Setting Boundaries

{VAN} HACKS

Francisco Díaz

@fco_diaz

AZ

iOS Love

3

iOS Devs

28

hours

1

project

== Merge Conflicts

What do we want?

→ Minimize duplication of code.

→ Develop independently without *stepping on each other's toes*.

Feature verticals:

→ Big Panic button

→ Today widget

→ Knock

Big Panic button:

What needs to be done?

→ Create the button.

→ We need a way to create reports.

→ Make a backend call to save this information.

Today widget:

What needs to be done?

→ Create the extension button.

→ We need a way to create reports.

→ Make a backend call to save this information.

What was it that we wanted?

→ ~~Minimize duplication of code.~~

→ ~~Develop independently without *stepping on each other's toes*.~~

Let's try again!

→ Create the button.

→ We need a way to create reports.

→ Make a backend call to save this information.

UI / Presentation

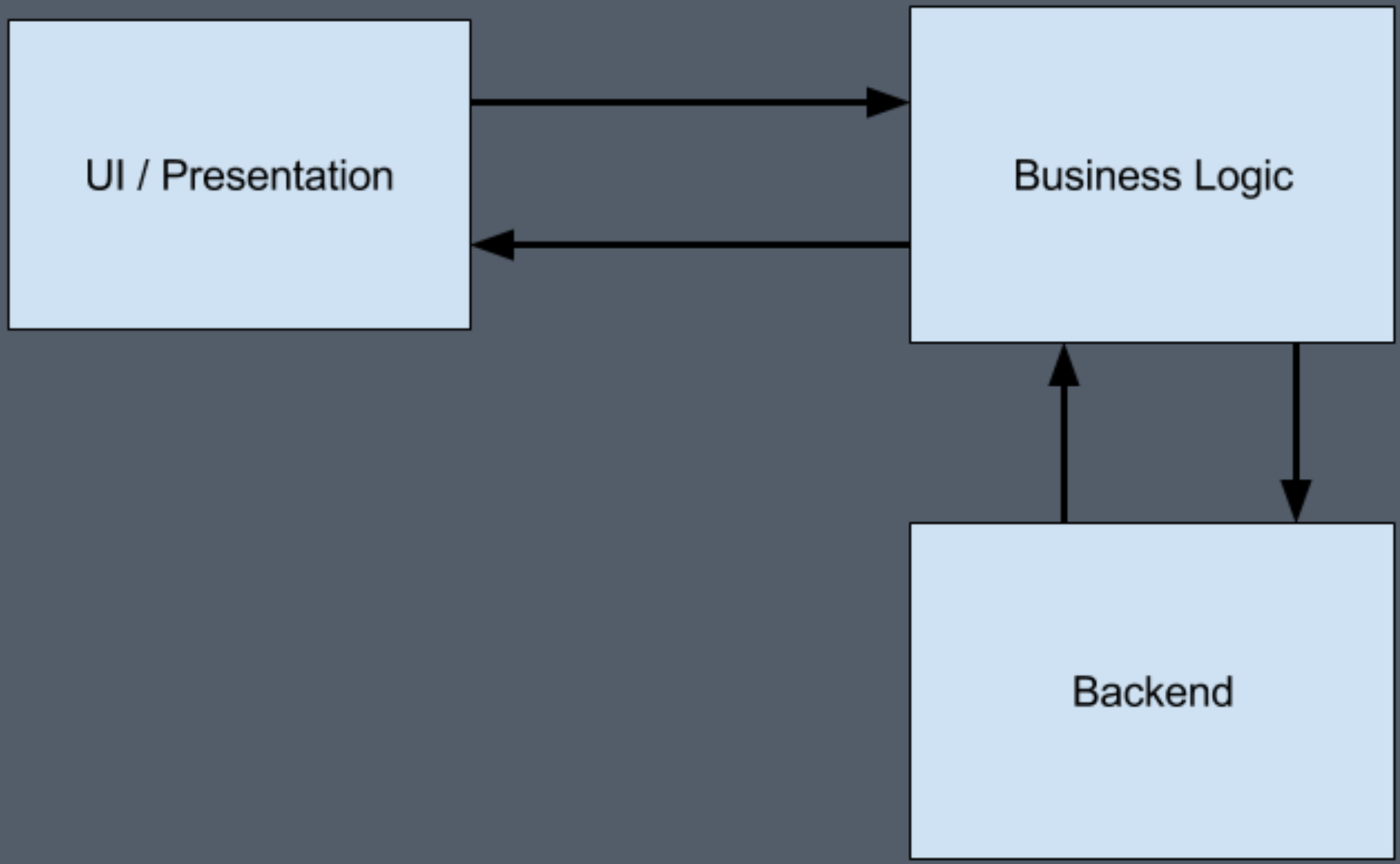
Create the button.

Business logic

We need a way to create reports.

Backend connection

Make a backend call to save this information.


UI / Presentation
Business Logic
Backend

To recap:

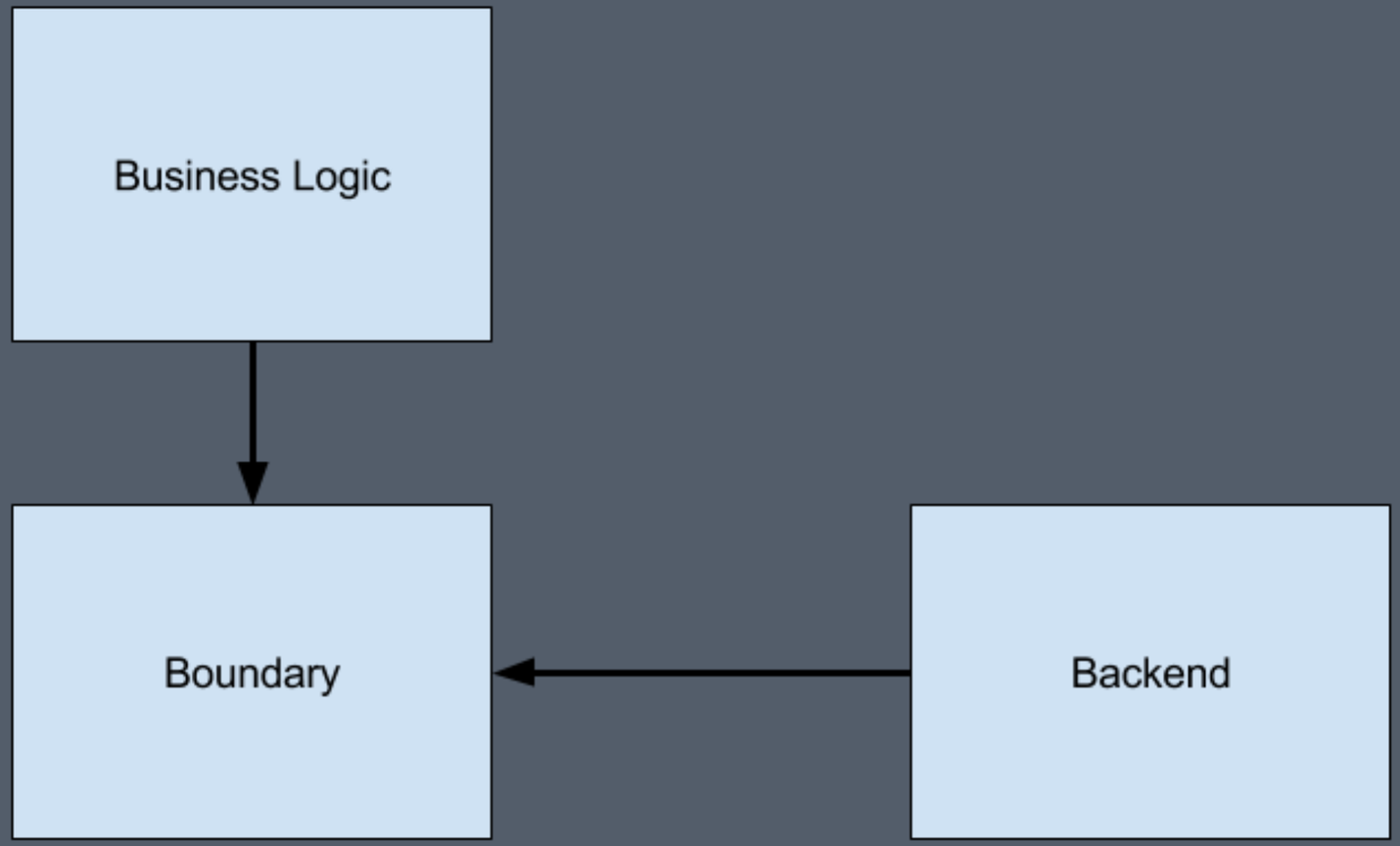
→ Minimize duplication of code.

→ ~~Develop independently without *stepping on each other's toes*.~~

We can solve any problem by introducing an extra level of indirection

— David Wheeler

Dependency inversion



```
struct ModelDataManager {
    let APIClient: APIType

    init(APIClient: APIType) {
        self.APIClient = APIClient
    }
}
```

```
protocol APIType {
    func createReport(completion: JSONDictionary? -> Void)
}

struct API {
    private let manager: Alamofire.Manager

    init() {
        manager = Alamofire.Manager()
    }
}

extension API: APIType {
    func createReport(completion: JSONDictionary? -> Void) {
        manager.request(.POST, "https://some.com/api/report")
            .responseJSON { response in
                completion(response)
        }
    }
}
```

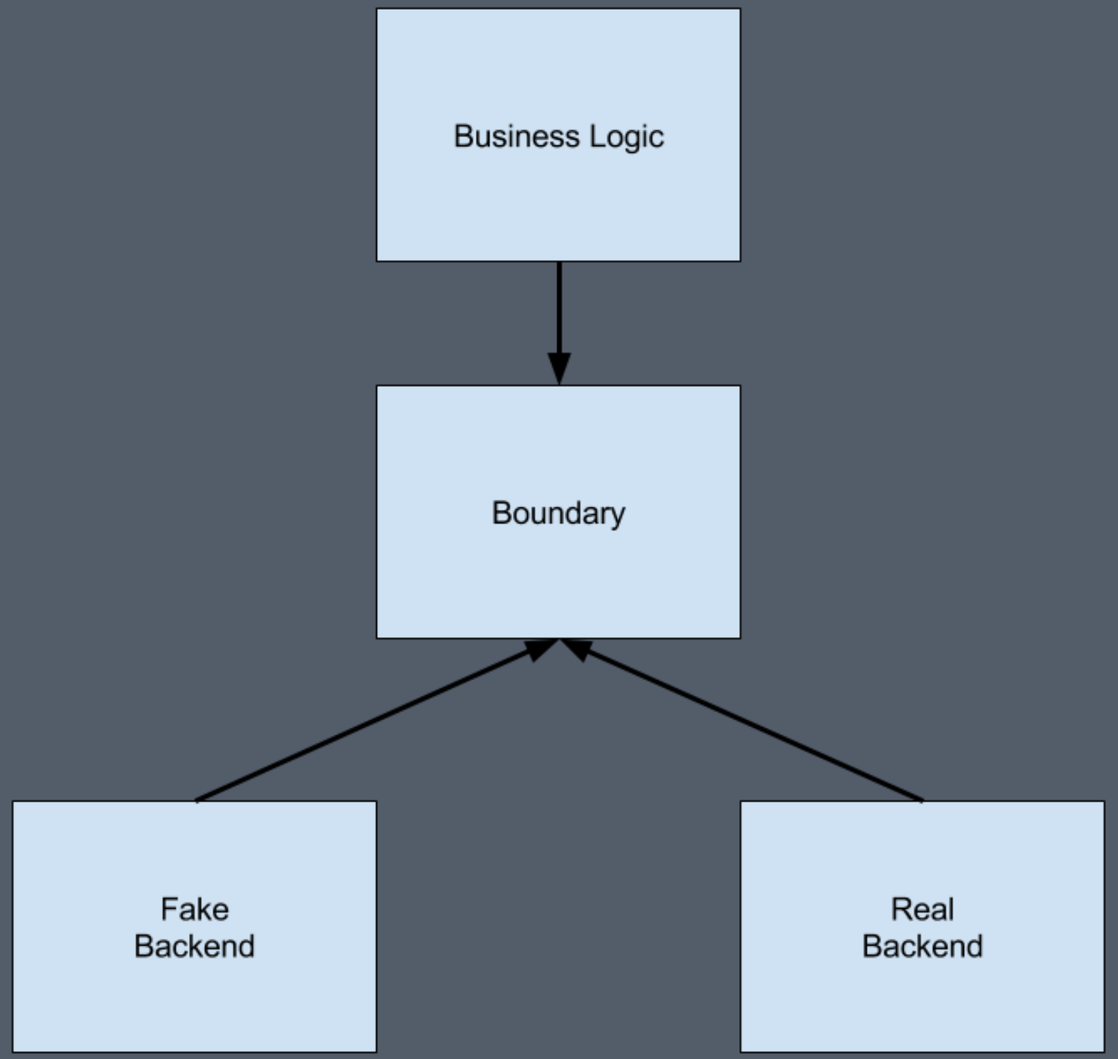
```
struct ModelDataManager {
    let APIClient: APIType

    init(API: APIType) {
        self.APIClient = API
    }

    static func defaultManager() -> ModelDataManager {
        let APIClient = API()
        return ModelDataManager(API: APIClient)
    }

    func createReport(completionHandler completion: Report? -> Void) {
        APIClient.createReport() { jsonDictionary in
            let report = ... // Parse jsonDictionary into Report
            completion(report)
        }
    }
}
```

Benefits

→ Testable.

→ Decoupled.

→ Easy to fake our networking layer.


Business Logic
Boundary
Fake
Backend
Real
Backend

```
struct FakeAPI: APIType {
    func createReport(completion: JSONDictionary? -> Void) {
        let dictionary = ["id": 12345]
        completion(dictionary)
    }
}
```

```
struct ModelDataManager {
    let APIClient: APIType

    init(API: APIType) {
        self.APIClient = API
    }

    static func defaultManager() -> ModelDataManager {
        // let APIClient = API()
        let APIClient = FakeAPI()
        return ModelDataManager(API: APIClient)
    }
}
```

Questions?

Slides are available at:
https://github.com/fdiaz/settings-boundaries-talk

References:

Architecture: The Lost Years

The Clean Architecture