

# Contents

<b>Preface</b>	<b>7</b>
Limit of Liability/Disclaimer of Warranty . . . . .	8
PDF & EPUB . . . . .	8
Preface . . . . .	8
Who this book is for . . . . .	8
What this book covers . . . . .	9
Contributing, Errata and Source code . . . . .	12
<b>1 Stream live cryptocurrency prices from the Binance WSS</b>	<b>13</b>
1.1 Objectives . . . . .	13
1.2 Create a new umbrella app . . . . .	13
1.3 Create a supervised application inside an umbrella . . . . .	13
1.4 Connect to Binance's WebSocket Stream using the WebSockets module . . . . .	14
1.5 Decode incoming events using the Jason module . . . . .	17
<b>2 Create a naive trading strategy - a single trader process without supervision</b>	<b>22</b>
2.1 Objectives . . . . .	22
2.2 Initialization . . . . .	22
2.3 How trading strategy will work? . . . . .	26
<b>3 Introduce PubSub as a communication method</b>	<b>33</b>
3.1 Objectives . . . . .	33
3.2 Design . . . . .	33
3.3 Implementation . . . . .	36

<b>4</b>	<b>Mock the Binance API</b>	<b>38</b>
4.1	Objectives	38
4.2	Design	38
4.3	Create “BinanceMock” app	40
4.4	Implement getting exchange info	41
4.5	Implement placing buy and sell orders	42
4.6	Implement order retrieval	46
4.7	Implement callback for incoming trade events	47
4.8	Upgrade trader and config	50
4.9	Test the implementation	52
<b>5</b>	<b>Enable parallel trading on multiple symbols</b>	<b>53</b>
5.1	Objectives	53
5.2	Introduction - architectural design	53
5.3	Implement <code>Naive.SymbolSupervisor</code>	57
5.4	Implement <code>Naive.Leader</code>	58
<b>6</b>	<b>Introduce a <code>buy_down_interval</code> to make a single trader more profitable</b>	<b>67</b>
6.1	Objectives	67
6.2	Why we need to buy below the current price? Feature overview	68
6.3	<code>Naive.Trader</code> implementation	69
6.4	<code>Naive.Leader</code> implementation	71
<b>7</b>	<b>Introduce a trader budget and calculating the quantity</b>	<b>73</b>
7.1	Objectives	73
7.2	Fetch <code>step_size</code>	73
7.3	Append budget and <code>step_size</code> to the Trader’s state inside the Leader	75
7.4	Append budget and <code>step_size</code> to the Trader’s state	75
7.5	Calculate quantity	76
<b>8</b>	<b>Add support for multiple transactions per order</b>	<b>78</b>
8.1	Objectives	78
8.2	The issue with the current implementation	78
8.3	Improve buy order filled callback	80

8.4	Implement buy order “filled” callback . . . . .	82
8.5	Improve sell order callback . . . . .	82
8.6	Test the implementation . . . . .	84
<b>9</b>	<b>Run multiple traders in parallel</b>	<b>85</b>
9.1	Objectives . . . . .	85
9.2	Describe and design the required functionality . . . . .	85
9.3	Implement rebuy inside <code>Naive.Trader</code> . . . . .	86
9.4	Implement rebuy in the <code>Naive.Leader</code> . . . . .	88
9.5	Improve logs by assigning ids to traders . . . . .	91
9.6	Test the implementation . . . . .	93
<b>10</b>	<b>Fine-tune trading strategy per symbol</b>	<b>96</b>
10.1	Objectives . . . . .	96
10.2	Describe and design the required functionality . . . . .	96
10.3	Add docker to project . . . . .	97
10.4	Set up <code>ecto</code> inside the <code>naive</code> app . . . . .	98
10.5	Create and migrate the DB . . . . .	100
10.6	Seed symbols’ settings . . . . .	103
10.7	Update the <code>Naive.Leader</code> to fetch settings . . . . .	105
<b>11</b>	<b>Supervise and autostart streaming</b>	<b>108</b>
11.1	Objectives . . . . .	108
11.2	Describe and design the required functionality . . . . .	108
11.3	Register the <code>Streamer.Binance</code> processes with names . . . . .	109
11.4	Set up <code>ecto</code> inside the <code>streamer</code> app . . . . .	110
11.5	Create and migrate the db . . . . .	111
11.6	Seed default settings . . . . .	113
11.7	Implement the supervision tree and start streaming functionality . . . . .	114
11.8	Implement the stop functionality . . . . .	116
11.9	Implement the autostart streaming functionality . . . . .	117
11.10	Test the implementation . . . . .	120

<b>12 Start, stop, shutdown and autostart trading</b>	<b>122</b>
12.1 Objectives	122
12.2 Describe and design the required functionality	122
12.3 (Re-)Implement the start trading functionality	123
12.4 Implement the stop trading functionality	126
12.5 Implement the autostart trading functionality	127
12.6 Implement the shutdown trading functionality	129
<b>13 Abstract duplicated supervision code</b>	<b>136</b>
13.1 Objectives	136
13.2 Overview of requirements	136
13.3 Pseudo generalize Core.ServiceSupervisor module	137
13.4 Utilize pseudo generalized code inside the Naive DynamicSymbolSupervisor	140
13.5 Implement a truly generic Core.ServiceSupervisor	143
13.6 Remove boilerplate using use macro	151
13.7 Use the Core.ServiceSupervisor module inside the streamer application	157
<b>14 Store trade events and orders inside the database</b>	<b>161</b>
14.1 Objectives	161
14.2 Overview of requirements	161
14.3 Create a new data_warehouse application in the umbrella	162
14.4 Connect to the database using Ecto	162
14.5 Store trade events' data	164
14.6 Store orders' data	168
14.7 Implement supervision	174
<b>15 Backtest trading strategy</b>	<b>184</b>
15.1 Objectives	184
15.2 Overview of requirements	184
15.3 Implement the storing task	187
15.4 Test the backtesting	190

<b>16 End-to-end testing</b>	<b>193</b>
16.1 Objectives	193
16.2 Decide on the tested functionality	193
16.3 Implement basic test	195
16.4 Introduce environment based config files	199
16.5 Add convenience aliases	200
16.6 Cache initial seed data inside a file	202
16.7 Update seeding scripts to use the BinanceMock	205
16.8 Introduce the Core application	207
<b>17 Mox rocks</b>	<b>209</b>
17.1 Objectives	209
17.2 Introduction to mock based tests	209
17.3 Add the mox package	212
17.4 Investigate the <code>Naive.Trader</code> module	212
17.5 Implement a test of the <code>Naive.Trader</code> module	219
17.6 Define an alias to run unit tests	223
<b>18 Functional Elixir</b>	<b>226</b>
18.1 Objectives	226
18.2 The reasoning behind the functional approach	226
18.3 Simplifying by splitting	227
18.4 Abstracting the “pure” logic	231
18.5 Dealing with dirty code	241
18.6 Making dirty code testable	242
18.7 The power <code>with-in</code>	245
18.8 Do or not to do	247
18.9 Final thoughts	248
<b>19 Idiomatic OTP</b>	<b>249</b>
19.1 Objectives	249
19.2 The concept	249
19.3 Initial implementation	250
19.4 Idiomatic solution	256

<b>20 Idiomatic trading strategy</b>	<b>261</b>
20.1 Objectives . . . . .	261
20.2 Following the OHLC footsteps . . . . .	261
20.3 Simplifying the Naive supervision tree . . . . .	262
20.4 Supporting multiple positions . . . . .	264
20.5 Retrofitting the “shutdown” functionality . . . . .	274
20.6 Updating the Strategy to handle rebuys . . . . .	279
20.7 Fetching active positions . . . . .	281
20.8 Tidying up . . . . .	283
20.9 Final thoughts . . . . .	283