CO565
Web Services

CW1

21611431

16th May 2018

Computing & Web Development
Buckinghamshire New University

Summary

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 cryptocurrencys, 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 betweeen 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 cryptocurrenncies 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 posess 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

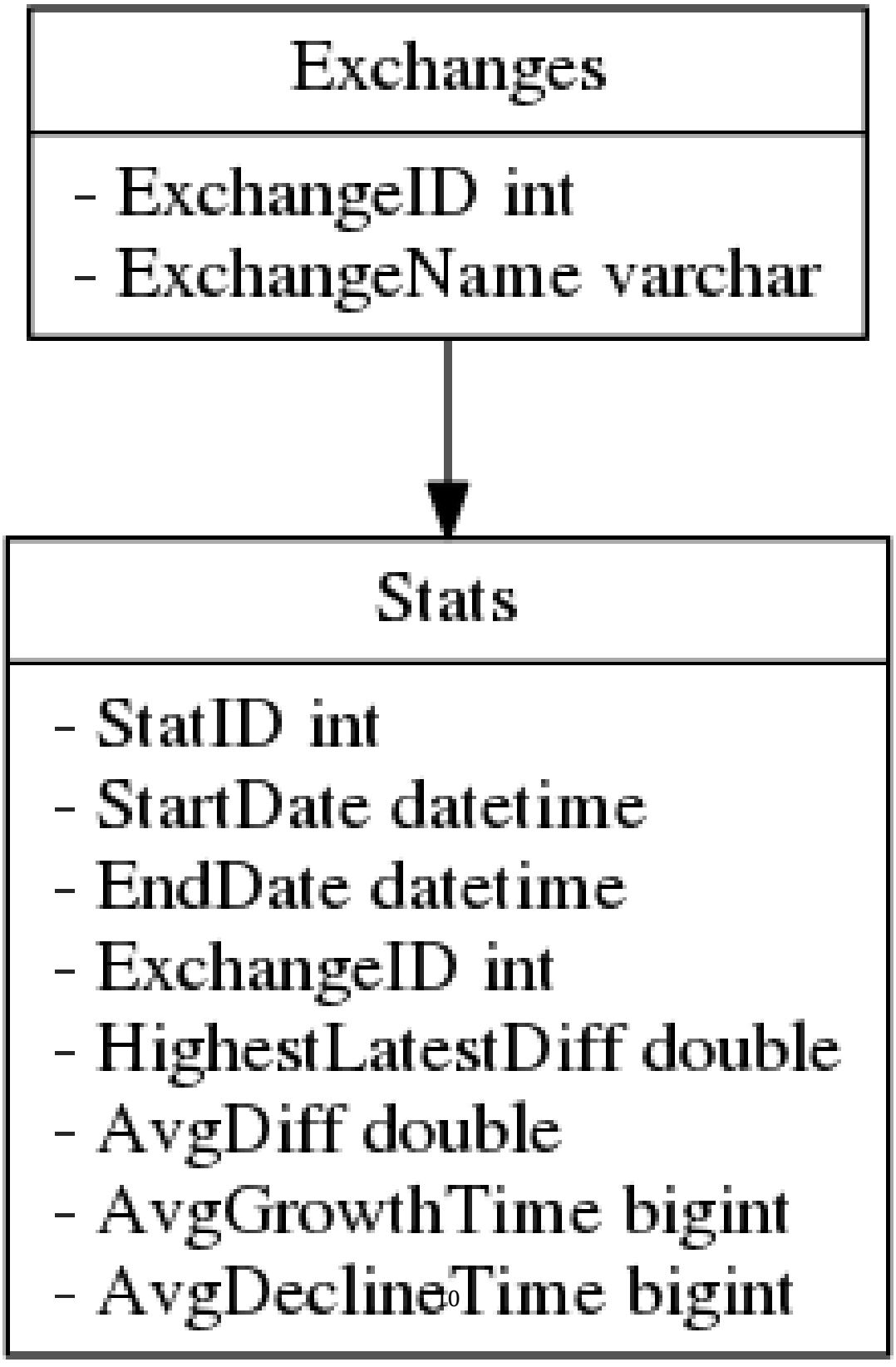

Figure 5.2: 1st Sketches

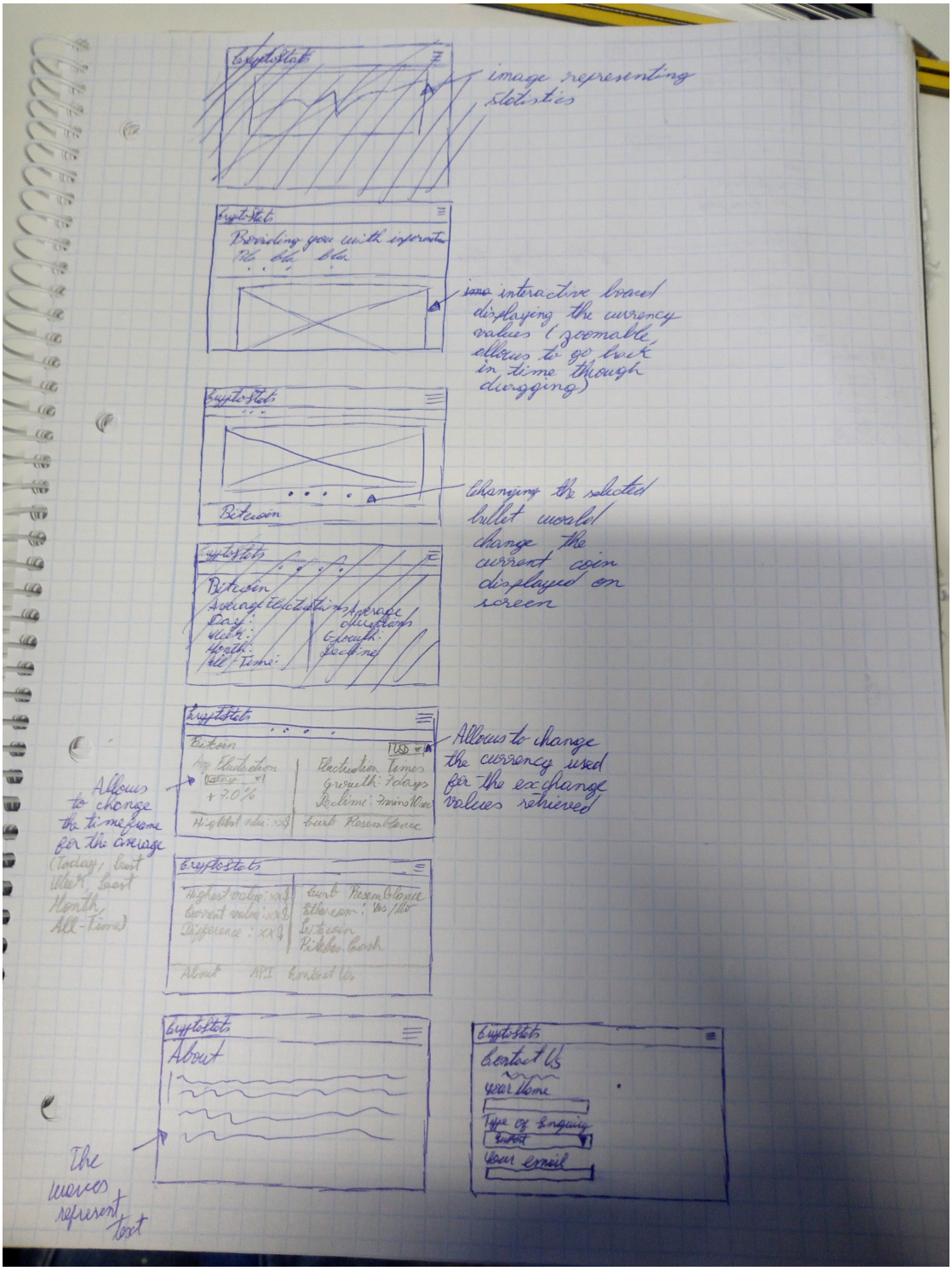

Figure 5.3: 2nd Sketches

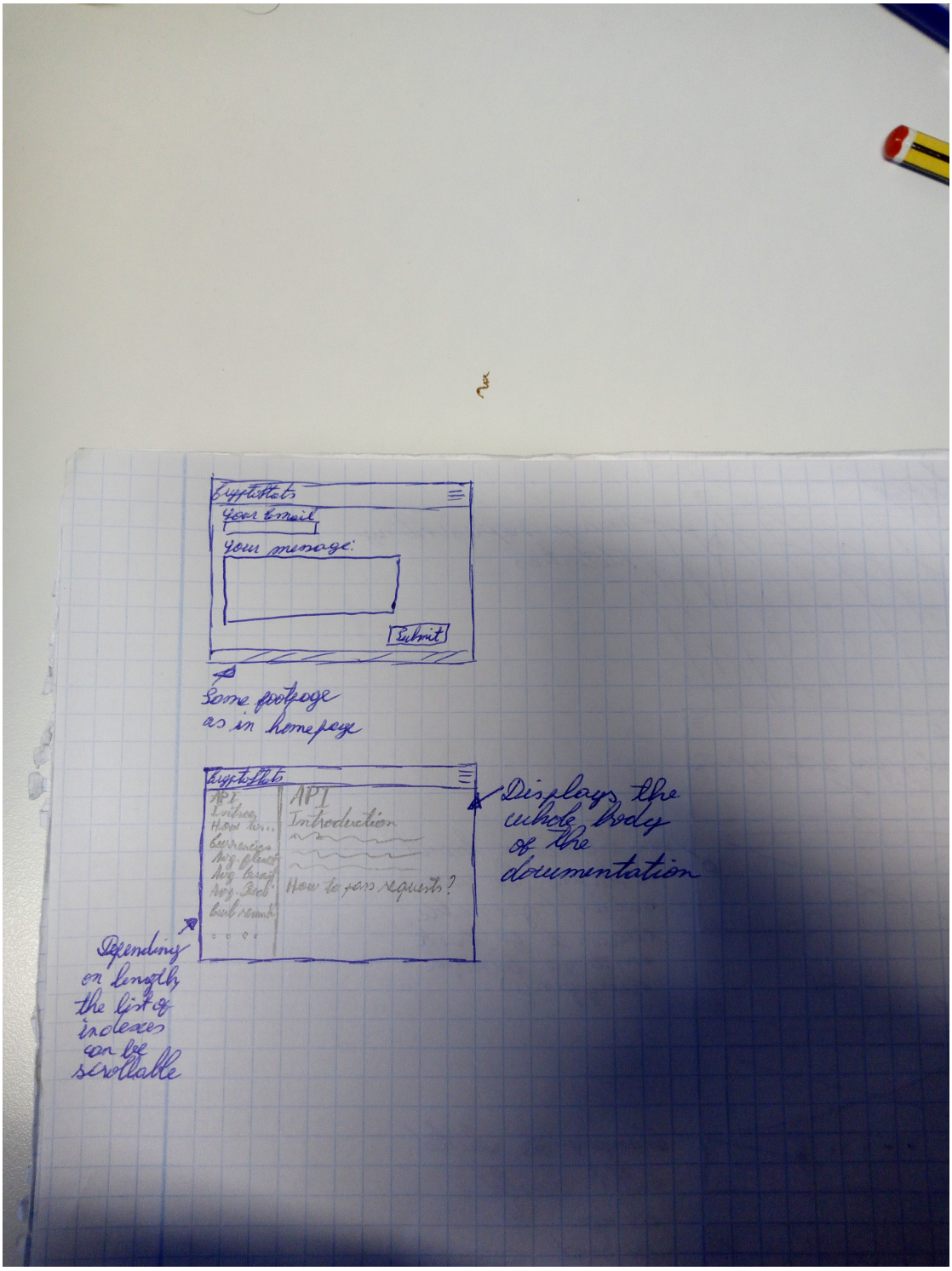

Figure 5.4: Homepage

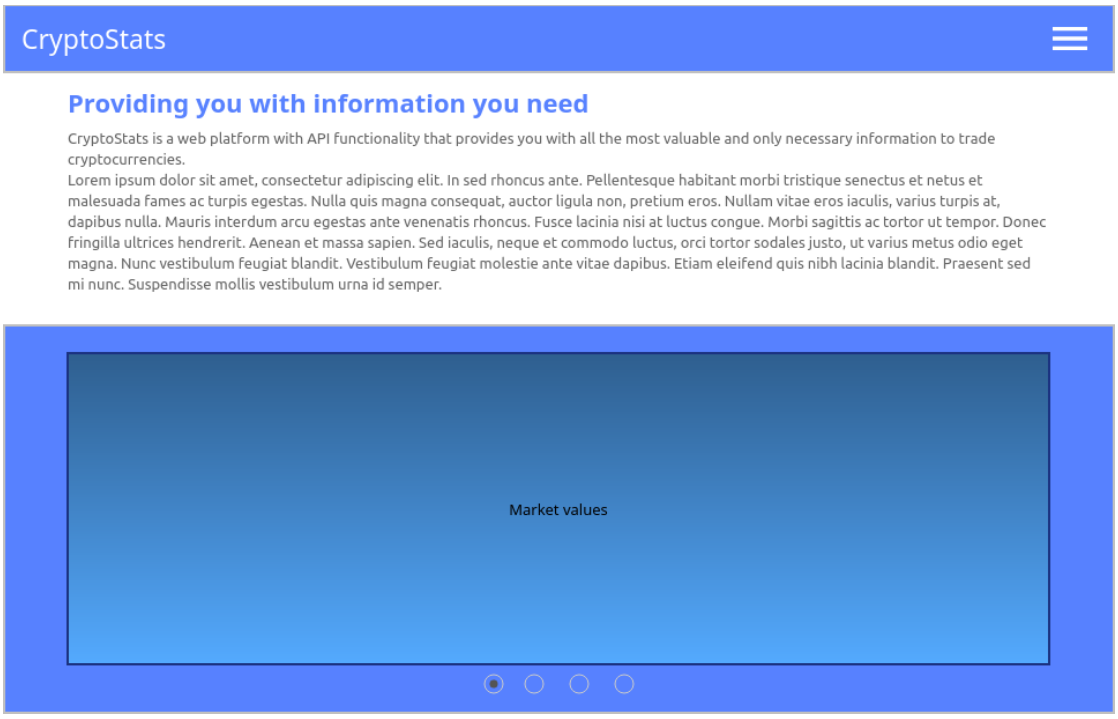

Figure 5.5: Homepage Bottom

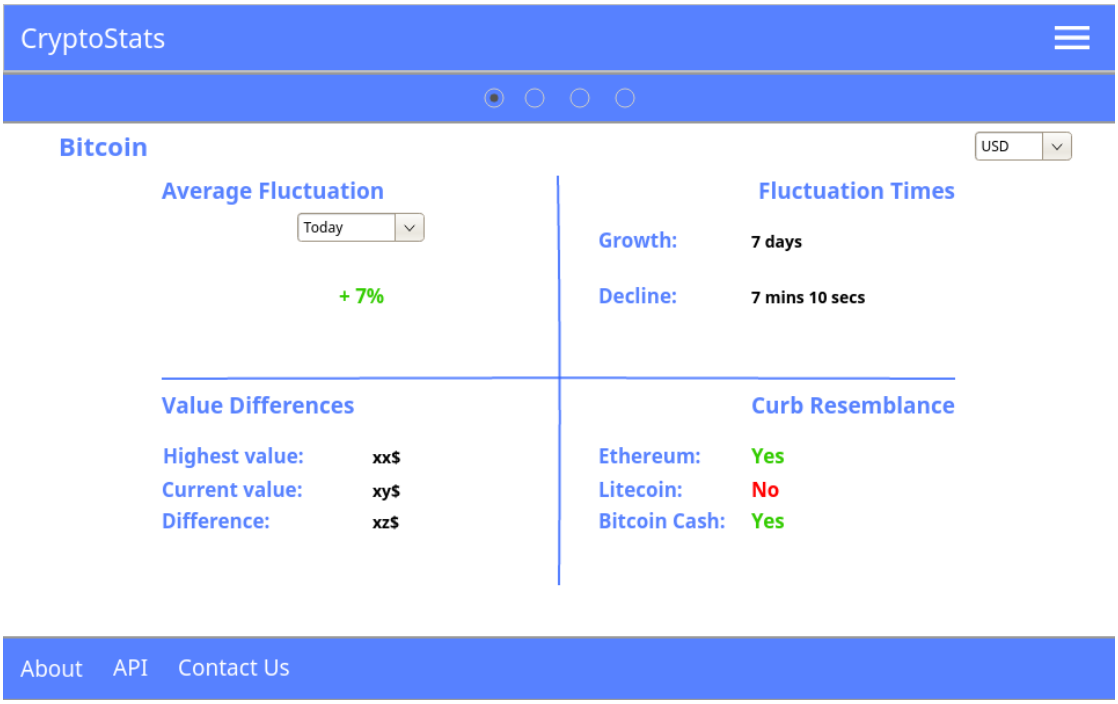

Figure 5.6: About Page

Figure 5.7: Contact Us Page

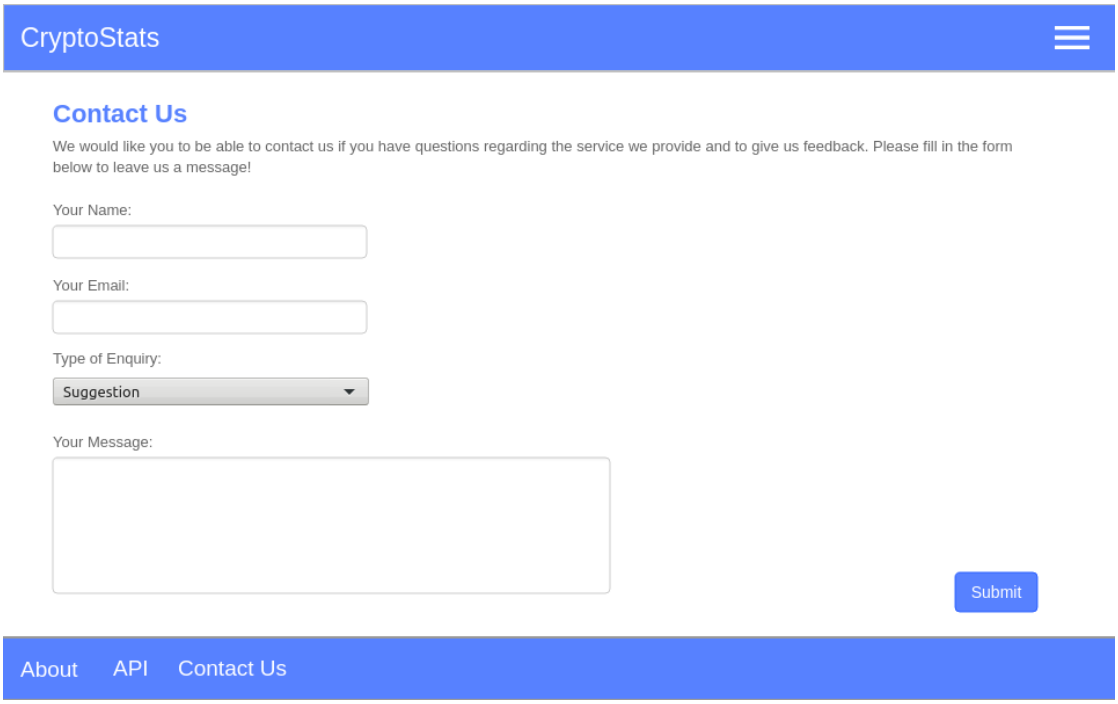

Figure 5.8: API Page

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 funcionality 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, to that everyone that is 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

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;
using CryptoStats.Models;

namespace CryptoStats
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var host = BuildWebHost(args);

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

                try
                {
                    SeedData.Initialize(services);
```

```
                }
                catch (Exception ex)
                {
                    var logger = services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "An error occurred seeding the DB.");
                }
            }

            host.Run();
        }

        public static IWebHost BuildWebHost(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>()
                .Build();
    }
}
```

Figure 8.2: Startup.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Sqlite;
using CryptoStats.Models;

namespace CryptoStats
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }
```

```
24          public IConfiguration Configuration { get; }
25          // This method gets called by the runtime. Use this method to add services to
26          public void ConfigureServices(IServiceCollection services)
27          {
28              services.AddDbContext<CryptoContext>(opt => opt.UseSqlite("Data Source=ap
29              services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_
30          }
31          // This method gets called by the runtime. Use this method to configure the H
32          public void Configure(IApplicationBuilder app, IHostingEnvironment env)
33          {
34              if (env.IsDevelopment())
35              {
36                  app.UseDeveloperExceptionPage();
37              }
38              else
39              {
40                  app.UseHsts();
41              }
42              app.UseHttpsRedirection();
43              app.UseMvc();
44          }
45      }
46  }
```

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

```
using System.Collections.Generic;

namespace CryptoStats.Models
{
    public class Exchange
    {
        public int ExchangeId{get;set;}
        public string Name{get;set;}

        //Setting list for foreign key
        public List<Stat> Stats{get;set;}
    }
}
```

Figure 8.5: Stat.cs

```
using System;

namespace CryptoStats.Models
{
    public class Stat
    {
        public int StatId{get;set;}
        public DateTime startDate {get;set;}
        public DateTime endDate{get;set;}
        public decimal HighestLatestDiff{get;set;}
        public decimal avgDiff {get;set;}
        public TimeSpan avgGrowthTime {get;set;}
        public TimeSpan avgDeclineTime {get;set;}

        //Setting variables for foreign key
        public int ExchangeId{get;set;}
        public Exchange Exchange {get;set;}
    }
}
```

Figure 8.6: SeedData.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using System;
using System.Linq;
using System.Collections.Generic;
using CryptoStats.GDAX;
using CryptoStats.GDAX.Services.Products.Models;
```

```
using CryptoStats.GDAX.Services.Products.Types;
using CryptoStats.GDAX.Shared.Types;

namespace CryptoStats.Models
{
    public static class SeedData
    {
        private static GDAXClient client = new GDAXClient();
        public static void Initialize(IServiceProvider serviceProvider)
        {
            using(var context = new CryptoContext(serviceProvider.GetRequiredService<
            {
                if(context.Exchanges.Any() && context.Stats.Any())
                {
                    return;
                }

                if(!context.Exchanges.Any())
                {
                    List<Exchange> exchanges = new List<Exchange>();
                    var products = client.ProductsService.GetAllProductsAsync().Resul
                    foreach(Product prod in products)
                    {
                        exchanges.Add(new Exchange(){Name = prod.Id.ToString()});
                    }
                    context.Exchanges.AddRange(exchanges);
                    context.SaveChanges();
                }

                if(!context.Stats.Any())
                {
                    foreach(Exchange e in context.Exchanges)
                    {
                        seedStat(e.ExchangeId, DateTime.Now.AddDays(-7), DateTime.Now
                    }

                }
            }
        }

        public static void seedStat(int exchangeID, DateTime startDate, DateTime endD
        {
            using(context)
            {
                foreach(Product p in client.ProductsService.GetAllProductsAsync().Res
```

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

```
        {
            inGrowth = true;
            declineTimes.Add(growthCandles.Last().Tim
            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.Sub
    }
    else
    {
        declineTimes.Add(growthCandles.Last().Time.Su
    }

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

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

```
                inGrowth = true;
                declineTimes.Add(growthCandles.Last().Tim
                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.Sub
        }
        else
        {
            declineTimes.Add(growthCandles.Last().Time.Su
        }

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

        totalTimes = 0;
        foreach(TimeSpan time in declineTimes)
        {
            totalTimes += t.TotalMilliseconds;
        }
        adder.avgDeclineTime = TimeSpan.FromMilliseconds(
        context.Stats.Add(adder);
        endDate.AddDays(-1);
```

```
                                    }
                                    context.SaveChanges();
                            }
                        }
                    }
                }
            }
        }
```

Figure 8.7: DbContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace CryptoStats.Models
{
    public class CryptoContext: DbContext
    {
        public CryptoContext(DbContextOptions<CryptoContext> options) : base(options)
        {
        }

        public DbSet<Exchange> Exchanges{get;set;}
        public DbSet<Stat> Stats{get;set;}
    }
}
```

8.3.3 Controllers

Figure 8.8: ExchangesController.cs

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using System;
using CryptoStats.Models;

namespace CryptoStats.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ExchangesController : ControllerBase
    {
        private readonly CryptoContext _context;

        public ExchangesController(CryptoContext context)
```

```
        {
            _context = context;
        }

        [HttpGet]
        public ActionResult<List<Exchange>> GetAll()
        {
            return _context.Exchanges.ToList();
        }

        [HttpGet("{id}", Name="GetExchange")]
        public ActionResult<Exchange> GetById(int id)
        {
            var item = _context.Exchanges.Find(id);
            if(item == null)
            {
                return NotFound();
            }
            return item;
        }
    }
}
```

Figure 8.9: ExchangeStatsController.cs

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using System;
using CryptoStats.Models;

namespace CryptoStats.Controllers
{
    [Route("api/")]
    [ApiController]
    public class ExchangeStatsController : ControllerBase
    {
        private readonly CryptoContext _context;

        public ExchangeStatsController(CryptoContext context)
        {
            _context = context;
        }

        [HttpGet("exchanges/{id}/stats")]
```

```
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          [HttpGet("exchanges/{id}/stats/id/{statId}")]
31          public ActionResult<Stat> GetByID(int id, int statId)
32          {
33              var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
34              if(exchange == null)
35              {
36                  return NotFound();
37              }
38              else
39              {
40                  return exchange.Stats.Where(s => s.StatId == statId).First();
41              }
42          }

43          [HttpGet("exchanges/{id}/stats/startDate/{startDate}")]
44          public ActionResult<List<Stat>> GetByStartDate(int id, DateTime startDate)
45          {
46              var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
47              if(exchange == null)
48              {
49                  return NotFound();
50              }
51              else
52              {
53                  return exchange.Stats.Where(s => s.startDate.Equals(startDate)).ToLis
54              }
55          }

56          [HttpGet("exchanges/{id}/stats/endDate/{endDate}")]
57          public ActionResult<List<Stat>> GetByEndDate(int id, DateTime endDate)
58          {
59              var exchange = _context.Exchanges.Where(e => e.ExchangeId== id).First();
```

```
            if(exchange == null)
            {
                return NotFound();
            }
            else
            {
                return exchange.Stats.Where(s => s.endDate.Equals(endDate)).ToList();
            }
        }

        [HttpGet("exchanges/{id}/stats/dates/{startDate}/{endDate}")]
        public ActionResult<Stat> GetByDates(int id, DateTime startDate, DateTime enc
        {
            var exchange = _context.Exchanges.Where(e => e.ExchangeId == id).First();
            if(exchange == null)
            {
                return NotFound();
            }
            else
            {
                return exchange.Stats.Where(s => s.startDate.Equals(startDate) && s.e
            }
        }
    }
}
```

Figure 8.10: StatsController.cs

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using System;
using CryptoStats.Models;

namespace CryptoStats.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class StatsController : ControllerBase
    {
        private readonly CryptoContext _context;

        public StatsController(CryptoContext context)
        {
            _context = context;
```

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

7   namespace CryptoStats.GDAX
8   {
9       public class GDAXClient
10      {
11          public GDAXClient(bool sandBox = false) : this(new HttpClient(), sandBox){}

12          public GDAXClient(IHttpClient httpClient, bool sandBox = false)
13          {
14              var clock = new Clock();
15              var httpRequestMessageService = new HttpRequestMessageService(sandBox);
16              var queryBuilder = new QueryBuilder();

17              ProductsService = new ProductsService(httpClient, httpRequestMessageServi
18          }

19          public ProductsService ProductsService{get;}
20      }
21  }
```

Figure 8.12: ProductsService.cs

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

```
using System.Net.Http;
using System.Threading.Tasks;
using CryptoStats.GDAX.Services;
using CryptoStats.GDAX.Network.HttpClient;
using CryptoStats.GDAX.Network.HttpRequest;
using CryptoStats.GDAX.Services.Products.Models;
using CryptoStats.GDAX.Services.Products.Models.Responses;
using CryptoStats.GDAX.Services.Products.Types;
using CryptoStats.GDAX.Shared.Types;
using CryptoStats.GDAX.Shared.Utilities.Extensions;
using CryptoStats.GDAX.Shared.Utilities.Queries;

namespace CryptoStats.GDAX.Services.Products
{
    public class ProductsService: AbstractService
    {
        private readonly IQueryBuilder queryBuilder;

        public ProductsService(IHttpClient httpClient, IHttpRequestMessageService htt
        {
            this.queryBuilder = queryBuilder;
        }

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

        public async Task<ProductsOrderBookResponse> GetProductOrderBookAsync(
                        ProductType productId,
                        ProductLevel productLevel = ProductLevel.One)
        {
            var productsOrderBookJsonResponse = await SendServiceCall<ProductsOrderBo
            var productOrderBookResponse = ConvertProductOrderBookResponse(productsOr

            return productOrderBookResponse;
        }

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

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

```
43              return await SendServiceCall<ProductStats>(HttpMethod.Get, $"/products/{p
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<ProductT
51              return httpResponseMessage;
52          }
53          public async Task<IList<Candle>> GetHistoricRatesAsync(
54                          ProductType productPair,
55                          DateTime start,
56                          DateTime end,
57                          CandleGranularity granularity)
58          {
59              const int maxPeriods = 300;
60              var candleList = new List<Candle>();
61              DateTime? batchEnd = end;
62              DateTime batchStart;
63              var maxBatchPeriod = (int)granularity * maxPeriods;
64              do
65              {
66                  if (batchEnd == null) {
67                      break;
68                  }
69                  batchStart = batchEnd.Value.AddSeconds(-maxBatchPeriod);
70                  if (batchStart < start) batchStart = start;
71                  candleList.AddRange(await GetHistoricRatesAsync(productPair, batchSta
72                  batchEnd = candleList.LastOrDefault()?.Time;
73              } while (batchStart > start);
74              return candleList;
75          }
```

```
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/{
89          }
90          private ProductsOrderBookResponse ConvertProductOrderBookResponse(
91              ProductsOrderBookJsonResponse productsOrderBookJsonResponse,
92              ProductLevel productLevel)
93          {
94              var askList = productsOrderBookJsonResponse.Asks.Select(product => produc
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 => produc
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(productsOrde
113             return productOrderBookResponse;
114         }
```

```
115         }
116     }
```

8.4 Testing

References