Deck Of Cards Documentation

Release 0.0.1

Andrew Brad

Oct 22, 2019

Introduction

1 The Project **3**

2 Design Philosphy **5**

2.1 Lean Principles 5

2.2 S.O.L.I.D. Principles 5

2.3 Principles of Microservices 6

3 Running **7**

What is this Project?

The Deck of Cards API is a living project that tries to demonstrate the principles of Clean Architure using Command-Query Responsibility Segregation, domain events, and in-process messaging. It can be used to copy paste the occasional snippet, demonstrate how meaningful tests can be written, and see some advanced Api functionality in Asp.Net Core MVC .

I say 'try' because as developers, we are constantly learning, evolving, and applying a little bit of design opinion into the things we build. This project is meant to capture my ongoing learning and show other developers my victories and failures in developing public-facing, enterprise software.

The project strives to prove out the following features:

- **In Process Messaging:** Using the minimal, yet amazing MediatR library, we can achieve direct messaging or broadcast semantics for executing code, allowing us to develop with a publish/subscribe pattern as a first class citizen, improving our ability to write maintainable code.
- **Robust structured logging with Serilog:** This project uses Serilog to persist logs via Sinks. Old school file sinks? Check. ELK stack? Check. Logging done right - everyone loves this.
- **Integration testing with XUnit:** Sure you think your code works, but can you prove it? I tend to favor integration testing over unit testing, but the project strives to have a good amount of tests that cover core functionality, as well as assert some of the basic configurations to ensure nothing breaks when we tweak the MVC builder.
- **CQRS Request Handlers:** A controller shouldn't be involved in business logic, or care about custom validation rules. To this end, controllers receive pre-validated models, and only dispatch a request object into a direct message broker (Mediatr). This means our Api ends up being a true interface over our core application, and maximizes utility while minimizing breakage.
- **Resilience Engineering with Polly:** We should work towards ensuring our software never fails. Ever. No exceptions. The defacto standard for this in .Net is Polly.

CHAPTER 1

The Project

Why a Deck of Cards?

Most demo apps have to be written and/or presented within the space of 1-2 hours. This means that there's a slew of material out there that ends up being 'to-do lists', and stripped down e-commerce applications.

While these may work for demoing specific pieces of functionality, this project exists as a reference as to how to build testable, maintainable software.

CHAPTER 2

Design Philospshy

Before digging into the code, it's recommended to better understand the philosophy behind engineering.

Everything we do should be in the name of software quality.

Everything.

2.1 Lean Principles

Lean principles guide everything. The tricky part is the fact that quality means different things to different people.

- **External Customers - End Users:** Your end users are the reason we all get paid. Listen to them.
- **Internal Customers - Ops:** Nowadays with Devops, some lines are blurred here, but when it comes to production outages, it's rare a developer is called in to mess around with infrastructure. Partner with your infrastructure team and ensure the software behaves like they would expect it to without any 'tribal knowledge' or hacks.
- **Internal Customers - Engineers:** Your codebase should get you excited about writing code. If it doesn't, have a retrospective with teammates and get on track. It should *not* be difficult to change (fragile), or prone to failure. What was your first day like as a new developer on a codebase? Was the product well documented, and the standards and expectations made clear? Technical debt is something that needs to be measured, managed, or at least observed and refactored continually.

If you'd like to read about the history and interactions between Lean Manufacturing and Software Design, I'd highly recommend Implementing Lean Software Development. Next, let's cover some principles already established in the industry.

2.2 S.O.L.I.D. Principles

WIP

2.3 Principles of Microservices

WIP

CHAPTER 3

Running

The best dev experience involves cloning the repo, and running a script to build, test, or run the project.

Unfortunately, we're not quite there yet.

//todo