Angular

1. MVC - Model - service, View - template, Controller - Component.ts

2. Template View

“Tworzenie dynamicznej strony HTML wg podobnych zasad jak w przypadku strony statycznej np. za pomocą edytora WYSIWYG, ale z osadzeniem na tej stronie znaczników, które podczas przetwarzania strony zostaną zamienione na dynamiczne wywołania pobierające odpowiednie informacje (np. wyniki zapytań). Dzięki temu odseparowanie części projektowej (odpowiedzialny projektant strony) i logicznej (odpowiedzialny programista). W MVC widok z reguły tworzony w oparciu o Template View (zaleta: możliwość stworzenia strony na podstawie analizy jej struktury, separacja projektowania strony od logiki programowej,”
np. ngIf, ngFor

API

1. Page Controller (api.add_resource) endpoint
“Obiekt obsługujący żądania skierowane do **konkretnej strony** lub określonej operacji udostępnianej przez witrynę
•Odpowiedzialność wzorca Page Controller: – dekodowanie adresu URL i pobranie informacji z formularza w celu uzyskania wszystkich danych niezbędnych do wykonania operacji; – stworzenie i wywołanie obiektów modelu koniecznych do przetworzenia danych (obiekt modelu nie powinien odwoływać się bezpośrednio do żądania HTTP); , Page Controller może być implementowany jako jedna lub wiele klas (może wywoływać obiekty pomocnicze).”
Np User advice opdpowiadaja za poszczegolne podstrony

2. Front Controller (api glowny obiekt,framework)
“Konsolidacja obsługi żądań przez skierowanie ich do **jednego obiektu(api)**, który realizuje powszechne operacje (zabezpieczenia, obsługa wersji językowych itp.) eliminując ich powielanie w kodzie. Takie operacje można modyfikować w trakcie działania aplikacji za pomocą dekoratorów. Obiekt obsługujący żądanie **przekazuje do polecenia obiekty odpowiadające konkretnym żądaniom.**”
Głowny obiekt API

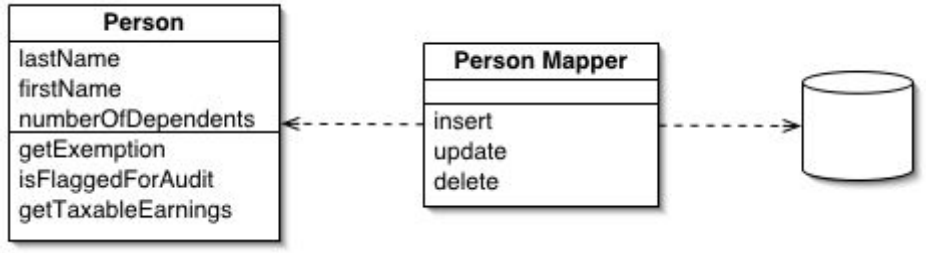
3.Application Controller (Dispatchers)

- ApplicationController - obiekt odpowiedzialny za przekierowywanie żądań do odpowiednich obiektów implementujących akcje i widoki;
- centralizacja sterowania i poprawa zarządzania przepływem sterowania;
- ułatwienie pielęgnacji kodu, zwiększenie możliwości ponownego użycia kodu odpowiedzialnego za obsługę żądań;
-

”

3.Table Data Gateway(User,Advice.getAll)
Interface do operacji na tabeli
W api wystawiam obiekty np. User,Advice, Comment które zawierają metody
getAlll, getSingle
dla nich liczba mnoga np Users wystawiaja Data Gateway
“The answer is so common that it's hardly worth stating. Wrap all the special API code into a class whose interface looks like a regular object. Other objects access the resource through this Gateway, which translates the simple method calls into the appropriate specialized API.”

4.Mapper
W api wystawiam obiekty np. User,Advice, Comment które zawierają metody
getAll, getSingle



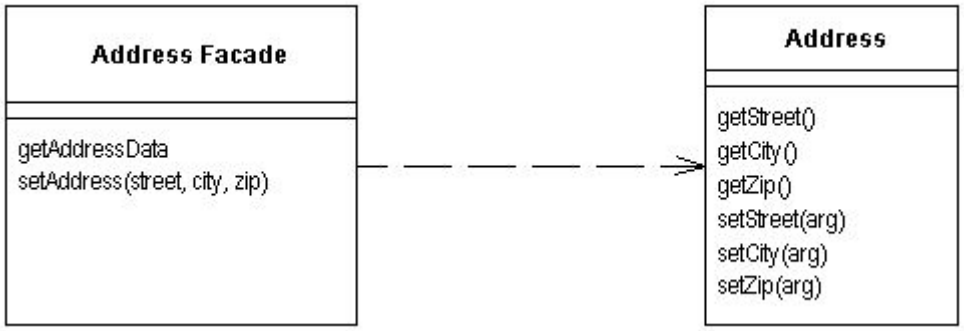
https://martinfowler.com/eaaCatalog/dataMapper.html
u nas User gdzie User bezposrednio na bazie a Mapper to createUser, deleteUser, changePassword

5.Layer supertype
Podzielone modele w api na Advices,Categories,Comments,Users

“• Gdy wszystkie obiekty z danej warstwy powinny posiadać wspólne cechy lub zachowania.
• Layer Supertype – klasa bazowa dla wszystkich klas warstwy, implementująca wspólne cechy i zachowania. ”

6. Remote fasade
Zamiast wybierac poszczegolne dane z uzytkownika stosujemy `getSingleUser`

i otrzymujemy dane zbiorcze.



u nas np. getAdviceComments(Id) ----------> getAdvice(Id),getComment(Id)

7. Repository

Repozytorium to kolejny sposób na organizację naszego dostępu do bazy danych. To w pełni obiektowy wzorzec, który pozwala nam spełnić wszystkie dobre zasady programowania.

np klasa Comments:
getAllComments
getSingleComment