Desenvolvimento de Sistemas II

Agenda

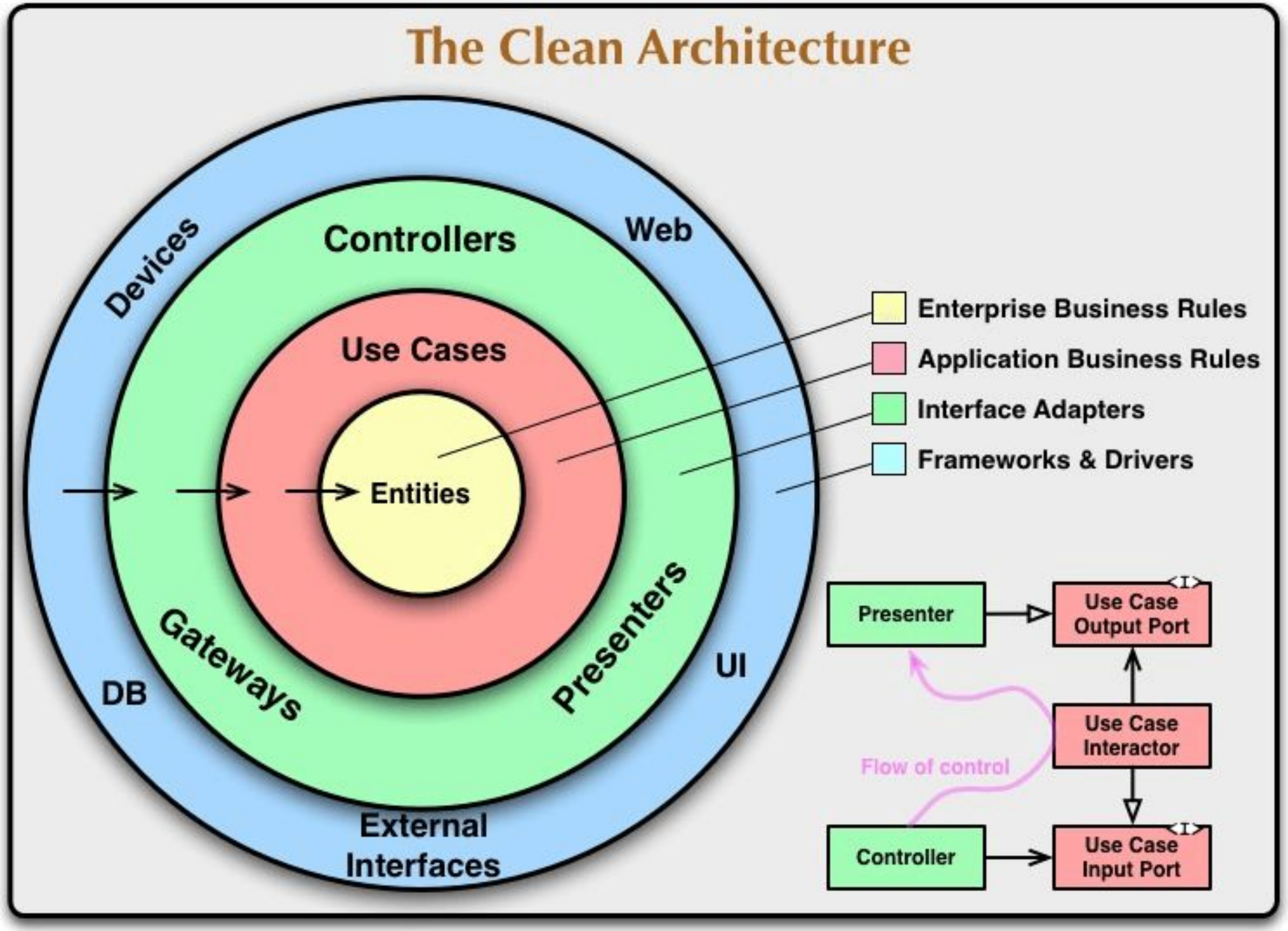
Melhores práticas de programação

Style Guides de linguagens

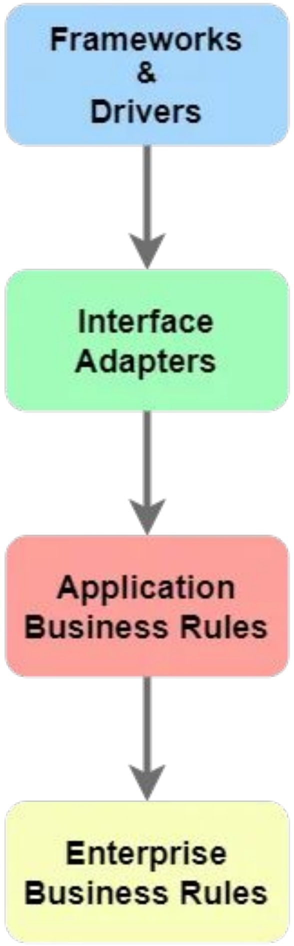
Indentação

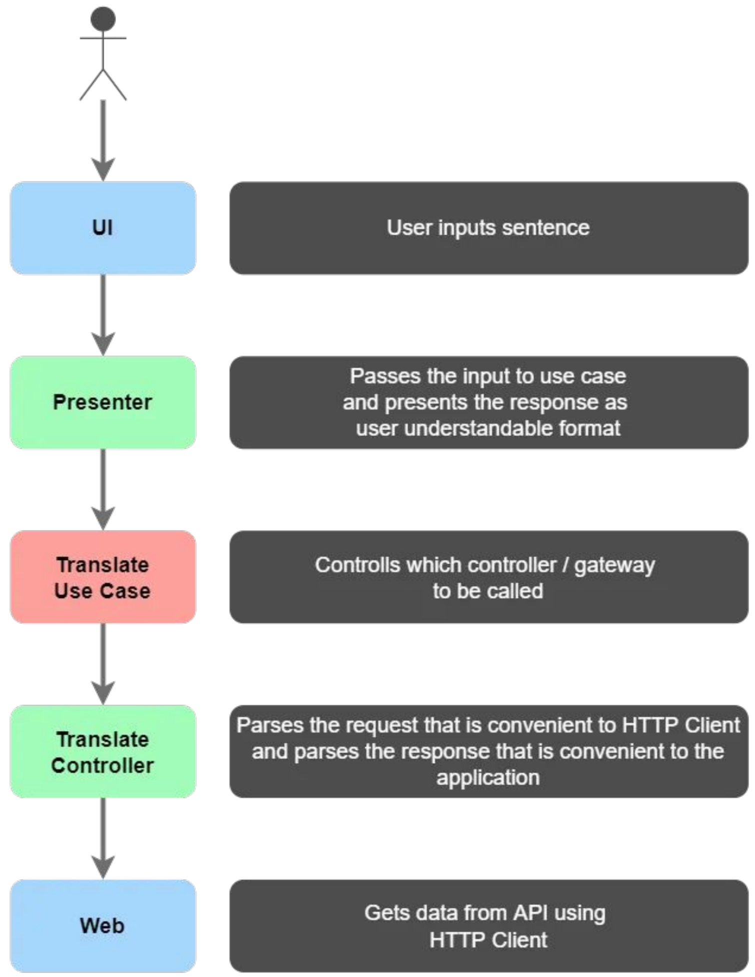
Legibilidade

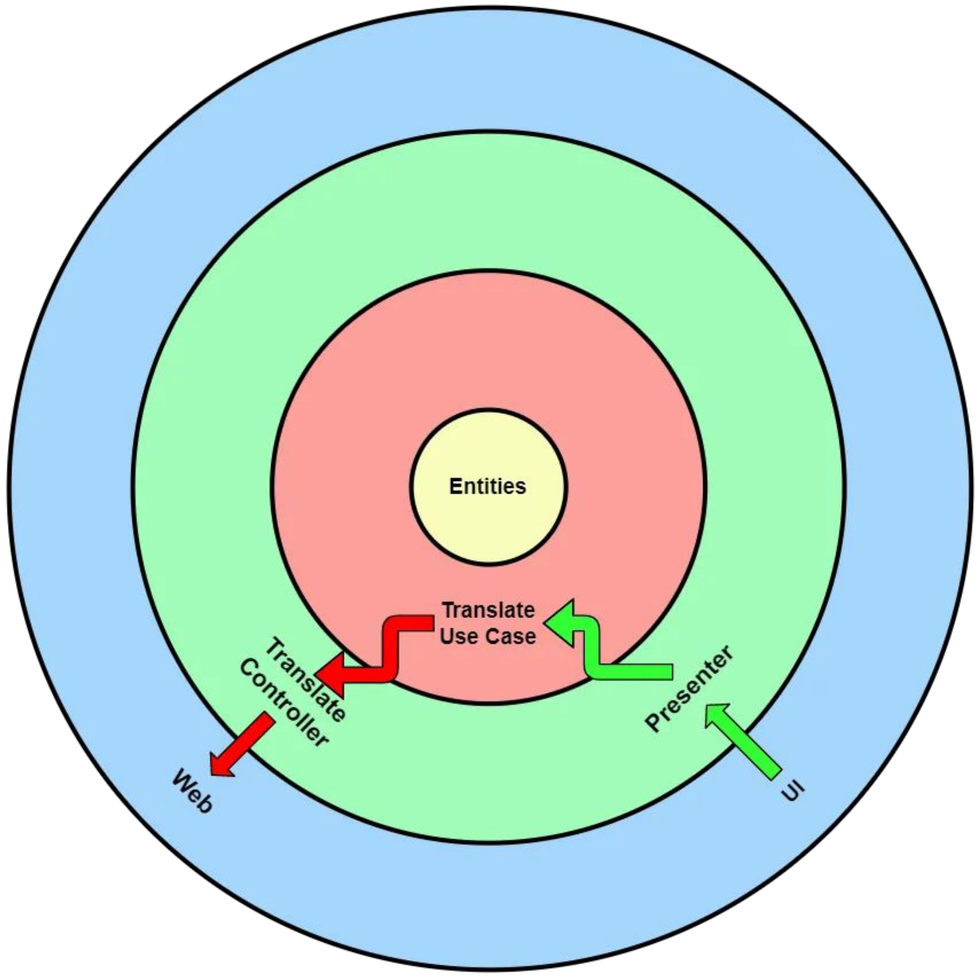
Correção- CleanCode

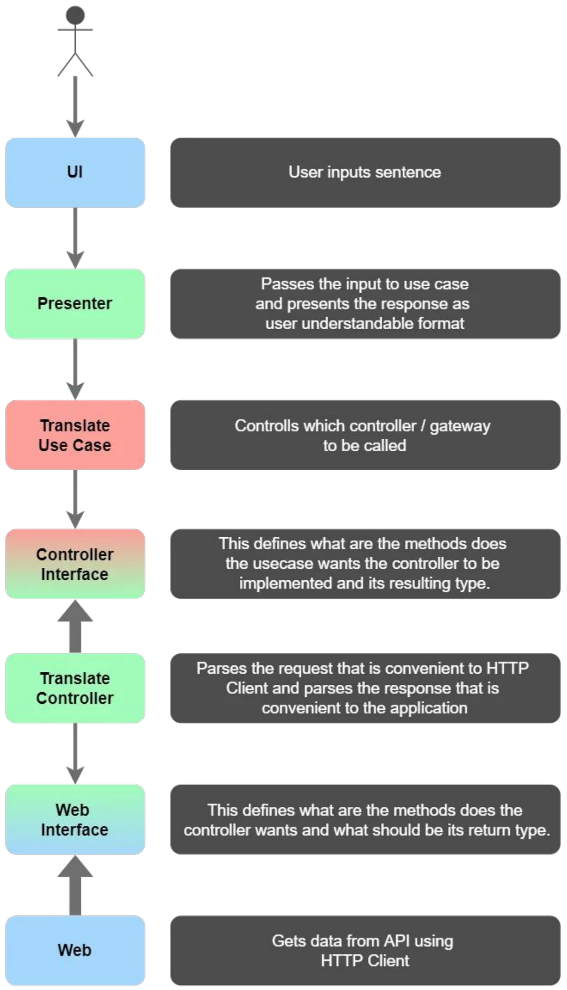


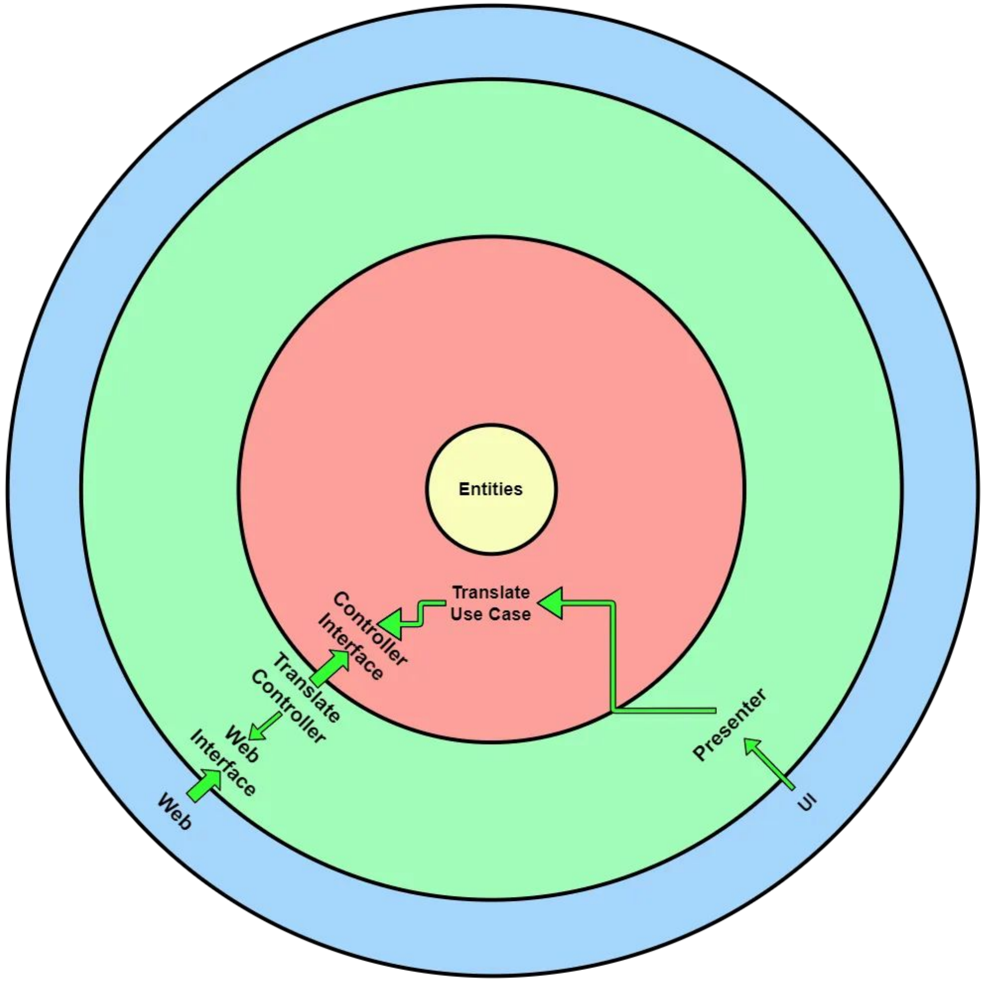
The Clean Architecture
Devices
Web
Controllers
Use Cases
Entities
Gateways
Presenters
DB
UI
External Interfaces
Enterprise Business Rules
Application Business Rules
Interface Adapters
Frameworks & Drivers
Presenter
Use Case Output Port
Use Case Interactor
Controller
Use Case Input Port
Flow of control
I
I

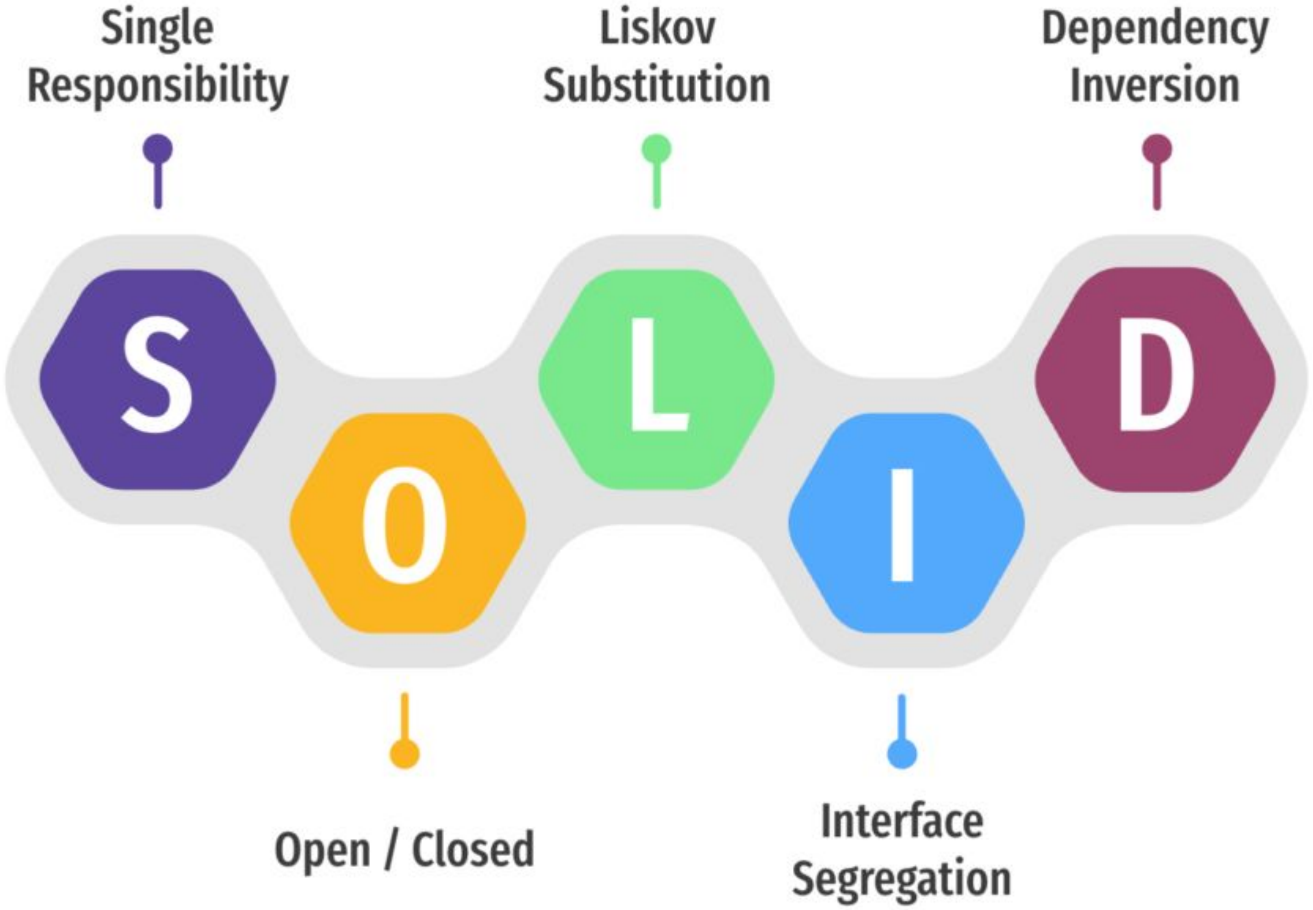

Frameworks
&
Drivers
Interface
Adapters
Application
Business Rules
Enterprise
Business Rules


UI
User inputs sentence
Presenter
Passes the input to use case and presents the response as user understandable format
Translate Use Case
Controlls which controller / gateway to be called
Translate Controller
Parses the request that is convenient to HTTP Client and parses the response that is convenient to the application
Web
Gets data from API using HTTP Client


Entities
Translate Use Case
Translate Controller
Presenter
Web
UI


UI
User inputs sentence
Presenter
Passes the input to use case and presents the response as user understandable format
Translate Use Case
Controlls which controller / gateway to be called
Controller Interface
This defines what are the methods does the usecase wants the controller to be implemented and its resulting type.
Translate Controller
Parses the request that is convenient to HTTP Client and parses the response that is convenient to the application
Web Interface
This defines what are the methods does the controller wants and what should be its return type.
Web
Gets data from API using HTTP Client


Entities
Translate Use Case
Controller Interface
Translate Controller
Web Interface
Web
Presenter
UI


Single Responsibility
Liskov Substitution
Dependency Inversion
S
O
L
I
D
Open / Closed
Interface Segregation

Mão na massa - Exercício

```
public class RestService
        {
                private HttpClient client;
         private Post post;
         private ObservableCollection<Post> posts;
                private JsonSerializerOptions serializerOptions;
         public RestService()
         {
             client = new HttpClient();
                     serializerOptions= new JsonSerializerOptions{
                                   PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
                                   WriteIndented = true
                     };
                }

                public async Task<ObservableCollection<Post>> getPostAsync()
                {

                     Uri uri = new Uri("https://jsonplaceholder.typicode.com/posts");
                     try
                     {
                            HttpResponseMessage response = await client.GetAsync(uri);
                            if (response.IsSuccessStatusCode)
                            {
                                   string content = await response.Content.ReadAsStringAsync();
                                   posts = JsonSerializer.Deserialize<ObservableCollection<Post>>(content, serializerOptions);
                            }
                     }
                     catch (Exception ex)
                     {
                            Debug.WriteLine(@"\tERROR {0}", ex.Message);
                     }

                     return posts;
                }
    }
```

https://github.com/helpdeveloper

https://medium.com/luizalabs/criando-uma-aplica%C3%A7%C3%A3o-modular-muito-al%C3%A9m-do-clean-architecture-5dde3687c5d6

https://code-maze.com/onion-architecture-in-aspnetcore/

https://balta.io/blog/clean-code

https://betterprogramming.pub/the-clean-architecture-beginners-guide-e4b7058c1165