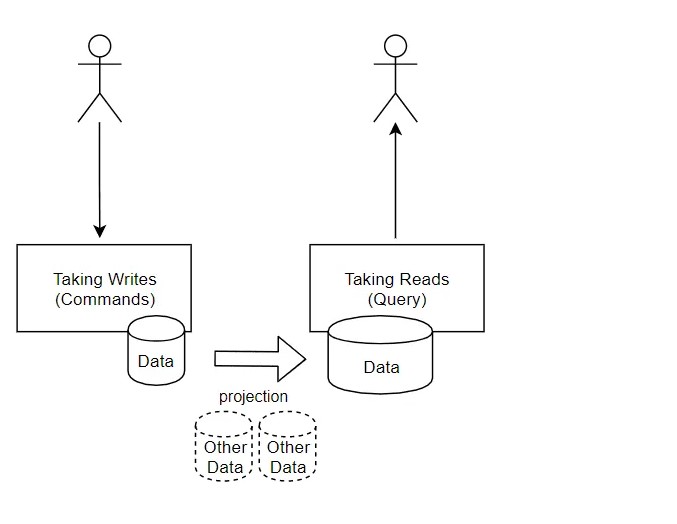
## **What Is CQRS?**

CQRS, Command Query Responsibility Segregation is a design pattern that separates the read and write operations of a data source.



## **Pros Of CQRS الايجابيات**

### **Optimised Data Transfer Objects: كائنات نقل البيانات الأمثل**

Thanks to the segregated approach of this pattern, we will no longer need those complex model classes within our application. Rather we have one model per data operation that gives us all the flexibility in the world.

### **Highly Scalable قابلة للتطوير بدرجة كبيرة**

Having control over the models in accordance with the type of data operations makes your application highly scalable in the long run.

### **Improved Performance تحسين الأداء**

Practically speaking there are always 10 times more Read Operations as compared to the Write Operation. With this pattern you could speed up the performance on your read operations by introducing a cache or NOSQL Db like Redis or Mongo. CQRS pattern will support this usage out of the box, you would not have to break your head trying to implement such a cache mechanism.

### **Secure Parallel Operations عمليات متوازية آمنة**

Since we have dedicated models per oprtation, there is no possibility of data loss while doing parellel operations.

## **Cons Of CQRS سلبيات**

### **Added Complexity and More Code التعقيد المضاف والمزيد من التعليمات البرمجية**

The one thing that may concern a few programmers is that this is a code demanding pattern. In other words, you will end up with at least 3 or 4 times more code-lines than you usually would. But everything comes for a price. This according to me is a small price to pay while getting the awesome features and possibilities with the pattern.

-----------------------------------------------------------------------------------------------------------------------------------

### **The Mediator Pattern**

Mediator pattern is yet another design pattern that dramatically reduces the coupling between various components of an application by making them communicate indirectly, usually via a special mediator object. We will dive deep into Mediator in another post. Essentially, the Mediator pattern is well suited for CQRS implementation.

#### **Installing the required Packages**

Install-Package Microsoft.EntityFrameworkCoreInstall-Package Microsoft.EntityFrameworkCore.RelationalInstall-Package Microsoft.EntityFrameworkCore.SqlServerInstall-Package MediatRInstall-Package MediatR.Extensions.Microsoft.DependencyInjectionInstall-Package Swashbuckle.AspNetCoreInstall-Package Swashbuckle.AspNetCore.Swagger

https://codewithmukesh.com/blog/cqrs-in-aspnet-core-3-1/