

Hands-On Software Architecture with C# 8 and .NET Core 3

Architecting software solutions using microservices, DevOps,
and design patterns for Azure Cloud

Gabriel Baptista
Francesco Abbruzzese

Packt>

BIRMINGHAM - MUMBAI

Table of Contents

<u>Preface</u>	1
<u>Section 1: Transforming Customer Needs in Real-World Applications</u>	
Chapter 1: Understanding the Importance of Software Architecture	g
Technical requirements	10
What is software architecture?	10
Creating an Azure account	12
Software development process models	14
Reviewing traditional software development process models	14
Understanding the waterfall model principles	14
Analyzing the incremental model	15
Understanding agile software development process models	16
Getting into the Scrum model	17
Enabling aspects to be gathered to design high-quality software	18
Understanding the requirements gathering process	18
Practicing the elicitation of user needs	19
Analyzing requirements	20
Writing the specifications	21
Reviewing the specification	21
Using design thinking as a helpful tool	22
Understanding the principles of scalability, robustness, security, and performance	23
Some cases where the requirements gathering process impacted system results	24
Case 1 – my website is too slow to open that page!	24
Understanding caching	25
Applying asynchronous programming	25
Dealing with object allocation	25
Getting better database access	26
Case 2 – the user's needs are not properly implemented-	27
Case 3 – the usability of the system does not meet user needs	28
Case study – detecting user needs	28
Book case study – introducing World Wild Travel Club	29
Book case study – understanding user needs and system requirements	30
Summary	32
Questions	32
Further reading	33
Chapter 2: Functional and Nonfunctional Requirements	35

Technical requirements	36
How does scalability interact with Azure and .NET Core?	36
Creating a scalable web app in Azure	37
Vertical scaling (Scale up)	40
Horizontal scaling (Scale out)	42
Creating a scalable web app with .NET Core	43
Performance issues that need to be considered when programming in C#	47
String concatenation	48
Exceptions	49
Multithreading environments for better results – do's and don'ts	50
Usability – why inserting data takes too much time	53
Designing fast selection logic	54
Selecting from a huge amount of items	58
The fantastic world of interoperability with .NET Core	59
Creating a service in Linux	61
Book use case – understanding the main types of .NET Core projects	62
Summary	64
Questions	64
Further reading	65
Chapter 3: Documenting Requirements with Azure DevOps	67
Technical requirements	67
Introducing Azure DevOps	67
Organizing your work using Azure DevOps	72
Azure DevOps repository	73
Package feeds	75
Test plans	77
Pipelines	78
Managing system requirements in Azure DevOps	79
Epics work items	80
Features work items	80
Product Backlog items/User Story work items	81
Book use case – presenting use cases in Azure DevOps	82
Summary	86
Questions	87
Further reading	87
<u>Section 2: Architecting Software Solutions in a Cloud-Based Environment</u>	
Chapter 4: Deciding the Best Cloud-Based Solution	91
Technical requirements	91
Different software deployment models	92

Infrastructure as a service and Azure opportunities	92
Security responsibility in IaaS	94
PaaS – a world of opportunities for developers	95
Web apps	96
Azure SQL Server	97
Azure Cognitive Services	99
SaaS – just sign in and get started!	103
Understanding what serverless means	103
Why are hybrid applications so useful in many cases?	104
Use case – a hybrid application	105
Book use case – which is the best cloud platform for this use case?	106
Summary	107
Questions	107
Further reading	108
Chapter 5: Applying a Microservice Architecture to Your Enterprise	
Application	109
Technical requirements	110
What are microservices?	110
Microservices and the evolution of the concept of modules	112
Microservice design principles	113
Containers and Docker	117
When do microservices help?	119
Layered architectures and microservices	119
When is it worth considering microservice architectures?	122
How does .NET Core deal with microservices?	123
.NET Core communication facilities	124
Resilient task execution	126
Using generic hosts	128
Visual Studio support for Docker	132
Azure and Visual Studio support for microservice orchestration	137
Which tools are needed to manage microservices?	140
Defining your private Docker registry in Azure	140
Azure Service Fabric	142
Step 1: Basic information	143
Step 2: Cluster configuration	144
Step 3: Security configuration	146
Azure Kubernetes Service (AKS)	149
Use case – logging microservices	153
Ensuring message idempotency	157
The Interaction library	160
Implementing the receiving side of communication	162
Implementing service logic	165
Defining the microservice's host	171
Communicating with the service	172

Testing the application	174
Summary	174
Questions	175
Further reading	175
Chapter 6: Interacting with Data in C# - Entity Framework Core	177
Technical requirements	178
Understanding ORM basics	178
Configuring Entity Framework Core	181
Defining DB entities	182
Defining the mapped collections	185
Completing the mapping configuration	186
Entity Framework Core migrations	187
Understanding stored procedures and direct SQL commands	191
Querying and updating data with Entity Framework Core	192
Returning data to the presentation layer	195
Issuing direct SQL commands	196
Handling transactions	198
Deploying your data layer	198
Understanding Entity Framework Core advanced feature – global filters	199
Summary	200
Questions	201
Further reading	201
Chapter 7: How to Choose Your Data Storage in the Cloud	203
Technical requirements	204
Understanding the different repositories for different purposes	204
Relational databases	205
NoSQL databases	207
Redis	208
Disk memory	210
Choosing between structured or NoSQL storage	210
Azure Cosmos DB – an opportunity to manage a multi-continental database	212
Cosmos DB client	219
Cosmos DB Entity Framework Core provider	220
Use case – storing data	221
Implementing the destinations/packages database with Cosmos DB	222
Summary	227
Questions	227
Further reading	228
Chapter 8: Working with Azure Functions	229
Technical requirements	229

Understanding the Azure Functions App	230
Consumption Plan	231
App Service Plan	231
Programming Azure Functions using C#	232
Listing Azure Functions templates	236
Maintaining Azure Functions	237
Use case – implementing Azure Functions to send emails	239
First Step – creating Azure Queue Storage	241
Summary	247
Questions	247
Further reading	248
<u>Section 3: Applying Design Principles for Software Delivered in the 21st Century</u>	
Chapter 9: Design Patterns and .NET Core Implementation	251
Technical requirements	251
Understanding design patterns and their purpose	252
Builder pattern	253
Factory pattern	255
Singleton pattern	256
Proxy pattern	259
Command pattern	261
Publisher/Subscriber pattern	263
Dependency Injection pattern	264
Understanding the available design patterns in .NET Core	266
Summary	267
Questions	267
Further reading	268
Chapter 10: Understanding the Different Domains in Software Solutions	269
Technical requirements	270
What are software domains?	270
Understanding domain-driven design	273
Entities and value objects	277
Using SOLID principles to map your domains	281
Aggregates	283
The repository and Unit of Work patterns	284
DDD entities and Entity Framework Core	286
Command Query Responsibility Segregation (CQRS) pattern	287
Command handlers and domain events	290
Event sourcing	293
Use case – understanding the domains of the use case	294
Summary	297

Questions	298
Further reading	298
Chapter 11: Implementing Code Reusability in C# 8	299
Technical requirements	299
Understanding the principles of code reusability	300
What is not code reuse?	300
What is code reuse?	302
Inserting reusability into your development cycle	303
Using .NET Standard for code reuse	304
Creating a .NET Standard library	304
How does C# deal with code reuse?	306
Object-oriented analysis	306
Generics	308
Use case – reusing code as a fast track to deliver good and safe software	309
Summary	310
Questions	311
Further reading	311
Chapter 12: Applying Service-Oriented Architectures with .NET Core	313
Technical requirements	314
Understanding the principles of the SOA approach	314
SOAP web services	318
REST web services	320
The OpenAPI standard	326
REST services authorization and authentication	326
How does .NET Core deal with SOA?	329
A short introduction to ASP.NET Core	331
Implementing REST services with ASP.NET Core	335
ASP.NET Core service authorization	339
ASP.NET Core support for OpenAPI	342
.Net Core HTTP clients	346
Use case – exposing WWTravelClub packages	349
Summary	355
Questions	355
Further reading	356
Chapter 13: Presenting ASP.NET Core MVC	357
Technical requirements	358
Understanding the presentation layers of web applications	358
Understanding the ASP.NET Core MVC structure	359
How ASP.NET Core pipeline works	359
Loading configuration data and using it with the options framework	363
Defining the ASP.NET Core MVC pipeline	368

Defining controllers and ViewModels	373
Understanding Razor Views	378
Learning Razor flow of control statements	379
Understanding Razor View properties	381
Using Razor tag helpers	382
Reusing view code	386
What is new in .NET Core 3.0 for ASP.NET Core?	390
Understanding the connection between ASP.NET Core MVC and design principles	392
Advantages of the ASP.NET Core pipeline	393
Server-side and client-side validation	393
ASP.NET Core globalization	394
The MVC pattern	399
Use case – implementing a web app in ASP.NET Core MVC	400
Defining application specifications	400
Defining the application architecture	401
Defining the domain layer	404
Defining the data layer	407
Defining the application layer	412
Controllers and views	417
Summary	423
Questions	423
Further reading	424
<u>Section 4: Programming Solutions for an Unavoidable Future Evolution</u>	
Chapter 14: Best Practices in Coding C# 8	427
Technical requirements	427
The more complex your code is, the worse a programmer you are	428
Maintainability index	429
Cyclomatic complexity	429
Depth of inheritance	433
Class coupling	434
Lines of code	436
Using a version control system	436
Dealing with version control systems in teams	437
Writing safe code in C#	437
try-catch	437
try-finally and using	438
The IDisposable interface	440
.NET Core tips and tricks for coding	440
WWTravelClub – dos and don'ts in writing code	442
Summary	443
Questions	443

Further reading	443
Chapter 15: Testing Your Code with Unit Test Cases and TDD	445
Technical requirements	446
Understanding automated tests	446
Writing automated (unit and integration) tests	448
Writing acceptance and performance tests	450
Understanding test-driven development (TDD)	451
Defining C# test projects	454
Using the xUnit test framework	455
Advanced test preparation and tear-down scenarios	457
Mocking interfaces with Moq	459
Use case – automating unit tests in DevOps Azure	461
Summary	469
Questions	470
Further reading	470
Chapter 16: Using Tools to Write Better Code	471
Technical requirements	472
Identifying a well-written code	472
Understanding and applying tools that can evaluate C# code	474
Applying extension tools to analyze code	478
Using Microsoft Code Analysis 2019	478
Applying SonarLint for Visual Studio 2019	480
Getting Code Cracker for Visual Studio 2017 as a helper to write better code	481
Checking the final code after analysis	481
Use case – evaluating the C# code before publishing the application	483
Summary	485
Questions	485
Further reading	486
<u>Section 5: Delivering Software Continuously and at a High Quality Level</u>	
Chapter 17: Deploying Your Application with Azure DevOps	489
Technical requirements	490
Understanding SaaS	490
Adapting your organization to a service scenario	490
Developing software in a service scenario	491
Technical implications of a service scenario	491
Adopting a SaaS solution	492
Preparing a solution for a service scenario	493
Use case – deploying our package-management application with Azure Pipelines	496

Creating the Azure Web App and the Azure database	496
Configuring your Visual Studio solution	498
Configuring Azure Pipelines	499
Adding a manual approval for the release	502
Creating a release	504
Summary	506
Questions	507
Further reading	507
Chapter 18: Understanding DevOps Principles	509
Technical requirements	510
Describing DevOps	510
Understanding DevOps principles	511
Defining continuous integration	511
Understanding continuous delivery and multistage environment with Azure DevOps	512
Defining continuous feedback and the related DevOps tools	515
Monitoring you software with Application Insights	516
Using the Test and Feedback tool to enable feedback	522
The WWTravelClub project approach	527
Summary	528
Questions	528
Further Reading	529
Chapter 19: Challenges of Applying CI Scenarios in DevOps	531
Technical requirements	532
Understanding CI	532
Understanding the risks and challenges when using CI	533
Disabling continuous production deployment	534
Incomplete features	535
Unstable solution for testing	538
Understanding the WWTravelClub project approach	542
Summary	542
Questions	543
Further reading	543
Chapter 20: Automation for Software Testing	545
Technical requirements	545
Understanding the purpose of functional tests	546
Using unit testing tools to automate functional tests in C#	548
Testing the staging application	549
Testing a controlled application	550
Use case – automating functional tests	552
Summary	556
Questions	556

Further reading	556
<u>Appendix A: Assessments</u>	557
Chapter 1	557
Chapter 2	557
Chapter 3	558
Chapter 4	558
Chapter 5	559
Chapter 6	559
Chapter 7	560
Chapter 8	560
Chapter 9	561
Chapter 10	562
Chapter 11	562
Chapter 12	563
Chapter 13	563
Chapter 14	564
Chapter 15	564
Chapter 16	565
Chapter 17	565
Chapter 18	566
Chapter 19	566
Chapter 20	567
<u>Other Books You May Enjoy</u>	569
Index	<u>573</u>