



Logols Learning

WEEKEND WEB DEVELOPMENT BOOT CAMP

TRAINING: ARCHITECTURE

SOLID

- ▶ **Single Responsibility**

- ▶ Open / Closed
- ▶ Liskov Substitution
- ▶ Interface Segregation

- ▶ **Dependency Inversion**

Single Responsibility Principle

Open Closed Principle

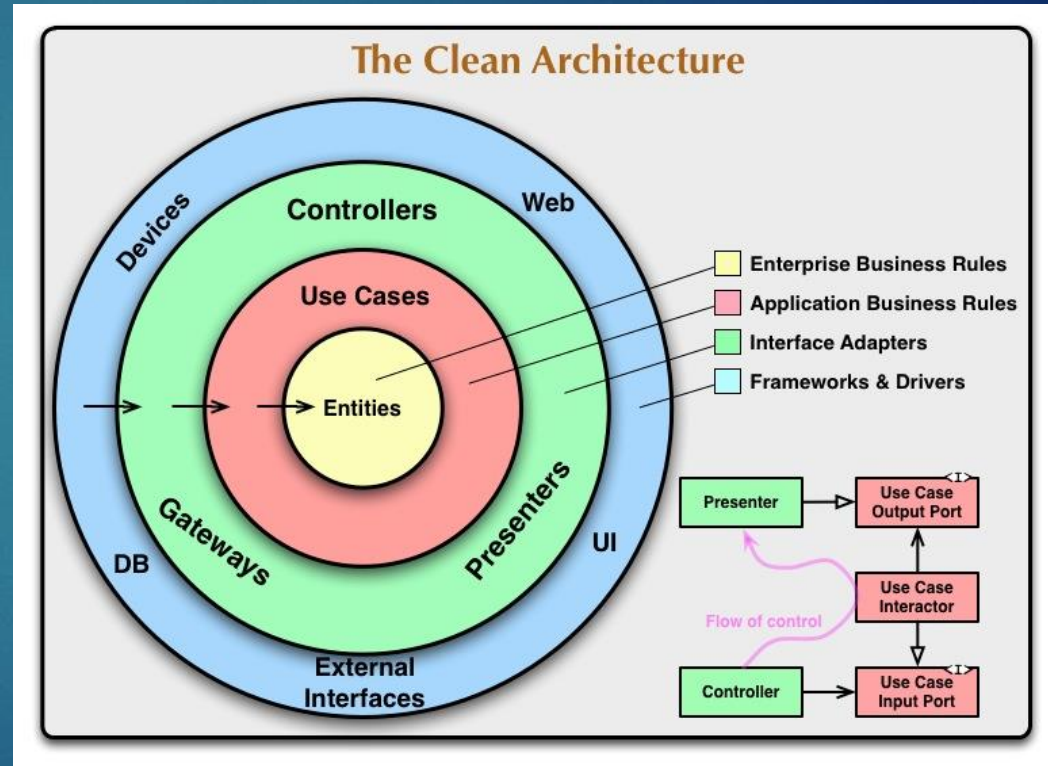
Liskov Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle

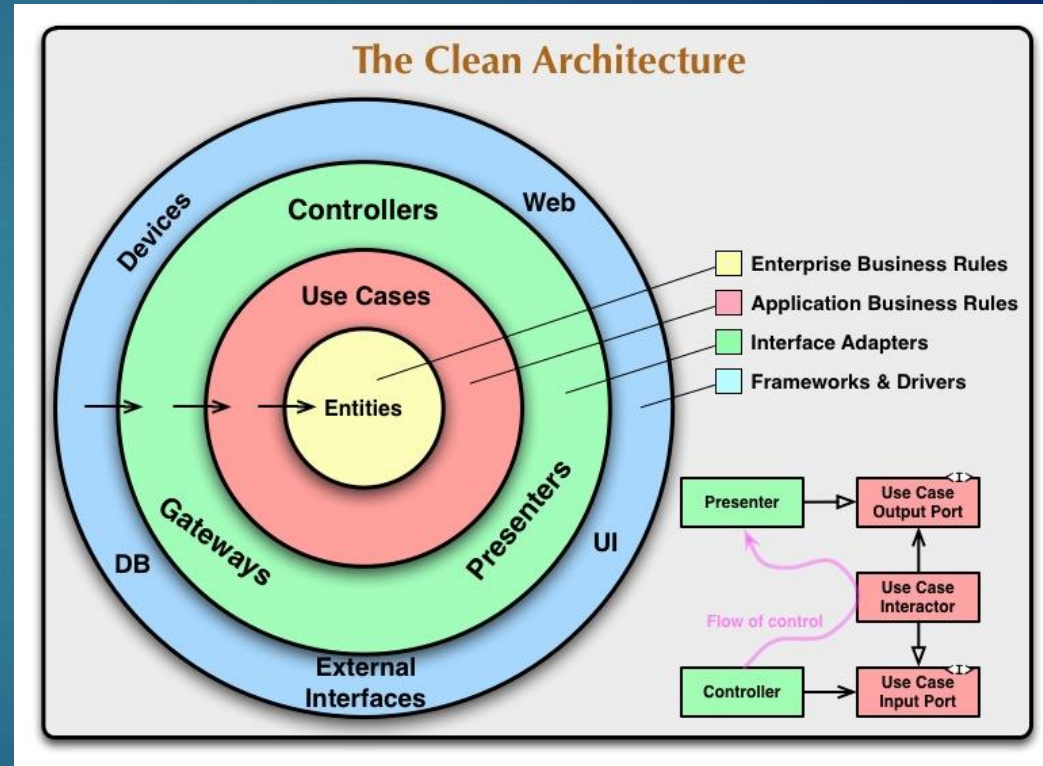
Clean Architecture

- ▶ SOLID extended to Components
- ▶ High Level or Core Components should not depend on Low Level Components or Details
- ▶ Components have Single Responsibility
- ▶ Details can Change



Components

- ▶ Database
- ▶ DAL
- ▶ Entities
- ▶ Web API
- ▶ UI



CLI Commands

- ▶ mkdir – Create Directory
- ▶ cd – Change Directory
- ▶ Add project:
 - ▶ dotnet new classlib
 - ▶ dotnet new webapi
- ▶ Add reference:
 - ▶ dotnet add reference [path]/[name.csproj]
 - ▶ dotnet add package Dapper
 - ▶ dotnet add package MySql.Data

Angular CLI

- ▶ Install Angular CLI
 - ▶ `npm install -g @angular/cli`
- ▶ Create a new Angular App
 - ▶ `ng new [app-name]`
- ▶ Change Directory
 - ▶ `cd [app-name]`
- ▶ Run the Application
 - ▶ `ng serve`



EXAMPLE

CREATING THE APPLICATION AND PROJECTS

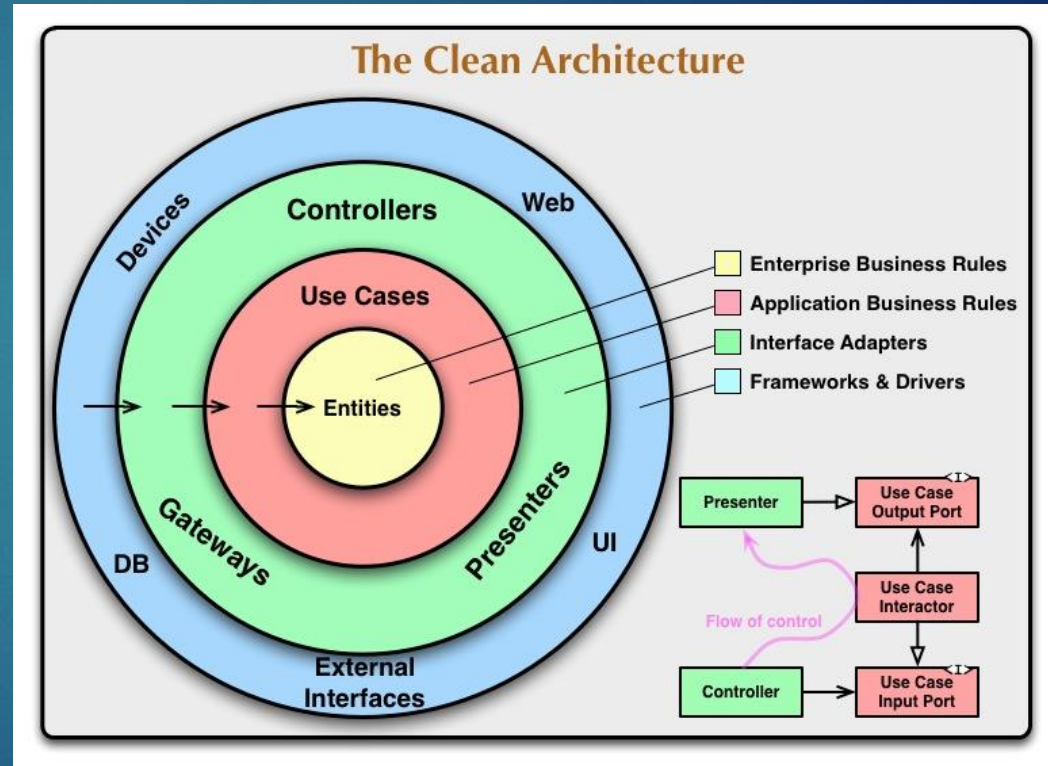


TEAM PROJECT

CREATING THE APPLICATION AND PROJECTS

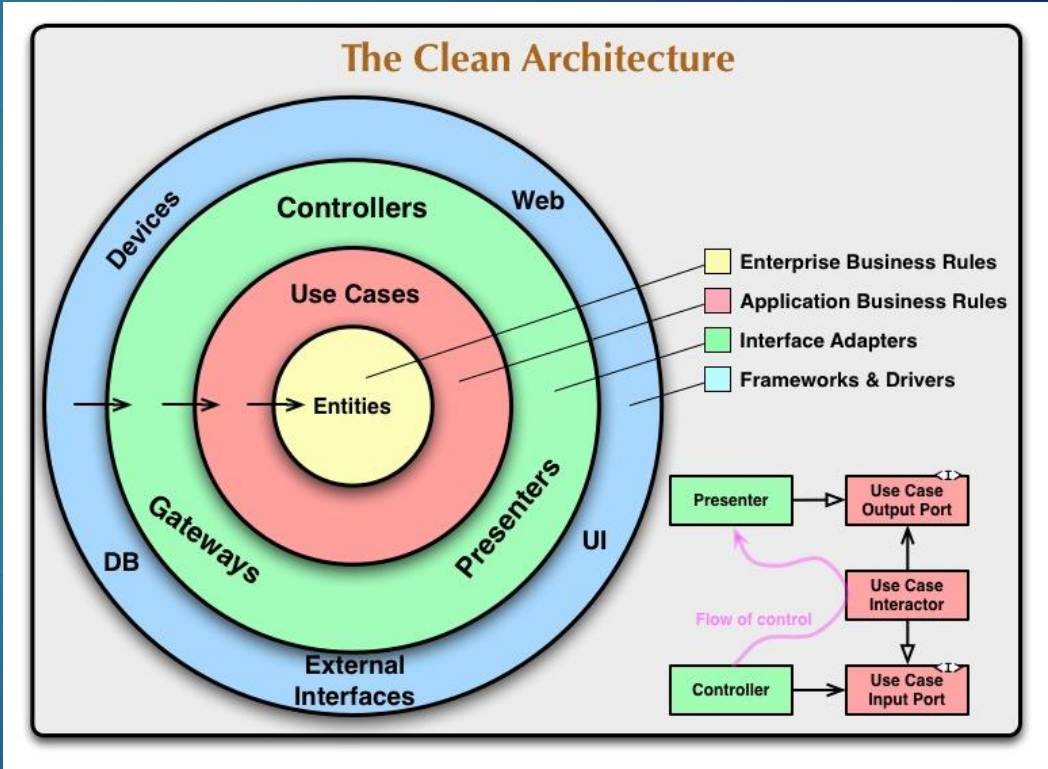
Entities

- ▶ Class with Properties or Data only
- ▶ Center of Clean Architecture
- ▶ Relates to Entity in Data Model
- ▶ Used to transfer data in application
- ▶ POCO – Plain Old CLI Object



100

- # The Clean Architecture
-
- The diagram illustrates 'The Clean Architecture' as a series of concentric layers, each representing a different level of abstraction and dependency. The layers are:
- Entities** (Yellow): The core business logic, independent of any external concerns.
 - Use Cases** (Pink): The application business rules, which depend on entities but are independent of frameworks and interfaces.
 - Controllers** (Light Green): The interface adapters, which depend on use cases and frameworks/drivers.
 - Presenters** (Light Green): The interface adapters, which depend on use cases and frameworks/drivers.
 - Gateways** (Light Blue): The frameworks & drivers, which depend on controllers and presenters.
 - External Interfaces** (Light Blue): The frameworks & drivers, which depend on controllers and presenters.
- External dependencies are shown as arrows pointing from the outer layers towards the inner layers. The diagram also includes a detailed view of the flow of control and data between components:
- Flow of control** (Pink arrow): From the **Controller** to the **Use Case Input Port**.
 - Flow of data** (Black arrow): From the **Use Case Input Port** to the **Use Case Interactor**, then to the **Use Case Output Port**, and finally to the **Presenter**.
 - Flow of control** (Pink arrow): From the **Presenter** back to the **Controller**.



Dependency Inversion

- ▶ Do not want Inner Components dependent on Outer Components
- ▶ Create interfaces for data repositories
- ▶ Entities only know about the interface
- ▶ Implementation passed in constructor
 - ▶ Known as constructor injection
- ▶ Could use DI/IOC framework

Single Responsibility Principle

Open Closed Principle

Liskov Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle

EXAMPLE

CREATING THE ENTITIES

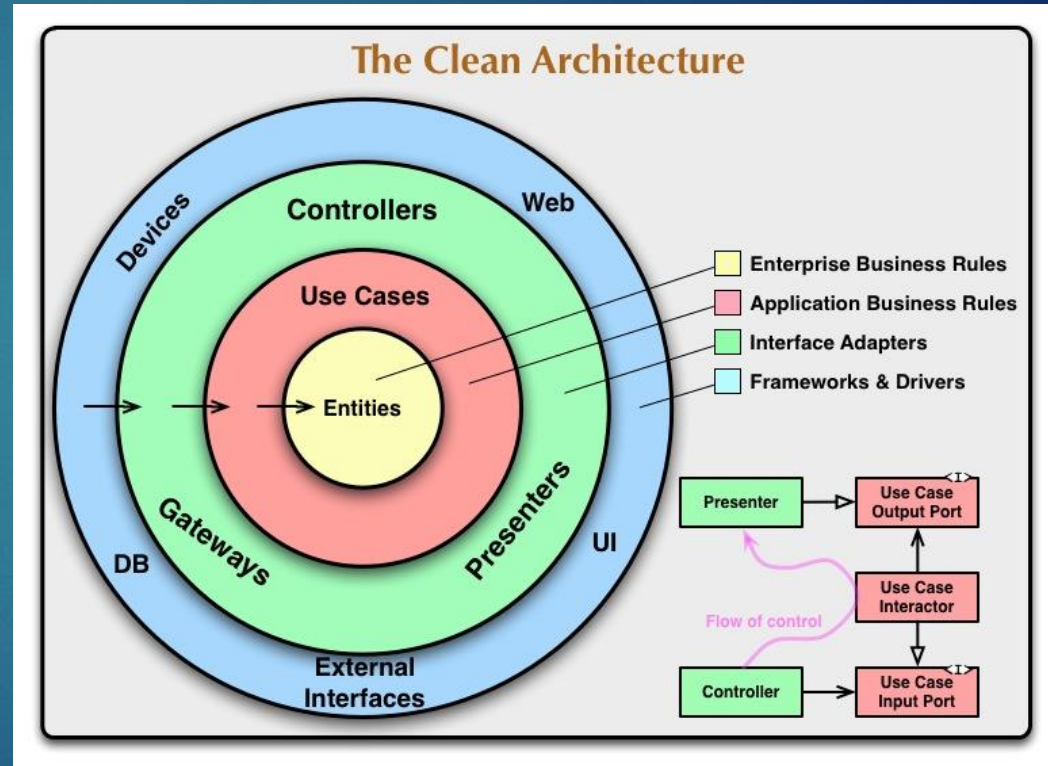


TEAM PROJECT

CREATING THE ENTITIES

DAL / Repository

- ▶ DAL – Data Access Layer
- ▶ Separates data access from the rest of the application.
- ▶ Allows for changes in database
- ▶ Repository pattern
 - ▶ Separation for each entity



EXAMPLE

CREATING THE DAL

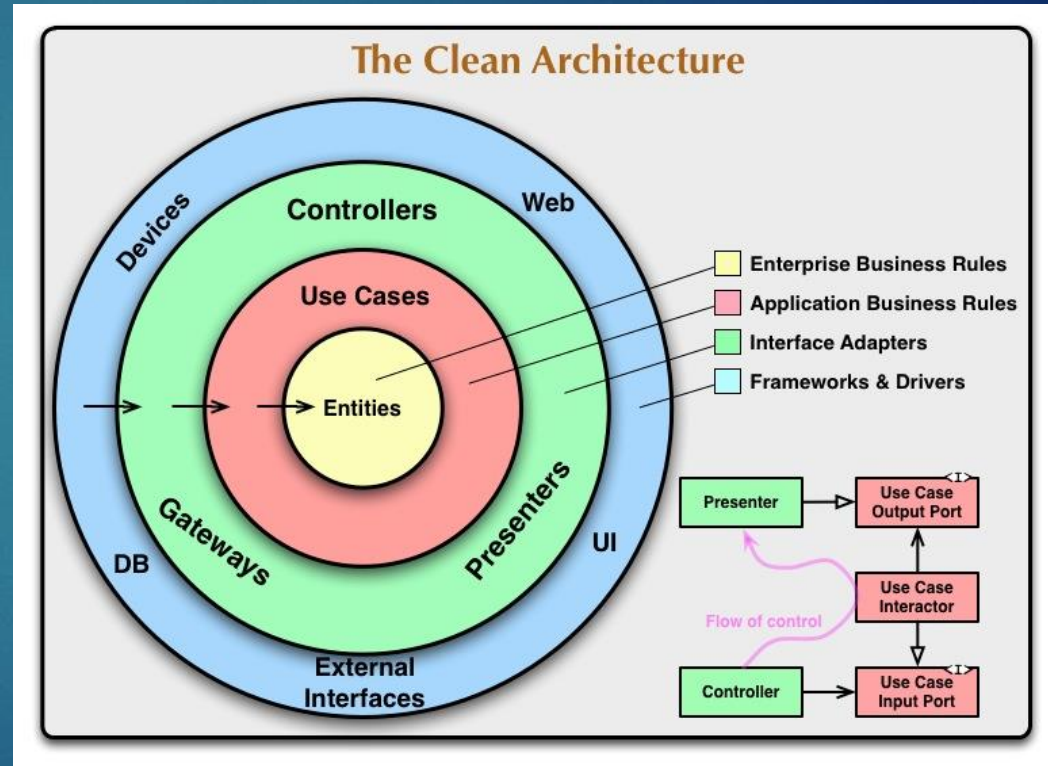


TEAM PROJECT

CREATING THE DAL

Web API

- ▶ Separates UI from the rest of the application.
- ▶ Allows for changes in UI
- ▶ Allows for multiple UI's.





EXAMPLE

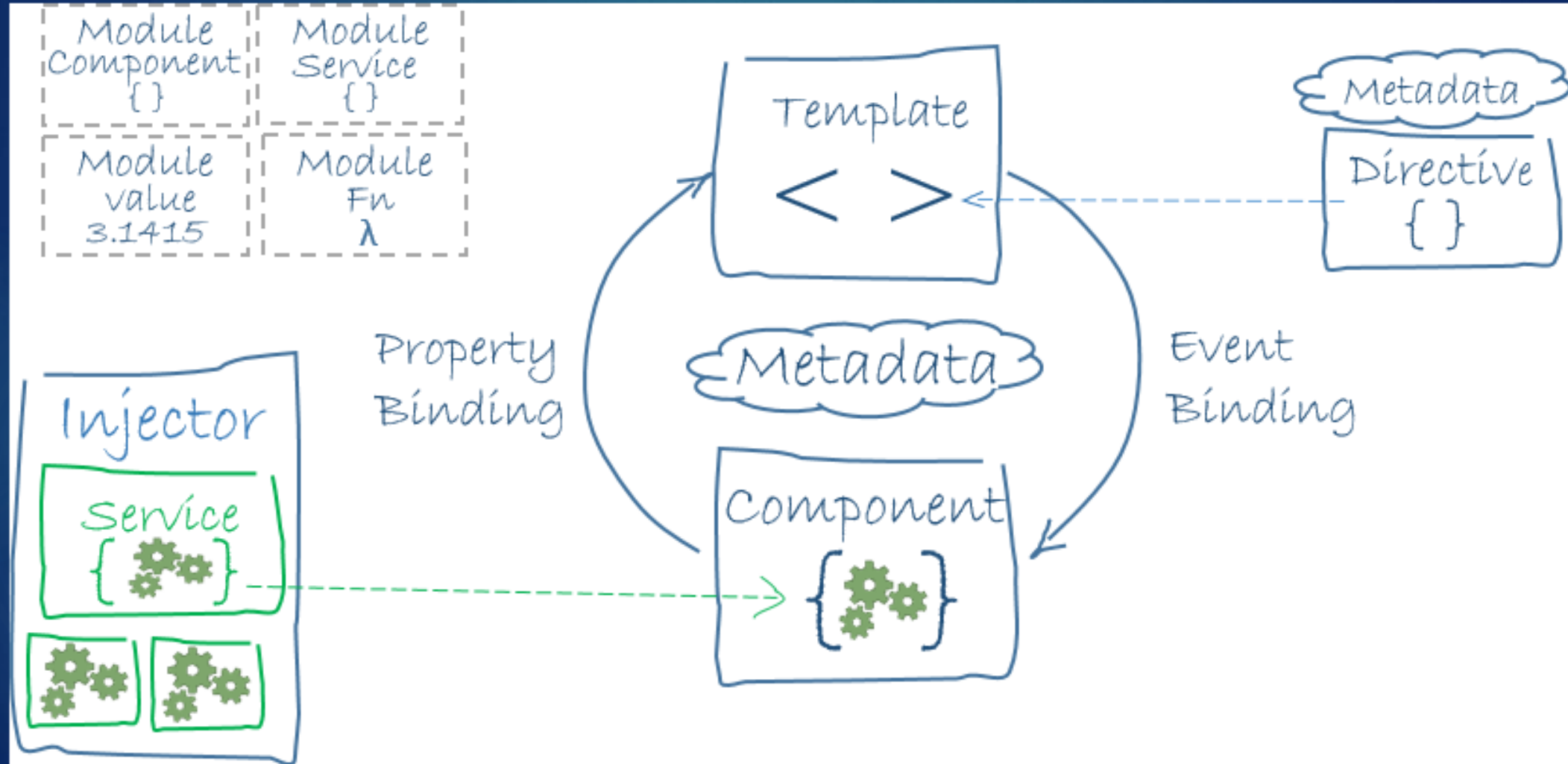
CREATING THE WEB API



TEAM PROJECT

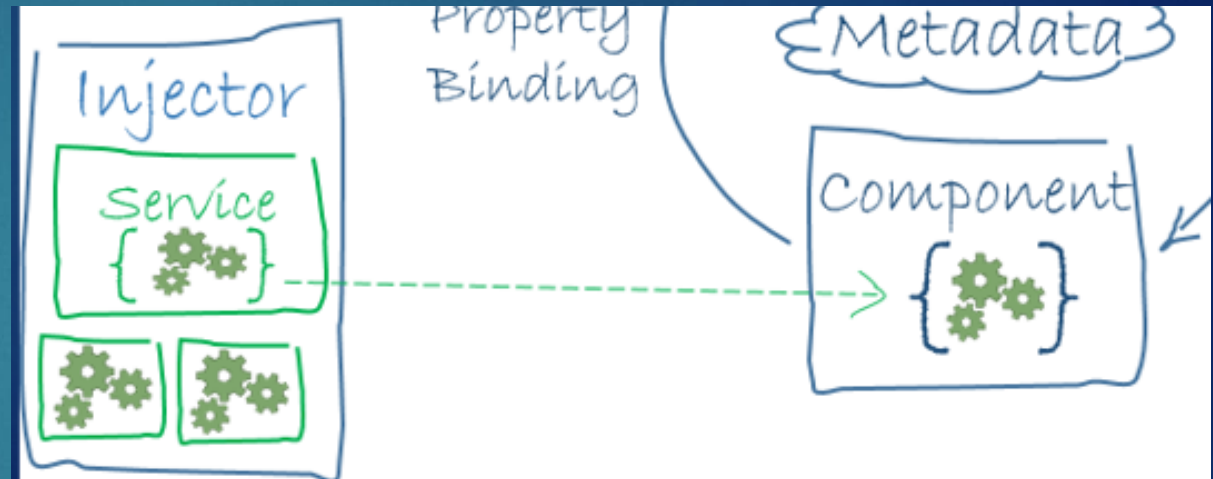
CREATING THE WEB API

Architecture of Angular



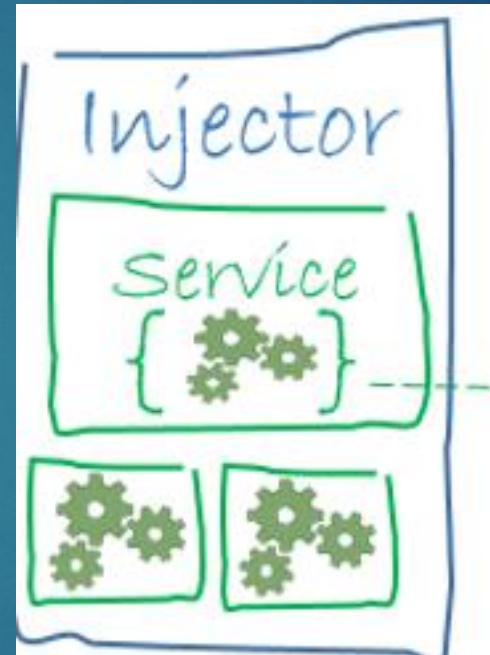
UI Entities

- ▶ Similar to Entities in C#
- ▶ In memory data
- ▶ Passed from Service to Component
- ▶ Used in Template



UI Services

- ▶ Interacts with Web API
- ▶ Retrieves data
- ▶ Could perform other logic
- ▶ Single Responsibility
- ▶ Abstracts data access from Component



EXAMPLE

CREATING THE UI IN ANGULAR



TEAM PROJECT

CREATING THE UI IN ANGULAR

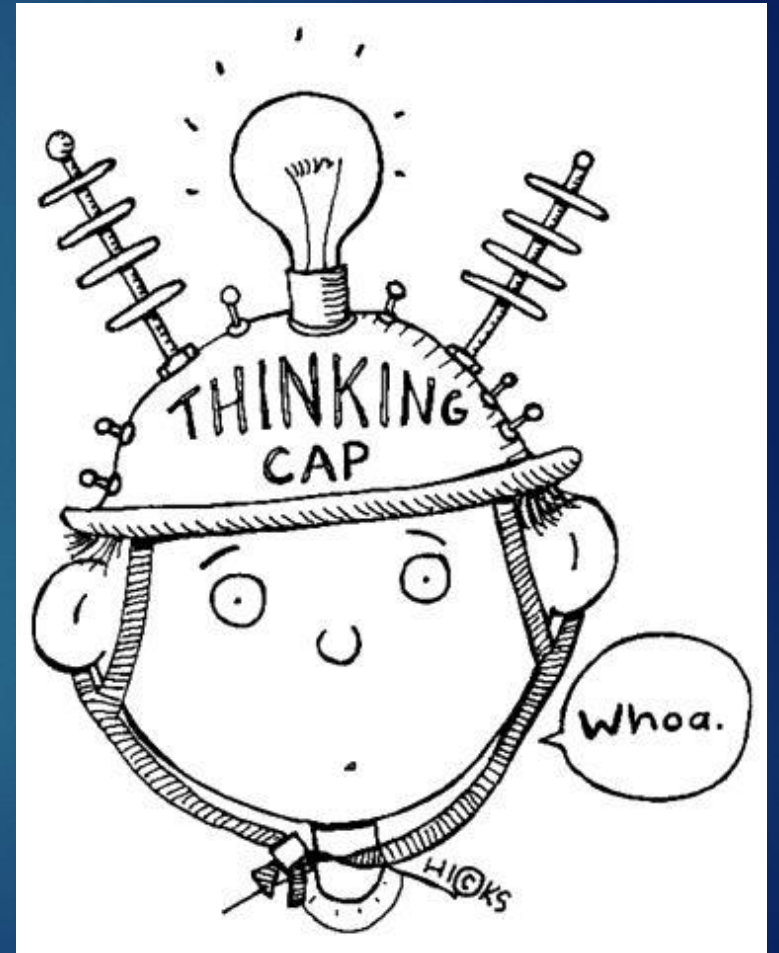
What about Security?

- ▶ Authentication
- ▶ Authorization
- ▶ Threats:
 - ▶ Sql Injection
 - ▶ Cross Site Scripting
 - ▶ Cross Site Request Forgery
 - ▶ OWASP top 10



ASSESSMENT

ARCHITECTURE



QUICK REVIEW

ARCHITECTURE



Not really a sign you'd want to see whilst driving through an eerily quiet neighbourhood...

Additional Resources

- ▶ Clean Architecture

- ▶ <https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html>

- ▶ Scotch.io

- ▶ <https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design>

- ▶ Design Patterns

- ▶ <http://www.dofactory.com/net/design-patterns>

- ▶ Refactoring

- ▶ <https://refactoring.guru/>