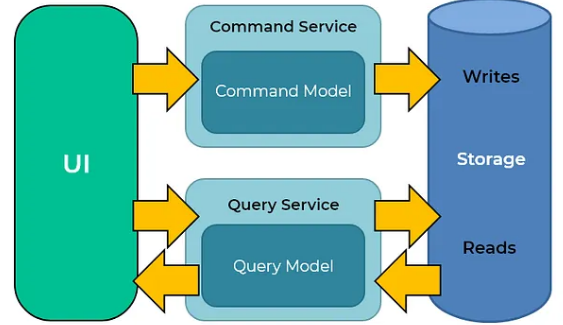
**CQRS (Command Query Responsibility Segregation) Nedir?**



CQRS, ana odağı write (yazma) ve read (okuma) sorumluluklarının ayrıştırılmasına dayanan bir mimari tasarım modelidir. CQRS mimarisi, CQS ilkesi baz alınarak kurulmuştur. CQS’in ana fikrinden bahsetmek gerekirse; bir metot objenin durumunu değiştirmelidir ya da geriye bir sonuç dönmelidir, ancak 2 işlemi birden yapmamalıdır.

Uygulamalarınızda CQRS mimari modeline göre oluşturursanız; uygulamanızın performansını, ölçeklenebilirliğini ve güvenliğini en üst düzeye çıkarabilirsiniz.

Bu yaklaşımda metotlar 2 farklı modele ayrılmalıdır:

* **Commands:**Objenin veya sistemin durumunu değiştirir.
* **Queries:** Sadece sonucu geriye döner herhangi bir objenin veya sistemin durumunu değiştirmez.

**Commands**

Yeni bir veri eklemek ya da var olan veri üzerinde güncelleme yapmak için kullanılır. Örnek vermek gerekirse; Insert, Update, Delete. Geriye veri döndürmez.

**Queries**

Veritabanından veri almak için kullanılır. Geriye sadece belirtilen modeli döner ve veri üzerinde herhangi bir değişiklik yapmaz. Oluşturacağımız Query’lerimiz genellikle ‘Get’ ön eki ile isimlendirilir.

**CQRS Ne Zaman Kullanılmalı ?**

Birbirinden ayrı sistemlerde olası bir servisin hata vermesi durumunda bu hatanın sistemin akışına olumsuz yönde etkisi olmuyorsa kullanılabilir.

Kompleks iş kurallarının olabileceği veya iş kurallarının sık sık değiştiği yapılarda kullanılabilir.

Yüksek veri trafiğinin olduğu sistemlerde kullanılabilir.

**CQRS’i Ne Zaman Kullanmamalıyız ?**

İş kurallarının basit ve çok değişmediği sistemlerde,

Basit CRUD işlemlerinin yapıldığı sistemlerde

CQRS kullanılması önerilmez.

**CQRS’in Avantajları**

* Read ve write operasyonlarının ayrılması performansı, ölçeklenebilirliği ve güvenliği artırmaya yardımcı olabilir.
* Read ve write işlemleriniz için farklı veritabanları kullanabilirsiniz.(Örneğin, yazma işlemleri için MySQL kullanırken okuma işlemleri için Couchbase kullanabilirsiniz).
* Read ve write işlemleri ayrıldığı için, herhangi yapılacak bir read işleminde write işlemini beklemek zorunda kalmayız.
* Her ekibin farklı Domain Logic’i üzerinde çalışabileceği bir yapı kurulmasına yardımcı olabilir.

**CQRS’in Dezavantajları**

* Kod karmaşıklığını arttırır.
* Event bazlı bir yapıya sahipseniz, uygulamanızın queue’da yer alan hataları ve tekrarlanan işlemleri yönetebilmesi gerekmektedir. Olası failover senaryolarını düşünmediğinizde veri kaybı veya daha büyük sorunlarla karşılaşabilirsiniz.

**MediatR**: Mediator Pattern’inin kullanılmasını sağlayan bir kütüphanedir.

**Mediator Pattern:**Aynı interface üzerinden türeyen nesneler arasındaki iletişimi, tek bir nokta üzerinden sağlar. Örnek vermek gerekirse; uçakların hepsi kule ile iletişime geçer, birbirleriyle doğrudan iletişime geçmezler. Bu örnekte, Mediator nesnesi kule, uçaklar da türeyen sınıflar diyebiliriz.