API - Model - Service - ViewController

Karol Kubicki
karol@tooploox.com
Tooploox

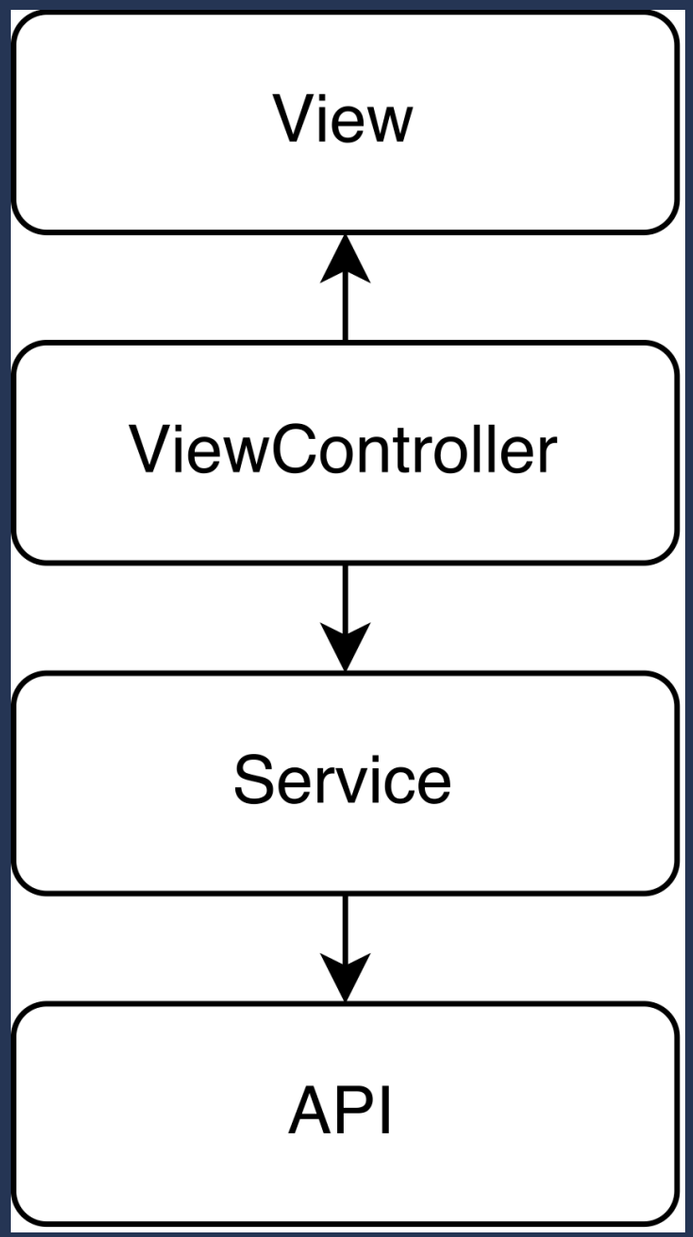
Aplikacja z wykładami

- Lista wykładów
- Przeglądanie szczegółów
- Edycja wykładu
- Dodawanie wykładu

Architektura

```
API <- Service <- ViewController -> View
```

Rozdzielenie na warstwy, aby można je niezależnie od siebie modyfikować



API

API

- Wysyłanie zapytań `HTTP` na podany URL (`NSURLSession`)
- Operacje HTTP
  - `GET` - pobieranie
  - `POST` - wysyłanie (czasem edycja)
  - `PUT` - edycja
  - `DELETE` - usuwanie

API

- Pobrane dane w formacie binarnym `NSData` konwertujemy na słownik `JSONa`
- Wysyłamy słownik `JSONa` w zapytaniu `POST` konwertując go na dane binarne `NSData`
- Jako serwera użyjemy Parse.com

Service

Service

- Zawiera w sobie obiekt `API`
- Z otrzymanego `JSON` z `API` tworzy obiekty modelu
- Miejsce do logiki biznesowej tzn. wysokopoziomowych operacji wymaganych do działania programu - Business logic
- Zwraca obiekty klasy modelu

ViewController

ViewController

- Zawiera obiekt `Service`
- Pobierane dane z serwisu - wyświetla je lub obsługuje błędy
- Wywołuje metody dodawania, edycji, pobierania na serwisie
- Reaguje na zdarzenia z widoku

View

View

- Najbardziej zewnętrzna część aplikacji
- Widok jest wypełniony danymi przez `ViewController`
- Wysyła zdarzenia interakcji użytkownika do swojego kontrolera

Zadanie

- Dopisanie funkcjonalności dodawanie wykładu
- Dodanie pól tekstowych i przycisku do widoku
- Podpięcie pól do kontrolera i akcji przycisku
- Wywołanie w kontrolerze metody dodawania wykładu na obiekcie serwisu
- Wywołanie w serwisie metody dodawania wykładu na obiekcie API

Parse - www.parse.com

- W prezentacji użyłem RESTowego API - dokumentacja REST
- Jest też biblioteka do iOS - dokumentacja iOS
- Po kilku dniach usunę aplikację z Parsa - musicie założyć sobie własna (jest to darmowe)
- Po stworzeniu aplikacji musicie podmienić `parseApplicationId` oraz `parseRestApiKey` w pliku `ParseAPIHeadersProvider`