**Clean Architecture API - .NET Web API**

**📌 Përshkrimi**

Ky është një projekt **.NET Web API** i ndërtuar duke ndjekur **Clean Architecture**, që siguron modularitet, testueshmëri dhe mirëmbajtje të lehtë të kodit.  
Projekti përfshin implementimin e **Prototype Pattern** (Creational) për klonimin e objekteve dhe **Adapter Pattern**(Structural) për të përshtatur formatin e të dhënave.

**🛠 Teknologjitë e Përdorura**

✅ **.NET 8 Web API**  
✅ **C# & ASP.NET Core**  
✅ **Clean Architecture**  
✅ **Entity Framework Core** (për bazën e të dhënave)  
✅ **Swagger UI** (për dokumentacionin e API-së)  
✅ **Dependency Injection (DI)**  
✅ **Prototype Pattern**  
✅ **Adapter Pattern**  
✅ **Unit Tests me xUnit & Moq**

**📂 Struktura e Projektit**

CleanArchitectureApi (Solution)

│-- ApplicationCore (Class Library) → Përmban entitetet dhe interfejsat

│ ├── Entities/ → Modelet dhe Prototype Pattern

│ ├── Interfaces/ → Interfejsat për Repository Pattern

│-- Infrastructure (Class Library) → Implementimi i repositorëve dhe adapterëve

│ ├── Repositories/ → Implementimi i IProductRepository

│ ├── Adapters/ → Implementimi i Adapter Pattern

│ ├── Data/ → DbContext & konfigurimi i bazës së të dhënave

│-- WebAPI (ASP.NET Core Web API) → Shtresa e API-së

│ ├── Controllers/ → Endpoints për produktet

│ ├── Program.cs → Konfigurimi i aplikacionit

│ ├── appsettings.json → Konfigurimet e API-së

│-- Tests (xUnit Testing) → Teste për logjikën e biznesit

**🎯 Arkitektura e Projektit**

**Clean Architecture** ndan përgjegjësitë në shtresa për të përmirësuar modularitetin:

**✅ 1. ApplicationCore (Layer-i i Domain-it)**

* Përmban **entitetet**, **interfejsat**, dhe **logjikën e biznesit**.
* Nuk ka varësi nga teknologji të jashtme (p.sh., baza e të dhënave).
* Implementon **Prototype Pattern**.

**✅ 2. Infrastructure (Layer-i i të Dhënave)**

* Implementon interfejsat nga ApplicationCore.
* Përmban **Repository Pattern** për qasjen në të dhëna.
* Implementon **Adapter Pattern** për formatin e të dhënave.

**✅ 3. WebAPI (Layer-i i Prezantimit)**

* Përmban **Controllers** për të ekspozuar API endpoints.
* Implementon **Dependency Injection** për varësitë.
* Konfigurimi i **Swagger UI** për dokumentacionin e API-së.

**🔄 Implementimi i Pattern-eve**

**🌀 Prototype Pattern**

**Qëllimi:** Lejon klonimin e objekteve pa krijuar instanca të reja manualisht.  
📌 **Vendndodhja:** ApplicationCore/Entities/Product.cs

public abstract class Prototype

{

public abstract Prototype Clone();

}

public class Product : Prototype

{

public string Name { get; set; }

public decimal Price { get; set; }

public override Prototype Clone()

{

return (Prototype)this.MemberwiseClone();

}

}

📌 **Shembull përdorimi:**

var originalProduct = new Product { Name = "Laptop", Price = 1200 };

var clonedProduct = (Product)originalProduct.Clone();

**🔌 Adapter Pattern**

**Qëllimi:** Lejon përshtatjen e të dhënave në një format të ndryshëm, si JSON.  
📌 **Vendndodhja:** Infrastructure/Adapters/JsonAdapter.cs

public interface IJsonConverter

{

string ConvertToJson(List<Product> products);

}

public class JsonAdapter : IJsonConverter

{

public string ConvertToJson(List<Product> products)

{

return JsonSerializer.Serialize(products);

}

}

📌 **Shembull përdorimi në Controller:**

[HttpGet("json")]

public IActionResult GetProductsAsJson()

{

var products = \_repository.GetAllProducts();

var adapter = new JsonAdapter();

var jsonResult = adapter.ConvertToJson(products);

return Ok(jsonResult);

}

**⚙️ Konfigurimi dhe Ekzekutimi i Projektit**

**Klonimi i Projektit**

git clone https://github.com/shubi06/CleanArchitectureApi.git

cd CleanArchitectureApi

**2️⃣ Instalimi i Varësive**

Sigurohu që ke të instaluar .NET SDK 8. Pastaj ekzekuto:

dotnet restore

**3️⃣ Ndërtimi dhe Ekzekutimi**

dotnet build

dotnet run --project WebAPI

**4️⃣ Testimi i API-së**

**Swagger UI** është i aktivizuar dhe mund të qaset për testim të API-së:

http://localhost:5000/swagger

Ose përdor **Postman** për të bërë një **GET request**:

GET http://localhost:5000/api/products

**✅ Testimi me xUnit & Moq**

Projekti përfshin **unit tests** për **repository dhe adapter-in**.  
📌 **Vendndodhja:** Tests/

**Ekzekutimi i testeve:**

dotnet test

Shembull testi për JsonAdapter:

public class JsonAdapterTests

{

[Fact]

public void ConvertToJson\_ShouldReturnValidJson()

{

var adapter = new JsonAdapter();

var products = new List<Product> { new Product { Name = "Phone", Price = 800 } };

var jsonResult = adapter.ConvertToJson(products);

Assert.Contains("\"Name\":\"Phone\"", jsonResult);

}

}

**📜 Dokumentacioni i API-së (Swagger UI)**

📌 Swagger është aktivizuar dhe mund të hapet në browser:

http://localhost:5000/swagger/index.html

**👥 Autorët**

👤 **Shuajb Bunjaku, Gentrit Gashi, Amela Abdullahu**