyProvider = MockDependencyProvider()
3     let componentPath = mockComponentPathBuilder()
4         .extendPath(to: BootstrapComponent.self)
5         .extendPath(to: TestComponent.self)
6         .build()
7     componentPath.register(dependencyProvider: testDependencyProvider)
8     defer {
9         componentPath.unregister()
10    }
11
12    let bootstrapComponent = BootstrapComponent()
13    let testComponent = TestComponent(parent: bootstrapComponent)
14 }
```

Тестирование

```
class ChatListAssembly: IChatListAssembly {
    private let componentProvider: () -> ChatListComponent

    init(componentProvider: @escaping () -> ChatListComponent) {
        self.componentProvider = componentProvider
    }

    func assemble() -> UIViewController {
        let component = componentProvider()
        let logger = component.logger
        let chatService = component.chatService
        // ..
    }
}
```

Тестирование

```swift
protocol IChatListComponent {
    var logger: ILogger { get }
    var chatService: IChatService { get }
}

class ChatListAssembly: IChatListAssembly {
    private let componentProvider: () -> IChatListComponent

    init(componentProvider: @escaping () -> ChatListComponent) {
        self.componentProvider = componentProvider
    }

    init(componentProvider: @escaping () -> IChatListComponent) {
        self.componentProvider = componentProvider
    }

    func assemble() -> UIViewController {
        let component = componentProvider()
        let logger = component.logger
        let chatService = component.chatService
        // ..
    }
}
```

```swift
class Tests: XCTestCase {
    func testSome() {
        let chatListComponentMock = ChatListComponentMock()
        let chatCoordinatorComponentMock = ChatCoordinatorComponentMock()
        chatCoordinatorComponentMock.chatListComponent = chatListComponentMock
        // ..

        let coordinatorAssembly = ChatCoordinatorAssembly {
            chatCoordinatorComponentMock
        }
        let coordinator = coordinatorAssembly.assemble()
        // ..
    }
}

class ChatCoordinatorComponentMock: IChatCoordinatorComponent {
    var chatListComponentMock: IChatListComponent!
    var chatListComponent: IChatListComponent {
        chatListComponentMock
    }

    var chatComponentMock: IChatComponent!
    var chatComponent: IChatComponent {
        chatComponentMock
    }

    var chatServiceMock: IChatService!
    var chatService: IChatService {
        chatServiceMock
    }
}

class ChatListComponentMock : IChatListComponent {
    var loggerMock: ILogger!
    var logger: ILogger { loggerMock }

    var chatServiceMock: IChatService!
    var chatService: IChatService { chatServiceMock }
}

```

```swift
class Tests: XCTestCase {
    func testSome() {
        let chatListComponentMock = ChatListComponentMock()
        let chatCoordinatorComponentMock = ChatCoordinatorComponentMock()
        chatCoordinatorComponentMock.chatListComponent = chatListComponentMock
        // ..

        let coordinatorAssembly = ChatCoordinatorAssembly {
            chatCoordinatorComponentMock
        }
        let coordinator = coordinatorAssembly.assemble()
        // ..
    }
}

class ChatCoordinatorComponentMock: IChatCoordinatorComponent {
    var chatListComponentMock: IChatListComponent!
    var chatListComponent: IChatListComponent {
        chatListComponentMock
    }

    var chatComponentMock: IChatComponent!
    var chatComponent: IChatComponent {
        chatComponentMock
    }

    var chatServiceMock: IChatService!
    var chatService: IChatService {
        chatServiceMock
    }
}

class ChatListComponentMock : IChatListComponent {
    var loggerMock: ILogger!
    var logger: ILogger { loggerMock }

    var chatServiceMock: IChatService!
    var chatService: IChatService { chatServiceMock }
}

```

```
class Tests: XCTestCase {
    func testSome() {
        let chatListComponentMock = ChatListComponentMock()
        let chatCoordinatorComponentMock = ChatCoordinatorComponentMock()
        chatCoordinatorComponentMock.chatListComponent = chatListComponentMock
        // ..

        let coordinatorAssembly = ChatCoordinatorAssembly {
            chatCoordinatorComponentMock
        }
        let coordinator = coordinatorAssembly.assemble()
        // ..
    }
}

class ChatCoordinatorComponentMock: IChatCoordinatorComponent {
    var chatListComponentMock: IChatListComponent!
    var chatListComponent: IChatListComponent {
        chatListComponentMock
    }

    var chatComponentMock: IChatComponent!
    var chatComponent: IChatComponent {
        chatComponentMock
    }

    var chatServiceMock: IChatService!
    var chatService: IChatService {
        chatServiceMock
    }
}

class ChatListComponentMock : IChatListComponent {
    var loggerMock: ILogger!
    var logger: ILogger { loggerMock }

    var chatServiceMock: IChatService!
    var chatService: IChatService { chatServiceMock }
}

```

Итоги

- Модуляризация
- Выбор DI фреймворка
- Needle
- Многомодульное приложение
- Тестирование

Ссылки

- https://github.com/uber/needle (Репозиторий Needle)
- https://dagger.dev (Репозиторий Dagger2)
- https://habr.com/ru/company/tinkoff/blog/546360/ (Введение и обзор DI фреймворков)
- https://github.com/leoniknik/NeedleModularExample (Пример многомодульного приложения с Needle)

Спасибо за внимание

IT's TINKOFF