

CO565
Web Services

CW1

21611431

16th May 2018

Computing & Web Development
Buckinghamshire New University

Summary

A Research	2
1 Technology - Wider Context	3
2 Web Services Technology	4
3 REST as an alternative to Web Services	5
4 Examples of Web Services	6
 B Report	 7
5 Project Proposal	8
5.1 Project Name	8
5.2 Overview of the Project	8
5.3 External Web Services	9
5.4 Data Exposed By The Service	9
5.5 Database Schema	9
5.6 User Interface Ideas	9
6 Engineering Approach	16
6.1 Requirements specification	16
6.2 Designing	17
6.3 Implementation	17
6.4 Testing	17
7 Project Management	18
8 Development	19
8.1 Requirements elicitation	19
8.2 Design	20
8.3 Implementation	20
8.4 Testing	38
References	39

Part A

Research

Section 1

Technology - Wider Context

Section 2

Web Services Technology

Section 3

REST as an alternative to Web Services

Section 4

Examples of Web Services

Part B

Report

Section 5

Project Proposal

5.1 Project Name

The name for the project is CryptoStats

5.2 Overview of the Project

The aim of the project is to provide users with relevant information regarding cryptocurrency exchanges. By analysing the data provided by the external API on the values of the cryptocurrencies, the service will provide the user with information about the patterns in the value of the cryptocurrency and data such as the average number of days that the currency's value declines and increases in one go, if there is any resemblance in the fluctuation between the currency and other major currencies.

The goal with the project is to be able to provide users with enough information to help them make informed decisions when doing an exchange for a particular coin, and also gather information about what makes cryptocurrencies fluctuate in value. Rather than just showing the value of the coin, the service aims to provide way more valuable information by analysing the data in the currency's history, which allows the user to possess very important and essential information when trading coins.

The information processed will be presented to the users in two different ways: The first in a web site that will display the data in a user-friendly manner, built using ASP.NET Core 2 for the back-end and Angular and Bootstrap for the front-end. The second is by presenting the processed information to the user through a RESTful API with a JSON response, in order to facilitate the integration of the information with other applications or services.

For the submission of the project, I only intend to provide information for major coins

such as Bitcoin, Ethereum, Litecoin and Bitcoin Cash, since they are the most known and there is many information and exchanges that trade these coins.

5.3 External Web Services

In order to obtain information about the coins, I will be using the [GDAX exchange API](#).

5.4 Data Exposed By The Service

The service aims to expose the following data:

- Average fluctuation for the last day, week, month, and year
- Average time when in growth
- Average time when in decline
- Difference between highest value and latest value

5.5 Database Schema

5.6 User Interface Ideas

The main concept for the User Interface of the web page is for it to be displayed on a single page, and using Angular components allow the interface to be responsive and show the right information that the user is looking for without having to load the page again or another page.

The first sketches for the interface are the following:

The Web application will have some additional pages explaining the project and how it came about, as well as a form allowing users to leave messages and an API page explaining how to work with the API.

The following images are wireframes that were produced based on the sketches:

Figure 5.1: Database Schema

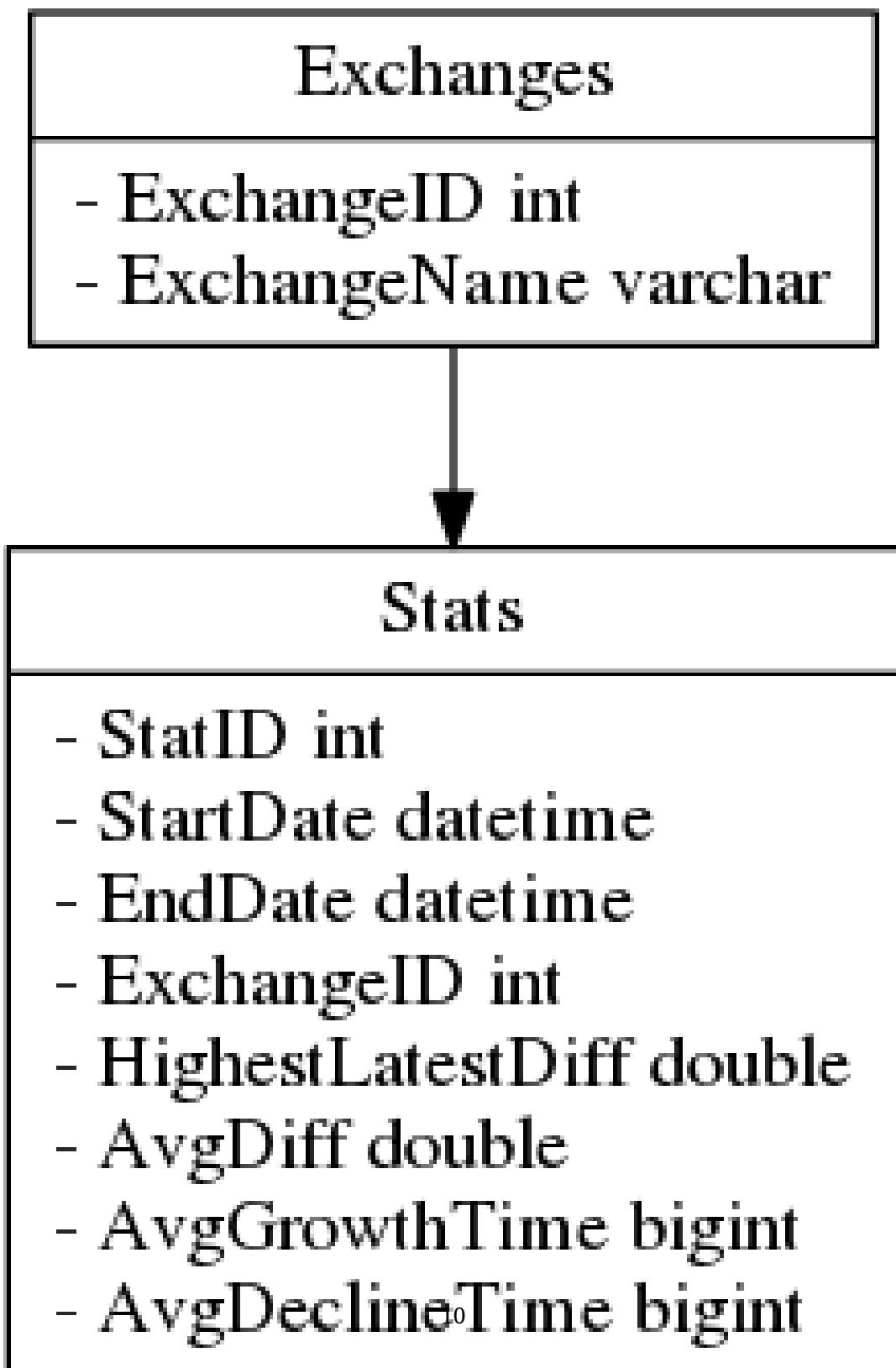


Figure 5.2: 1st Sketches

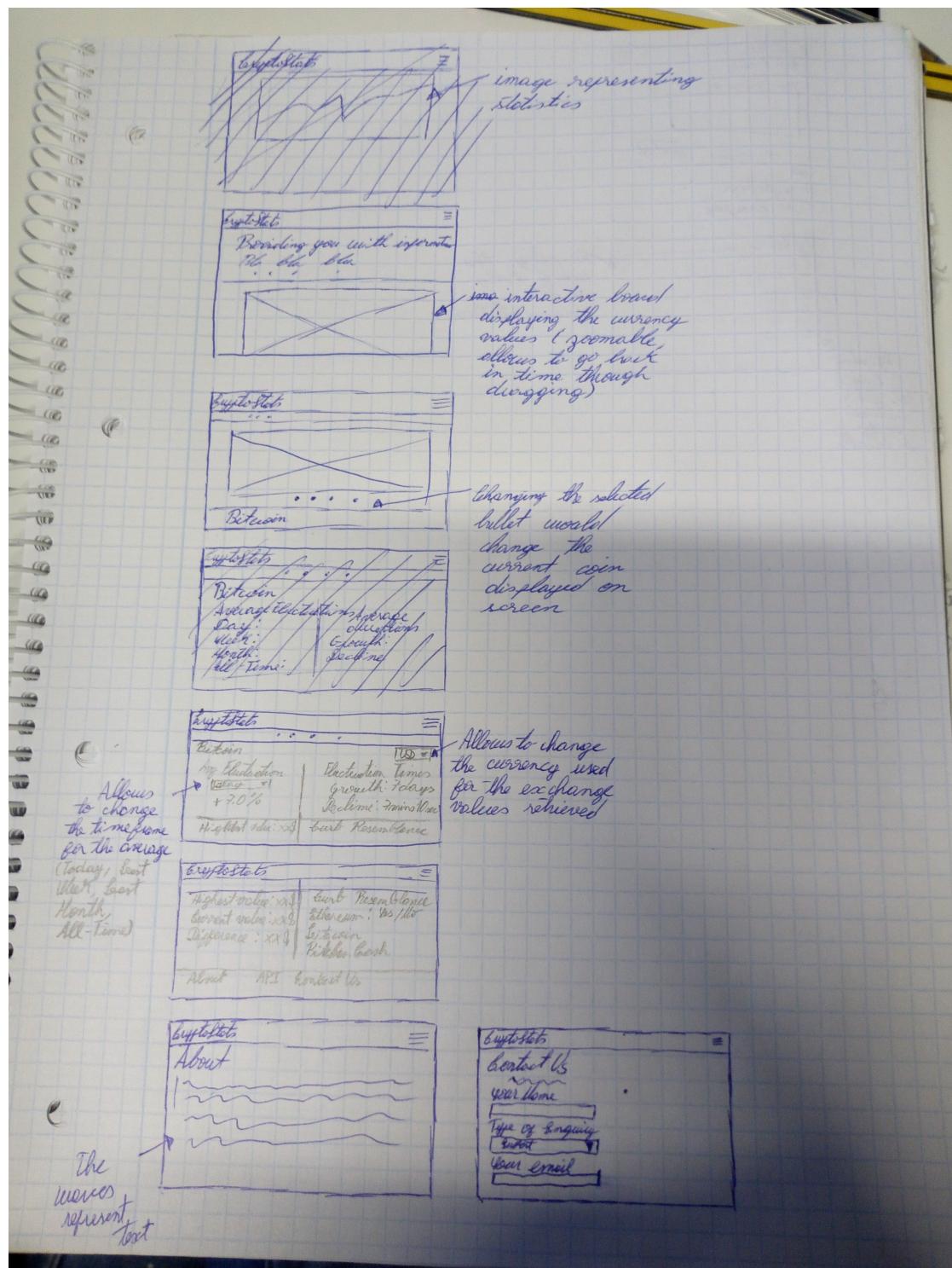


Figure 5.3: 2nd Sketches

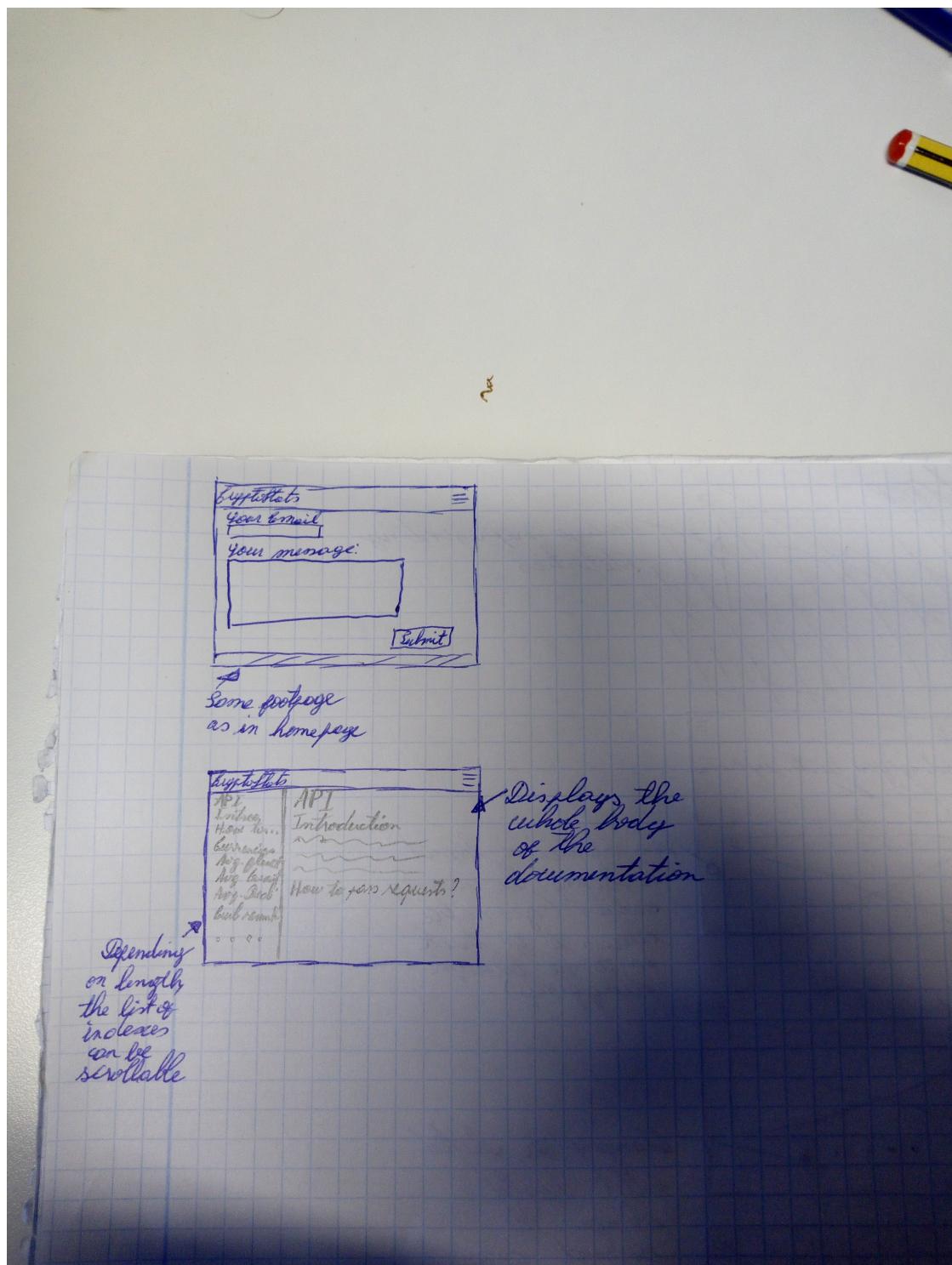


Figure 5.4: Homepage

The screenshot shows the top portion of the CryptoStats homepage. At the top left is the "CryptoStats" logo. To its right is a blue navigation bar with three horizontal lines indicating a menu. Below the logo, the text "Providing you with information you need" is displayed in blue. A detailed description of the platform follows, starting with "CryptoStats is a web platform with API functionality that provides you with all the most valuable and only necessary information to trade cryptocurrencies." This is followed by a long, mostly placeholder text block.

Market values

○ ○ ○ ○

Figure 5.5: Homepage Bottom

This screenshot shows the bottom section of the CryptoStats homepage, specifically for the Bitcoin section. The top part of the screenshot is identical to Figure 5.4. Below it, the "Bitcoin" section is visible. On the left, there's a "Average Fluctuation" section with a dropdown menu set to "Today" and a green " + 7%" indicator. On the right, there's a "Fluctuation Times" section showing "Growth: 7 days" and "Decline: 7 mins 10 secs". Below these are two more sections: "Value Differences" (listing "Highest value: xx\$", "Current value: xy\$", and "Difference: xz\$") and "Curb Resemblance" (listing "Ethereum: Yes", "Litecoin: No", and "Bitcoin Cash: Yes"). At the very bottom of the page, there's a blue footer bar with links for "About", "API", and "Contact Us".

Figure 5.6: About Page

The screenshot shows the 'About' section of the CryptoStats website. The page has a blue header bar with the 'CryptoStats' logo on the left and a three-line menu icon on the right. Below the header, the word 'About' is highlighted in bold. The main content area contains two blocks of placeholder text (Lorem ipsum and Maecenas) and a navigation bar at the bottom with links for 'About', 'API', and 'Contact Us'.

About

Placeholder text (Lorem ipsum):

Placeholder text (Maecenas):

About API Contact Us

Figure 5.7: Contact Us Page

The screenshot shows the 'Contact Us' section of the CryptoStats website. The page has a blue header bar with the 'CryptoStats' logo on the left and a three-line menu icon on the right. Below the header, the word 'Contact Us' is highlighted in bold. The main content area includes fields for 'Your Name' (text input), 'Your Email' (text input), 'Type of Enquiry' (dropdown menu with 'Suggestion' selected), 'Your Message' (text area), and a 'Submit' button. At the bottom, there is a navigation bar with links for 'About', 'API', and 'Contact Us'.

Contact Us

We would like you to be able to contact us if you have questions regarding the service we provide and to give us feedback. Please fill in the form below to leave us a message!

Your Name:

Your Email:

Type of Enquiry:

Suggestion

Your Message:

Submit

About API Contact Us

Figure 5.8: API Page

CryptoStats

API

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum in eros tellus. Quisque bibendum aliquet dolor id luctus. Ut ut metus eu ligula rhoncus tristique. Vestibulum sed urna lorem. Proin venenatis lectus a nunc aliquam, sit amet accumsan urna aliquam. Duis et rhoncus nibh. Donec posuere ante nec ante ultrices, in pretium arcu pellentesque.

Nunc ac ipsum rhoncus mauris viverra finibus euismod vel dolor. Curabitur quis purus condimentum, dapibus arcu sit amet, tempor metus. Vestibulum egestas, erat eget vehicula placerat, metus nisl aliquam ex, nec consequat ex arcu eu mi. Donec tellus libero, ultricies a sagittis eu, finibus quis nisl. Aenean vulputate, justo nec pellentesque laoreet, tortor dolor tristique tortor, nec iaculis turpis ipsum nec lacus. In vehicula magna id risus consectetur, ut pharetra lectus tincidunt. Quisque vestibulum in enim a blandit. Fusce et velit fermentum, aliquam tortor id, fringilla urna.

Phasellus accumsan lacinia aliquam. Praesent dapibus aliquam diam, at varius ante tristique ac. Ut aliquet sodales fermentum. Fusce blandit odio non lacus accumsan venenatis. Aliquam eu dui at leo rutrum elementum. Ut egestas eu nibh ac egestas. Curabitur sit amet luctus velit. Aliquam diam tortor, vehicula eget nulla at, pretium sagittis libero. Nulla ullamcorper convallis metus, in euismod metus dapibus ut. Morbi vel ligula vitae diam vehicula posuere ut id leo. Integer metus dolor, pellentesque quis justo non, venenatis tincidunt sapien. Suspendisse potenti.

About API Contact Us

Section 6

Engineering Approach

For this project, I worked in an agile and iterative manner, based on the SCRUM methodology.

6.1 Requirements specification

When specifying the requirements for the project, I started by choosing what I wanted the project to offer to the users initially. I started by thinking of the core functionality that I wanted for the project and what data I wanted my API to expose to the user. After deciding on the data that would be exposed, I started specifying the requirements that would lead to the exposure of the data, by doing a list of the core functionality that the API would have:

- Display all the exchanges and statistics
- Display exchanges and statistics by id
- Display statistics by start date
- Display statistics by end date
- Display statistics by start and end date
- Display statistics related to a single exchange
- Display statistics related to a single exchange by id
- Display statistics related to a single exchange by start date
- Display statistics related to a single exchange by end date
- Display statistics related to a single exchange by start and end date

Most of this functionality would allow the user to retrieve different series of data by the criteria that they choose (id and or range of dates) and the type of data that they are looking for (exchange or statistics).

6.2 Designing

I started designing the implementation of the features on paper, by writing down the parts of the algorithms that would implement the features that were the most difficult to implement.

In order to implement the features, I chose to go with a Model-Controller approach, where the data would be initially stored and retrieved through the models and the business logic and presentation of the data would occur in the controller. The business logic that would allow the API to seed the database and add data to it would also work in the models.

6.3 Implementation

For the implementation, I started by implementing the models and the database context that would allow the connection to the database, storage and retrieval of the data.

After that, I implemented the controllers and added the functionality that would display the JSON results to the user.

After all of that was done, I moved on to implement the features that would retrieve data from the input API and process it. For that, I studied and modified a library client that was available for the API I was using, so that only the functions that I needed for my project would be available, and created new namespaces for those functionalities.

After implementing the seeding and storage of new data functinoalities, I started implementing the front-end of my project. Based on my previous UI design and wireframes, I used the planned technologies to build a front-end that would resemble the original design as much as possible.

6.4 Testing

This project was throughly tested as it was being built. As soon as a new feature would be implemented, the project would be tested to make sure that everything was working as expected and that everything was also behaving and looking in an expected way.

Section 7

Project Management

In order to manage the project, the code as well as the project tasks were stored in a GitHub repository.

That choice was made because I already had previous experience storing my code and repositories in GitHub but also because the way GitHub does project management was very suitable for the approach that I was using to develop the project.

Using SCRUM terms, GitHub facilitates the planning of sprints and tasks by its use of issues and milestones, but also because it makes it very easy to keep track of how the project is evolving through the use of Kanban boards. It is also possible to assign the different tasks to different sprints and to users, so that everyone involved in the project knows where the project is and what they need to do.

In this particular case, I developed the whole project myself, but I still found it very useful to use the features provided by GitHub, especially because it allows me to comment on the tasks as I am getting them done, allowing me to have a log of the progress that I have made over time, as well as register the difficulties that I had to face and how I overcame them.

The project code and tasks can be found [here](#).

Section 8

Development

8.1 Requirements elicitation

8.1.1 API

- Display all stored exchanges and statistics
- Display exchanges with a specific id
- Display statistics with a specific id
- Display the statistics starting at a specific date
- Display the statistics ending at a specific date
- Display the statistics with a specific start and end dates
- Fetch all the statistics related to a specific exchange
- Fetch the statistic related to a particular exchange with a specific id
- Fetch all the statistics related to a specific exchange with a specific start date
- Fetch all the statistics related to a specific exchange with a particular end date
- Fetch a statistic related to a particular exchange with specific start and end dates

8.1.2 Front-End

- Display for a particular exchange:
 - Average fluctuation
 - Average time in growth

- Average time in decline
- Difference between highest and latest values
- Allow display of information for a period of the last day, week, month and year.

8.2 Design

8.3 Implementation

Here is the code for the API:

8.3.1 Main Program

Figure 8.1: Program.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Threading.Tasks;
6  using Microsoft.AspNetCore;
7  using Microsoft.AspNetCore.Hosting;
8  using Microsoft.Extensions.DependencyInjection;
9  using Microsoft.Extensions.Configuration;
10 using Microsoft.Extensions.Logging;
11 using CryptoStats.Models;

12 namespace CryptoStats
13 {
14     public class Program
15     {
16         public static void Main(string[] args)
17         {
18             var host = BuildWebHost(args);

19             using (var scope = host.Services.CreateScope())
20             {
21                 var services = scope.ServiceProvider;

22                 try
23                 {
24                     SeedData.Initialize(services);

```

```

25     }
26     catch (Exception ex)
27     {
28         var logger = services.GetRequiredService<ILogger<Program>>();
29         logger.LogError(ex, "An error occurred seeding the DB.");
30     }
31 }

32     host.Run();
33 }

34 public static IWebHost BuildWebHost(string[] args) =>
35     WebHost.CreateDefaultBuilder(args)
36         .UseStartup<Startup>()
37         .Build();
38     }
39 }
```

Figure 8.2: Startup.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Builder;
6  using Microsoft.AspNetCore.Hosting;
7  using Microsoft.AspNetCore.HttpsPolicy;
8  using Microsoft.AspNetCore.Mvc;
9  using Microsoft.Extensions.Configuration;
10 using Microsoft.Extensions.DependencyInjection;
11 using Microsoft.Extensions.Logging;
12 using Microsoft.Extensions.Options;
13 using Microsoft.EntityFrameworkCore;
14 using Microsoft.EntityFrameworkCore.Sqlite;
15 using CryptoStats.Models;

16 namespace CryptoStats
17 {
18     public class Startup
19     {
20         public Startup(IConfiguration configuration)
21         {
22             Configuration = configuration;
23         }

```

```

24     public IConfiguration Configuration { get; }
25
26     // This method gets called by the runtime. Use this method to add services to the container.
27     public void ConfigureServices(IServiceCollection services)
28     {
29         services.AddDbContext<CryptoContext>(opt => opt.UseSqlite("Data Source=app.db"));
30         services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
31     }
32
33     // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
34     public void Configure(IApplicationBuilder app, IHostingEnvironment env)
35     {
36         if (env.IsDevelopment())
37         {
38             app.UseDeveloperExceptionPage();
39         }
40         else
41         {
42             app.UseHsts();
43         }
44
45         app.UseHttpsRedirection();
46         app.UseMvc();
47     }
48 }

```

Figure 8.3: appsettings.json

```

1  {
2      "ConnectionStrings": {
3          "DefaultConnection": "DataSource=app.db"
4      },
5      "Logging": {
6          "LogLevel": {
7              "Default": "Warning"
8          }
9      },
10     "AllowedHosts": "*"
11 }

```

8.3.2 Models

Figure 8.4: Exchange.cs

```

1  using System.Collections.Generic;
2
3  namespace CryptoStats.Models
4  {
5      public class Exchange
6      {
7          public int ExchangeId{get;set;}
8          public string Name{get;set;}
9
10         //Setting list for foreign key
11         public List<Stat> Stats{get;set;}
12     }
13 }
```

Figure 8.5: Stat.cs

```

1  using System;
2
3  namespace CryptoStats.Models
4  {
5      public class Stat
6      {
7          public int StatId{get;set;}
8          public DateTime startDate {get;set;}
9          public DateTime endDate{get;set;}
10         public decimal HighestLatestDiff{get;set;}
11         public decimal avgDiff {get;set;}
12         public TimeSpan avgGrowthTime {get;set;}
13         public TimeSpan avgDeclineTime {get;set;}
14
15         //Setting variables for foreign key
16         public int ExchangeId{get;set;}
17         public Exchange Exchange {get;set;}
18     }
19 }
```

Figure 8.6: SeedData.cs

```

1  using Microsoft.EntityFrameworkCore;
2  using Microsoft.Extensions.DependencyInjection;
3  using System;
4  using System.Linq;
5  using System.Collections.Generic;
6  using CryptoStats.GDAX;
7  using CryptoStats.GDAX.Services.Products.Models;
```

```

8  using CryptoStats.GDAX.Services.Products.Types;
9  using CryptoStats.GDAX.Shared.Types;

10 namespace CryptoStats.Models
11 {
12     public static class SeedData
13     {
14         private static GDAXClient client = new GDAXClient();
15         public static void Initialize(IServiceProvider serviceProvider)
16         {
17             using(var context = new CryptoContext(serviceProvider.GetRequiredService<ICryptoContext>()))
18             {
19                 if(context.Exchanges.Any() && context.Stats.Any())
20                 {
21                     return;
22                 }
23
24                 if(!context.Exchanges.Any())
25                 {
26                     List<Exchange> exchanges = new List<Exchange>();
27                     var products = client.ProductService.GetAllProductsAsync().Result;
28                     foreach(Product prod in products)
29                     {
30                         exchanges.Add(new Exchange(){Name = prod.Id.ToString()});
31                     }
32                     context.Exchanges.AddRange(exchanges);
33                     context.SaveChanges();
34                 }
35
36                 if(!context.Stats.Any())
37                 {
38                     foreach(Exchange e in context.Exchanges)
39                     {
40                         seedStat(e.ExchangeId, DateTime.Now.AddDays(-7), DateTime.Now);
41                     }
42                 }
43
44             public static void seedStat(int exchangeID, DateTime startDate, DateTime endDate)
45             {
46                 using(context)
47                 {
48                     foreach(Product p in client.ProductService.GetAllProductsAsync().Result)
49                     {
50                         if(p.Id == exchangeID)
51                         {
52                             var stats = context.Stats.Where(s => s.ExchangeId == exchangeID).ToList();
53                             if(stats.Count == 0)
54                             {
55                                 var stat = new Stat();
56                                 stat.ExchangeId = exchangeID;
57                                 stat.Date = startDate;
58                                 stat.High = p.High;
59                                 stat.Low = p.Low;
60                                 stat.Open = p.Open;
61                                 stat.Close = p.Close;
62                                 context.Add(stat);
63                             }
64                             else
65                             {
66                                 var existingStat = stats[0];
67                                 existingStat.High = p.High;
68                                 existingStat.Low = p.Low;
69                                 existingStat.Open = p.Open;
70                                 existingStat.Close = p.Close;
71                             }
72                         }
73                     }
74                 }
75             }
76         }
77     }
78 }

```

```

48
49     {
50         if(p.Id.ToString().Equals(context.Exchanges.Find(exchangeID).Name))
51         {
52             TimeSpan t = endDate.Subtract(startDate);
53             int days = (int)t.TotalDays;
54             for(int i = 0; i < days; i++)
55             {
56                 var candles = client.ProductsService.GetHistoricR
57                 Stat adder = new Stat(){
58                     startDate = candles.First().Time,
59                     endDate = candles.Last().Time
60                 };
61
62                 //calculate highestLatestDiff
63                 decimal high = 0.00M;
64                 foreach(Candle c in candles)
65                 {
66                     if(c.High > high)
67                     {
68                         high = (decimal)c.High;
69                     }
70                 }
71                 adder.HighestLatestDiff = (decimal)candles.Last()
72
73                 //calculate avgDiff
74                 //Use difference between open and close of candle
75                 decimal candlesDiff = 0.00M;
76                 foreach(Candle c in candles)
77                 {
78                     candlesDiff += (decimal)(c.Close - c.Open);
79                 }
80                 adder.avgDiff = candlesDiff / candles.Count();
81
82                 //calculate average growth and decline times
83                 //store candles in a temporary list to retrieve t
84                 //growth and decline times are stored in a list o
85                 //average is calculated afterwards
86                 List<TimeSpan> growthTimes = new List<TimeSpan>()
87                 List<TimeSpan> declineTimes = new List<TimeSpan>()
88                 bool inGrowth = false;
List<Candle> growthCandles = new List<Candle>();

foreach(Candle c in candles)
{
    if(c.Close > c.Open && inGrowth == false)

```

```

89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131

    {
        inGrowth = true;
        declineTimes.Add(growthCandles.Last().Time);
        growthCandles.Clear();
        growthCandles.Add(c);
    }
    else if(c.Close > c.Open && inGrowth == true)
    {
        growthCandles.Add(c);
    }
    else if(c.Close < c.Open && inGrowth == true)
    {
        inGrowth = false;
        growthTimes.Add(growthCandles.Last().Time);
        growthCandles.Clear();
        growthCandles.Add(c);
    }
    else if(c.Close < c.Open && inGrowth == false)
    {
        growthCandles.Add(c);
    }
}
if(inGrowth)
{
    growthTimes.Add(growthCandles.Last().Time.Subtract(
}
else
{
    declineTimes.Add(growthCandles.Last().Time.Subtract(
}

double totalTimes = 0;
foreach(TimeSpan time in growthTimes)
{
    totalTimes += t.TotalMilliseconds;
}
adder.avgGrowthTime = TimeSpan.FromMilliseconds(totalTimes);

totalTimes = 0;
foreach(TimeSpan time in declineTimes)
{
    totalTimes += t.TotalMilliseconds;
}
adder.avgDeclineTime = TimeSpan.FromMilliseconds(totalTimes);
context.Stats.Add(adder);

```

```

132                         startDate.AddDays(1);
133                     }

134                     startDate.AddDays(-days);
135                     for(int i = 0; i < days; i++)
136                     {
137                         var candles = client.ProductsService.GetHistoricR
138                         Stat adder = new Stat(){
139                             startDate = candles.First().Time,
140                             endDate = candles.Last().Time
141                         };

142                         //calculate highestLatestDiff
143                         decimal high = 0.00M;
144                         foreach(Candle c in candles)
145                         {
146                             if(c.High > high)
147                             {
148                                 high = (decimal)c.High;
149                             }
150                         }
151                         adder.HighestLatestDiff = (decimal)candles.Last();

152                         //calculate avgDiff
153                         //Use difference between open and close of candle
154                         decimal candlesDiff = 0.00M;
155                         foreach(Candle c in candles)
156                         {
157                             candlesDiff += (decimal)(c.Close - c.Open);
158                         }
159                         adder.avgDiff = candlesDiff / candles.Count();

160                         //calculate average growth and decline times
161                         //store candles in a temporary list to retrieve t
162                         //growth and decline times are stored in a list o
163                         //average is calculated afterwards
164                         List<TimeSpan> growthTimes = new List<TimeSpan>()
165                         List<TimeSpan> declineTimes = new List<TimeSpan>()
166                         bool inGrowth = false;
167                         List<Candle> growthCandles = new List<Candle>();

168                         foreach(Candle c in candles)
169                         {
170                             if(c.Close > c.Open && inGrowth == false)
171                             {

```

```

172                     inGrowth = true;
173                     declineTimes.Add(growthCandles.Last().Time);
174                     growthCandles.Clear();
175                     growthCandles.Add(c);
176                 }
177             else if(c.Close > c.Open && inGrowth == true)
178             {
179                 growthCandles.Add(c);
180             }
181             else if(c.Close < c.Open && inGrowth == true)
182             {
183                 inGrowth = false;
184                 growthTimes.Add(growthCandles.Last().Time);
185                 growthCandles.Clear();
186                 growthCandles.Add(c);
187             }
188             else if(c.Close < c.Open && inGrowth == false)
189             {
190                 growthCandles.Add(c);
191             }
192         }
193         if(inGrowth)
194         {
195             growthTimes.Add(growthCandles.Last().Time.Subtract(
196             growthCandles.First().Time));
197         }
198         else
199         {
200             declineTimes.Add(growthCandles.Last().Time.Subtract(
201             growthCandles.First().Time));
202         }

203         double totalTimes = 0;
204         foreach(TimeSpan time in growthTimes)
205         {
206             totalTimes += t.TotalMilliseconds;
207         }
208         adder.avgGrowthTime = TimeSpan.FromMilliseconds(totalTimes);

209         totalTimes = 0;
210         foreach(TimeSpan time in declineTimes)
211         {
212             totalTimes += t.TotalMilliseconds;
213         }
214         adder.avgDeclineTime = TimeSpan.FromMilliseconds(totalTimes);
215         context.Stats.Add(add);
216         endDate.AddDays(-1);

```

```

215                     }
216                     context.SaveChanges();
217                 }
218             }
219         }
220     }
221 }
222 }
```

Figure 8.7: DbContext.cs

```

1  using Microsoft.EntityFrameworkCore;
2
3  namespace CryptoStats.Models
4  {
5      public class CryptoContext: DbContext
6      {
7          public CryptoContext(DbContextOptions<CryptoContext> options) : base(options)
8          {
9          }
10
11         public DbSet<Exchange> Exchanges{get;set;}
12         public DbSet<Stat> Stats{get;set;}
13     }
14 }
```

8.3.3 Controllers

Figure 8.8: ExchangesController.cs

```

1  using Microsoft.AspNetCore.Mvc;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System;
5  using CryptoStats.Models;
6
7  namespace CryptoStats.Controllers
8  {
9      [Route("api/[controller]")]
10     [ApiController]
11     public class ExchangesController : ControllerBase
12     {
13         private readonly CryptoContext _context;
14
15         public ExchangesController(CryptoContext context)
```

```

14     {
15         _context = context;
16     }
17
18     [HttpGet]
19     public ActionResult<List<Exchange>> GetAll()
20     {
21         return _context.Exchanges.ToList();
22     }
23
24     [HttpGet("{id}", Name="GetExchange")]
25     public ActionResult<Exchange> GetById(int id)
26     {
27         var item = _context.Exchanges.Find(id);
28         if(item == null)
29         {
30             return NotFound();
31         }
32         return item;
33     }

```

Figure 8.9: ExchangeStatsController.cs

```

1  using Microsoft.AspNetCore.Mvc;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System;
5  using CryptoStats.Models;
6
7  namespace CryptoStats.Controllers
8  {
9      [Route("api/")]
10     [ApiController]
11     public class ExchangeStatsController : ControllerBase
12     {
13         private readonly CryptoContext _context;
14
15         public ExchangeStatsController(CryptoContext context)
16         {
17             _context = context;
18         }
19
20         [HttpGet("exchanges/{id}/stats")]
21     }
22
23     [HttpGet("exchanges")]
24     public ActionResult<List<Exchange>> GetAll()
25     {
26         return _context.Exchanges.ToList();
27     }
28
29     [HttpGet("exchanges/{id}")]
30     public ActionResult<Exchange> GetById(int id)
31     {
32         var item = _context.Exchanges.Find(id);
33         if(item == null)
34         {
35             return NotFound();
36         }
37         return item;
38     }
39
40     [HttpPost("exchanges")]
41     public void Create([FromBody] Exchange item)
42     {
43         _context.Exchanges.Add(item);
44         _context.SaveChanges();
45     }
46
47     [HttpPut("exchanges/{id}")]
48     public void Update([FromBody] Exchange item)
49     {
50         var existingItem = _context.Exchanges.Find(item.Id);
51         if(existingItem != null)
52         {
53             _context.Exchanges.Update(item);
54             _context.SaveChanges();
55         }
56     }
57
58     [HttpDelete("exchanges/{id}")]
59     public void Delete(int id)
60     {
61         var item = _context.Exchanges.Find(id);
62         if(item != null)
63         {
64             _context.Exchanges.Remove(item);
65             _context.SaveChanges();
66         }
67     }
68
69     [HttpGet("exchanges/{id}/stats")]
70     public ActionResult<ExchangeStat> GetStats(int id)
71     {
72         var exchange = _context.Exchanges.Find(id);
73         if(exchange == null)
74         {
75             return NotFound();
76         }
77         var stats = new ExchangeStat
78         {
79             ExchangeId = exchange.Id,
80             Volume = exchange.Volume,
81             Price = exchange.Price,
82             LastUpdated = exchange.LastUpdated
83         };
84         return stats;
85     }
86
87     [HttpGet("exchanges")]
88     public ActionResult<List<ExchangeStat>> GetAllStats()
89     {
90         var exchanges = _context.Exchanges.ToList();
91         var stats = exchanges.Select(exchange => new ExchangeStat
92         {
93             ExchangeId = exchange.Id,
94             Volume = exchange.Volume,
95             Price = exchange.Price,
96             LastUpdated = exchange.LastUpdated
97         }).ToList();
98         return stats;
99     }
100 }

```

```

18     public ActionResult<List<Stat>> GetAll(int id)
19     {
20         var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
21         if(exchange == null)
22         {
23             return NotFound();
24         }
25         else
26         {
27             return exchange.Stats;
28         }
29     }
30
31     [HttpGet("exchanges/{id}/stats/id/{statId}")]
32     public ActionResult<Stat> GetByID(int id, int statId)
33     {
34         var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
35         if(exchange == null)
36         {
37             return NotFound();
38         }
39         else
40         {
41             return exchange.Stats.Where(s => s.StatId == statId).First();
42         }
43
44     [HttpGet("exchanges/{id}/stats/startDate/{startDate}")]
45     public ActionResult<List<Stat>> GetByStartDate(int id, DateTime startDate)
46     {
47         var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
48         if(exchange == null)
49         {
50             return NotFound();
51         }
52         else
53         {
54             return exchange.Stats.Where(s => s.startDate.Equals(startDate)).ToList();
55         }
56
57     [HttpGet("exchanges/{id}/stats/endDate/{endDate}")]
58     public ActionResult<List<Stat>> GetByEndDate(int id, DateTime endDate)
59     {
60         var exchange = _context.Exchanges.Where(e => e.ExchangeId== id).First();

```

```

60         if(exchange == null)
61     {
62         return NotFound();
63     }
64     else
65     {
66         return exchange.Stats.Where(s => s.endDate.Equals(endDate)).ToList();
67     }
68 }

69 [HttpGet("exchanges/{id}/stats/dates/{startDate}/{endDate}")]
70 public ActionResult<Stat> GetByDates(int id, DateTime startDate, DateTime endDate)
71 {
72     var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
73     if(exchange == null)
74     {
75         return NotFound();
76     }
77     else
78     {
79         return exchange.Stats.Where(s => s.startDate.Equals(startDate) && s.endDate.Equals(endDate));
80     }
81 }
82 }
83 }
```

Figure 8.10: StatsController.cs

```

1  using Microsoft.AspNetCore.Mvc;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System;
5  using CryptoStats.Models;

6  namespace CryptoStats.Controllers
7  {
8      [Route("api/[controller]")]
9      [ApiController]
10     public class StatsController : ControllerBase
11     {
12         private readonly CryptoContext _context;

13         public StatsController(CryptoContext context)
14         {
15             _context = context;
```

```

16 }
17
18 [HttpGet]
19 public ActionResult<List<Stat>> GetAll()
20 {
21     return _context.Stats.ToList();
22 }
23
24 [HttpGet("/id/{id}")]
25 public ActionResult<Stat> GetById(int id)
26 {
27     var item = _context.Stats.Find(id);
28     if(item == null)
29     {
30         return NotFound();
31     }
32     return item;
33 }
34
35 [HttpGet("/startDate/{startDate}")]
36 public ActionResult<List<Stat>> GetByStartDate(DateTime startDate)
37 {
38     var items = _context.Stats.Where(s => s.startDate.Equals(startDate)).ToList();
39     if(items == null)
40     {
41         return NotFound();
42     }
43     return items;
44 }
45
46 [HttpGet("/endDate/{endDate}")]
47 public ActionResult<List<Stat>> GetByEndDate(DateTime endDate)
48 {
49     var items = _context.Stats.Where(s => s.endDate.Equals(endDate)).ToList();
50     if(items == null)
51     {
52         return NotFound();
53     }
54     return items;
55 }
56
57 [HttpGet("/dates/{startDate}/{endDate}")]
58 public ActionResult<Stat> GetByDates(DateTime startDate, DateTime endDate)
59 {
60     var item = _context.Stats.Where(s => s.startDate.Equals(startDate) && s.e

```

```

56             if(item == null)
57             {
58                 return NotFound();
59             }
60             return item;
61         }
62     }
63 }
```

8.3.4 GDAX Client

Figure 8.11: GDAXClient.cs

```

1  using System;
2  using CryptoStats.GDAX.Network.HttpClient;
3  using CryptoStats.GDAX.Network.HttpRequest;
4  using CryptoStats.GDAX.Services.Products;
5  using CryptoStats.GDAX.Shared.Utilities.Clock;
6  using CryptoStats.GDAX.Shared.Utilities.Questions;

7  namespace CryptoStats.GDAX
8  {
9      public class GDAXClient
10     {
11         public GDAXClient(bool sandBox = false) : this(new HttpClient(), sandBox){}
12
13         public GDAXClient(IHttpClient httpClient, bool sandBox = false)
14         {
15             var clock = new Clock();
16             var httpRequestMessageService = new HttpRequestMessageService(sandBox);
17             var queryBuilder = new QueryBuilder();
18
19             ProductsService = new ProductsService(httpClient, httpRequestMessageService);
20         }
21     }
```

Figure 8.12: ProductsService.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Globalization;
4  using System.Linq;
```

```

5  using System.Net.Http;
6  using System.Threading.Tasks;
7  using CryptoStats.GDAX.Services;
8  using CryptoStats.GDAX.Network.HttpClient;
9  using CryptoStats.GDAX.Network.HttpRequest;
10 using CryptoStats.GDAX.Services.Products.Models;
11 using CryptoStats.GDAX.Services.Products.Models.Responses;
12 using CryptoStats.GDAX.Services.Products.Types;
13 using CryptoStats.GDAX.Shared.Types;
14 using CryptoStats.GDAX.Shared.Utilities.Extensions;
15 using CryptoStats.GDAX.Shared.Utilities.Queries;

16 namespace CryptoStats.GDAX.Services.Products
17 {
18     public class ProductsService: AbstractService
19     {
20         private readonly IQueryBuilder queryBuilder;

21         public ProductsService(IHttpClient httpClient, IHttpRequestMessageService http
22         {
23             this.queryBuilder = queryBuilder;
24         }

25         public async Task<List<Product>> GetAllProductsAsync()
26         {
27             return await SendServiceCall<List<Product>>(HttpMethod.Get, "/products");
28         }

29         public async Task<ProductsOrderBookResponse> GetProductOrderBookAsync(
30                         ProductType productId,
31                         ProductLevel productLevel = ProductLevel.One)
32         {
33             var productsOrderBookJsonResponse = await SendServiceCall<ProductsOrderBo
34             var productOrderBookResponse = ConvertProductOrderBookResponse(productsO

35             return productOrderBookResponse;
36         }

37         public async Task<ProductTicker> GetProductTickerAsync(ProductType productId)
38         {
39             return await SendServiceCall<ProductTicker>(HttpMethod.Get, $"{"/products/{"}
40         }

41         public async Task<ProductStats> GetProductStatsAsync(ProductType productId)
42         {

```

```

43     return await SendServiceCall<ProductStats>(HttpMethod.Get, $"/{productType}/{productId}/stats");
44 }

45 public async Task<IList<IList<ProductTrade>>> GetTradesAsync(
46     ProductType productId,
47     int limit = 100,
48     int numberOfPages = 0)
49 {
50     var httpResponseMessage = await SendHttpRequestMessagePagedAsync<ProductTrade>(
51         HttpMethod.Get, $"/{productType}/{productId}/trades", limit, numberOfPages);
52 
53     return httpResponseMessage;
54 }

55 public async Task<IList<Candle>> GetHistoricRatesAsync(
56     ProductType productPair,
57     DateTime start,
58     DateTime end,
59     CandleGranularity granularity)
60 {
61     const int maxPeriods = 300;
62 
63     var candleList = new List<Candle>();
64 
65     DateTime? batchEnd = end;
66     DateTime batchStart;
67 
68     do
69     {
70         if (batchEnd == null) {
71             break;
72         }
73 
74         batchStart = batchEnd.Value.AddSeconds(-maxBatchPeriod);
75         if (batchStart < start) batchStart = start;
76 
77         candleList.AddRange(await GetHistoricRatesAsync(productPair, batchStart));
78 
79         batchEnd = candleList.LastOrDefault()?.Time;
80 
81     } while (batchStart > start);
82 
83     return candleList;
84 }

85 }
```

```

76     private async Task<IList<Candle>> GetHistoricRatesAsync(
77         ProductType productId,
78         DateTime start,
79         DateTime end,
80         int granularity)
81     {
82         var isoStart = start.ToString("s");
83         var isoEnd = end.ToString("s");

84         var queryString = queryBuilder.BuildQuery(
85             new KeyValuePair<string, string>("start", isoStart),
86             new KeyValuePair<string, string>("end", isoEnd),
87             new KeyValuePair<string, string>("granularity", granularity.ToString()));

88         return await SendServiceCall<IList<Candle>>(HttpMethod.Get, $"#/products/{productId}/candles?{queryString}");
89     }

90     private ProductsOrderBookResponse ConvertProductOrderBookResponse(
91         ProductsOrderBookJsonResponse productsOrderBookJsonResponse,
92         ProductLevel productLevel)
93     {
94         var askList = productsOrderBookJsonResponse.Asks.Select(product => product);
95         {
96             OrderId = productLevel == ProductLevel.Three
97                 ? new Guid(askArray[2])
98                 : (Guid?)null,
99             NumberOfOrders = productLevel == ProductLevel.Three
100                ? (decimal?)null
101                : Convert.ToDecimal(askArray[2], CultureInfo.InvariantCulture);
102         }).ToArray();

103         var bidList = productsOrderBookJsonResponse.Bids.Select(product => product);
104         {
105             OrderId = productLevel == ProductLevel.Three
106                 ? new Guid(bidArray[2])
107                 : (Guid?)null,
108             NumberOfOrders = productLevel == ProductLevel.Three
109                 ? (decimal?)null
110                 : Convert.ToDecimal(bidArray[2], CultureInfo.InvariantCulture);
111         });

112         var productOrderBookResponse = new ProductsOrderBookResponse(productsOrderBookJsonResponse);
113         return productOrderBookResponse;
114     }

```

115 }
116 }

8.4 Testing

References