Crypto Marketplace Task

Documentation

by
Fabio Marku

19/08/2023

Contents

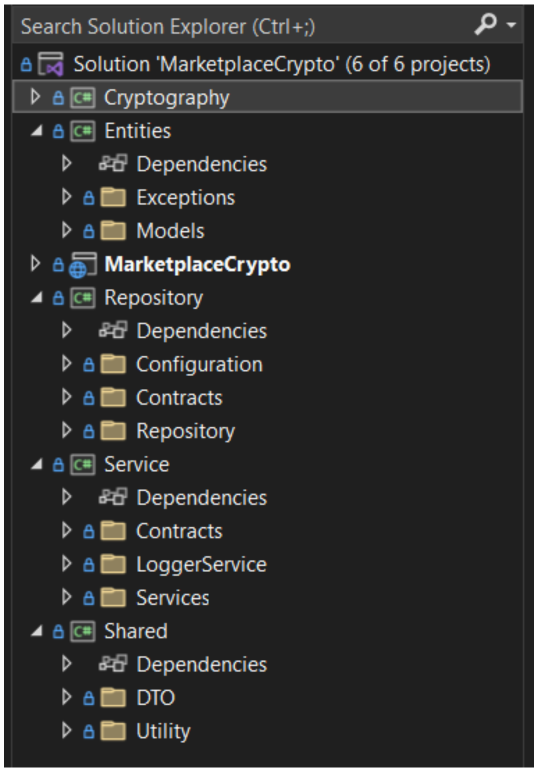
Introduction

The main objective of this task is the creation of a Crypto Marketplace service that will allow users to interact with different cryptocurrencies, see real-time market prices, be able to dive inside the graph for the selected crypto, and also track each selected data on the user Watchlist and other operations.

Architecture and Design

The way I've decided to implement the architecture of this project is something similar to CLEAN / ONION Architecture, where creating different architecture layers to separate the logic of the code.

**Id like to mention that I've tried to keep a fine-line for the structure of this project between "just completing the task" to "make this good as possible". Because the way I've implemented it can also be enhanced even more (which will take me more time(and it's a big issue for me here)) BUT also could be simplified. I'll try to explain them later on **



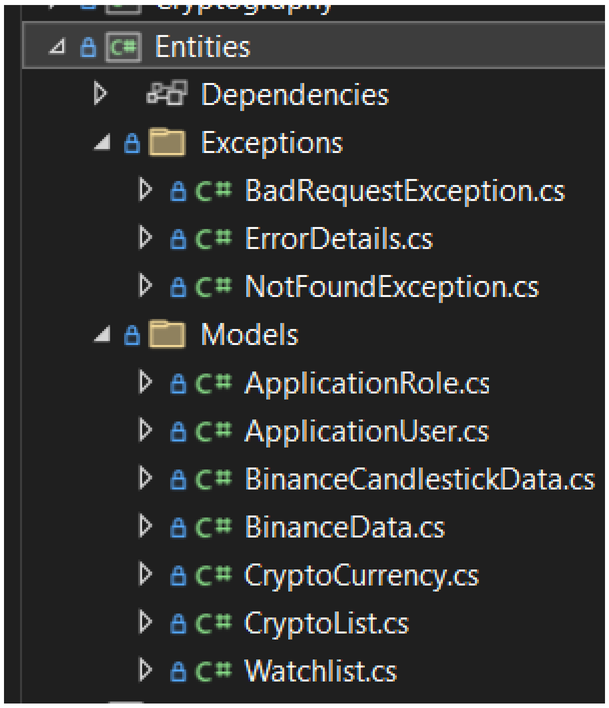
```
Search Solution Explorer (Ctrl+;)
Solution 'MarketplaceCrypto' (6 of 6 projects)
  Cryptography
  Entities
    Dependencies
    Exceptions
    Models
  MarketplaceCrypto
  Repository
    Dependencies
    Configuration
    Contracts
    Repository
  Service
    Dependencies
    Contracts
    LoggerService
    Services
  Shared
    Dependencies
    DTO
    Utility
```

First I've created the Main solution (MarketplaceCrypto) and then for each layer I've created them as Class Libraries instead of adding all of them to a single project .

ENTITIES

The *domain layer* contains the logic, like the entities and their specifications.

At the EXCEPTIONS folder , I’ve created the necessary entities to handle the error exceptions. And the MODELS folder contains all the other entities used in the application.

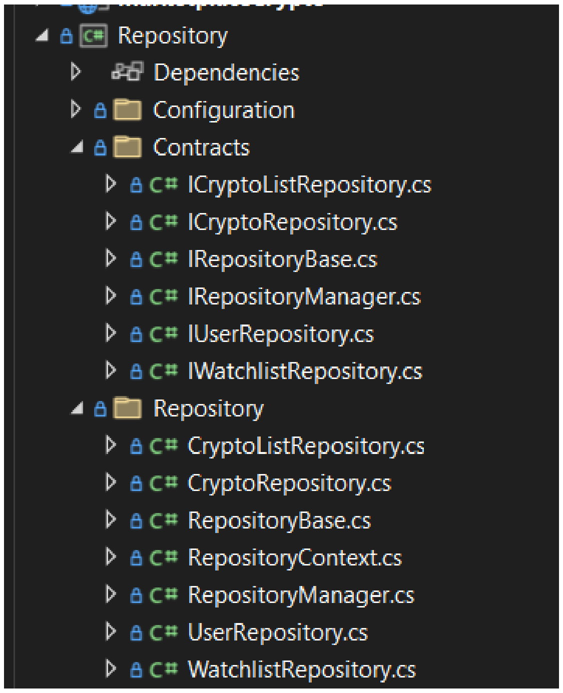


```
Entities
    Dependencies
    Exceptions
        BadRequestException.cs
        ErrorDetails.cs
        NotFoundException.cs
    Models
        ApplicationRole.cs
        ApplicationUser.cs
        BinanceCandlestickData.cs
        BinanceData.cs
        CryptoCurrency.cs
        CryptoList.cs
        Watchlist.cs
```

REPOSITORY

The *repository layer* act as a middle layer between the service layer and model objects. We will also keep all the database migrations and database context Objects in this layer. We will add the interfaces that consist of the data access pattern for operating CRUD operations with the database.

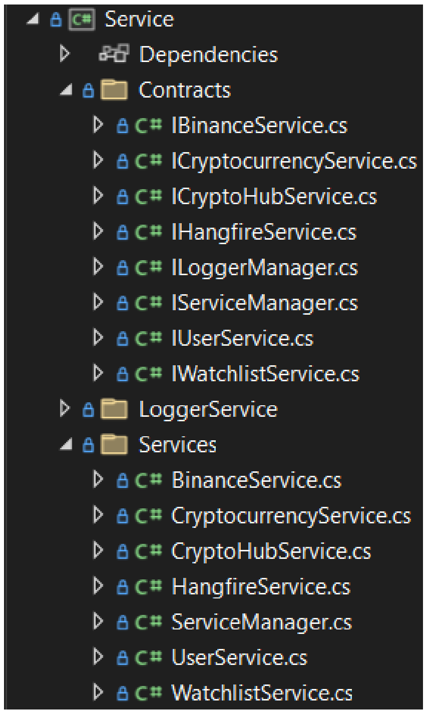
The CONTRACTS folder contains all the interfaces for each logic that will be implemented at the application. The REPOSITORY folder has all the class-es that implement each of their corresponding interfaces and I’ve also created a generic RepositoryBase that handles the CRUD operations for each class that extends it.



```
Repository
    Dependencies
    Configuration
    Contracts
        ICryptoListRepository.cs
        ICryptoRepository.cs
        IRepositoryBase.cs
        IRepositoryManager.cs
        IUserRepository.cs
        IWatchlistRepository.cs
    Repository
        CryptoListRepository.cs
        CryptoRepository.cs
        RepositoryBase.cs
        RepositoryContext.cs
        RepositoryManager.cs
        UserRepository.cs
        WatchlistRepository.cs
```

And also, RepositoryContext : Using Entity Framework Core setup for managing the database and its corresponding entities.

SERVICE



```
Service
  Dependencies
  Contracts
    IBinanceService.cs
    ICryptocurrencyService.cs
    ICryptoHubService.cs
    IHangfireService.cs
    ILoggerManager.cs
    IServiceManager.cs
    IUserService.cs
    IWatchlistService.cs
  LoggerService
  Services
    BinanceService.cs
    CryptocurrencyService.cs
    CryptoHubService.cs
    HangfireService.cs
    ServiceManager.cs
    UserService.cs
    WatchlistService.cs
```

This layer is used to communicate with the *presentation* and *repository* layer. This layer holds all the business logic of the entities. The services interfaces are kept separate from their implementation for loose coupling and separation of concerns.

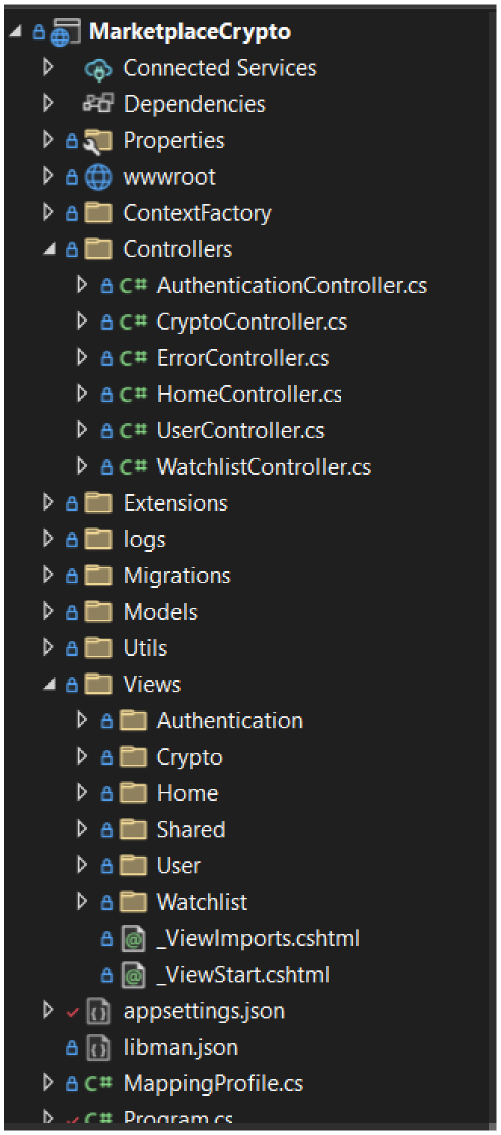
Now here I've could have extended it a little bit more , where a new Class Library project should have been created like: Service.Contracts , which would have only the interfaces , and the logic would be separated even more (The same thing with the REPOSITORY interfaces).
Another important class is the ServiceManager class, which serves as a central component responsible for managing and providing various services throughout the application.
-**Dependency Management:** The ServiceManager handles the instantiation and lifecycle management of different services required by the application.

- **Centralized Creation**: It consolidates the creation of services in one place, ensuring consistent and controlled instantiation across the application.

-**Lazy Initialization:** Services are created only when needed, improving startup performance and resource efficiency.

-**Abstraction and Encapsulation:** The class abstracts away the details of service creation, promoting cleaner and more modular code.

-**Service Composition:** Allows easy modification or extension of services without impacting the rest of the application.



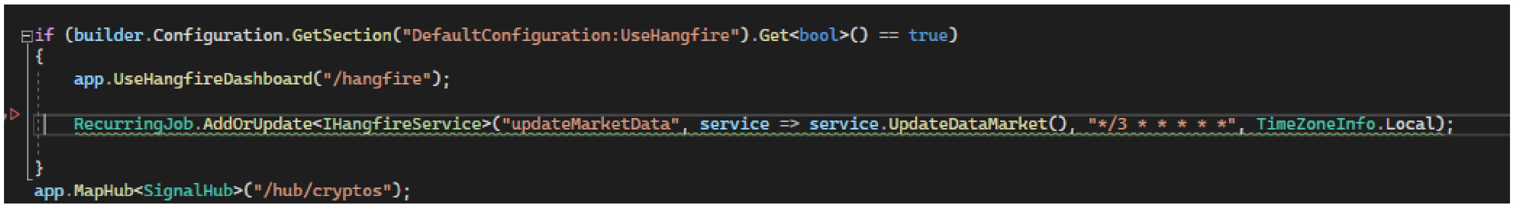
```
MarketplaceCrypto
  Connected Services
  Dependencies
  Properties
  wwwroot
  ContextFactory
  Controllers
    AuthenticationController.cs
    CryptoController.cs
    ErrorController.cs
    HomeController.cs
    UserController.cs
    WatchlistController.cs
  Extensions
  logs
  Migrations
  Models
  Utils
  Views
    Authentication
    Crypto
    Home
    Shared
    User
    Watchlist
    _ViewImports.cshtml
    _ViewStart.cshtml
  appsettings.json
  libman.json
  MappingProfile.cs
  Program.cs
```

The *presentation layer* , I've implemented it to the main project and separated the logic into CONTROLLERS and VIEW in different folders.
This could have a different implementation, such as the CONTROLLERS should have their own project separated from the main one to keep the logic and code clean, the same thing could be said for the VIEW . But lacking experience in the .NET MVC architecture made it a little difficult for me, and also lacking time to do more research for it made me implement it this way, BUT IT WORKS!

Real-Time Data Visualization

For the real-time data representation, I’ve decided to use 2 libraries such as: SignalR for the real-time update, and HangFire for background job-running to continuously fetch the data from the Binance API .

Starting with signalR, first I’ve created the necessary logic at the server side , so when we fetch the data from the API call , we handle these data to a method that will send it to the “front-end”/view , and there they are rendered/displayed at the user’s page. At the front-end/views, I’ve also established the connection of the signalR with the server, and the moment we get the data from the server, we populate the table with the new real-time data for the cryptocurrencies.

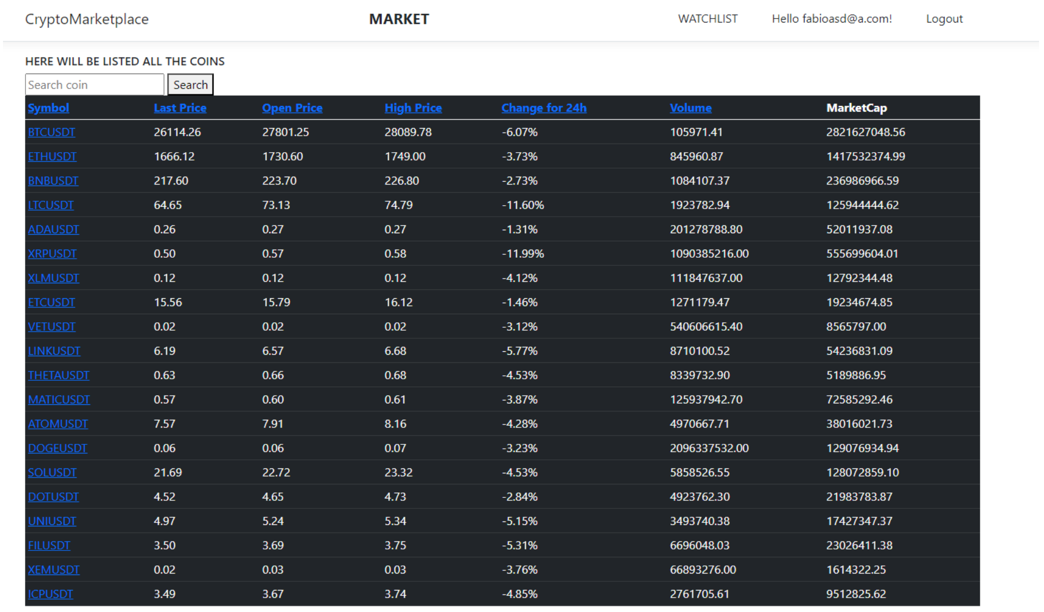


```
if (builder.Configuration.GetSection("DefaultConfiguration:UseHangfire").Get<bool>() == true)
{
    app.UseHangfireDashboard("/hangfire");

    RecurringJob.AddOrUpdate<IHangfireService>("updateMarketData", service => service.UpdateDataMarket(), "*/3 * * * * *", TimeZoneInfo.Local);

}
app.MapHub<SignalHub>("/hub/cryptos");
```

Now for the Hangfire, I’ve also created the logic for the background job scheduling, and assigned a method to the Hangfire service that will be scheduled for every 3 seconds (this number could be changed, maybe to a greater one, if we decrease it even more it will drastically affect the app performance, even though we are using multi-threading with asynchronous implementation!!) that will fetch the data from the api call and then the SignalR will do its work to update the values.

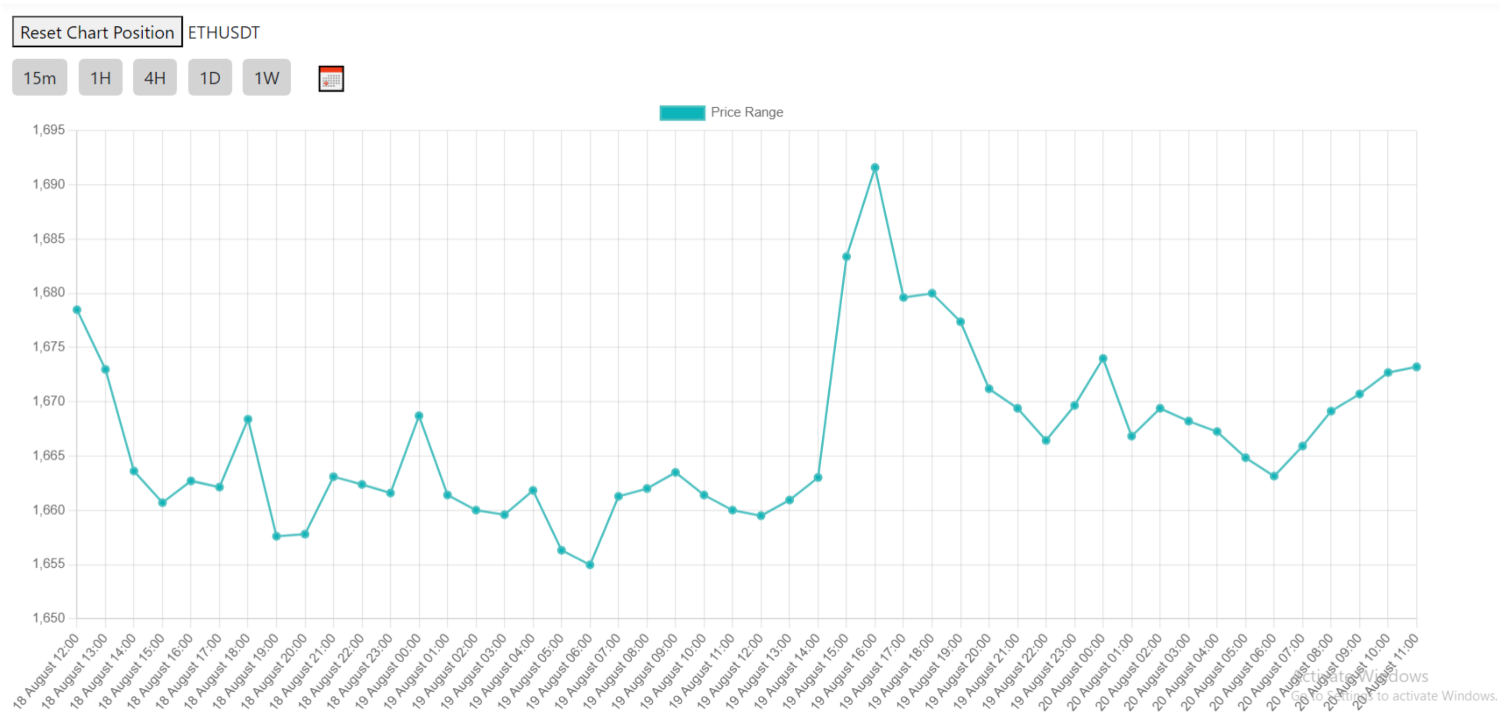


HERE WILL BE LISTED ALL THE COINS

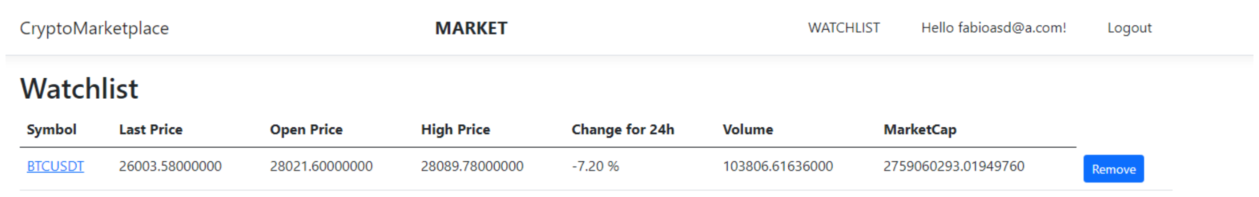
Symbol	Last Price	Open Price	High Price	Change for 24h	Volume	MarketCap
BTCUSDT	26114.26	27801.25	28089.78	-6.07%	105971.41	2821627048.56
ETHUSDT	1666.12	1730.60	1749.00	-3.73%	845960.87	1417532374.99
BNBUSDT	217.60	223.70	226.80	-2.73%	1084107.37	236986966.59
LTCUSDT	64.65	73.13	74.79	-11.60%	1923782.94	125944444.62
ADAUSDT	0.26	0.27	0.27	-1.31%	201278788.80	52011937.08
XRPUSDT	0.50	0.57	0.58	-11.99%	1090385216.00	555699604.01
XLMUSDT	0.12	0.12	0.12	-4.12%	111847637.00	12792344.48
ETCUSDT	15.56	15.79	16.12	-1.46%	1271179.47	19234674.85
VETUSDT	0.02	0.02	0.02	-3.12%	540606615.40	8565797.00
LINKUSDT	6.19	6.57	6.68	-5.77%	8710100.52	54236831.09
THETAUSDT	0.63	0.66	0.68	-4.53%	8339732.90	5189886.95
MATICUSDT	0.57	0.60	0.61	-3.87%	125937942.70	72585292.46
ATOMUSDT	7.57	7.91	8.16	-4.28%	4970667.71	38016021.73
DOGEUSDT	0.06	0.06	0.07	-3.23%	2096337532.00	129076934.94
SOLUSDT	21.69	22.72	23.32	-4.53%	5858526.55	128072859.10
DOTUSDT	4.52	4.65	4.73	-2.84%	4923762.30	21983783.87
UNIUSDT	4.97	5.24	5.34	-5.15%	3493740.38	17427347.37
FILUSDT	3.50	3.69	3.75	-5.31%	6696048.03	23026411.38
XEMUSDT	0.02	0.03	0.03	-3.76%	66893276.00	1614322.25
ICPUSDT	3.49	3.67	3.74	-4.85%	2761705.61	9512825.62

If we click on the coin symbol/name , it will redirect the user to a new page that will display the graph for that coin . For the construction of the graph I’ve used ChartJS library. The x-axis represents the TIME in 1-hour interval as default, and the y-axis represents the price.

The user has the options to select different time intervals such as 15minute , 1hour , 4hour, 1Day , 1Week and also a specific date range. Based on the selection of these options , the x-axis with DATE/TIME will have different sizes , such as if the user selects 15m interval the graph will have the data for last 2 DAYS, if it selects 4hour interval it will have the data for the last 25days and so on for the others. And also, when the user is “lost” while diving into the graph, it has the possibility to go back to the initial position using the RESET CHART POSTION button .



Watchlists



CryptoMarketplace MARKET WATCHLIST Hello fabioasd@a.com! Logout

Watchlist

Symbol	Last Price	Open Price	High Price	Change for 24h	Volume	MarketCap	
BTCUSDT	26003.58000000	28021.60000000	28089.78000000	-7.20 %	103806.61636000	2759060293.01949760	Remove

When the user is registered, then logged in, it will redirect it to the MARKET page where all the cryptocurrencies are displayed. Now, the moment the user clicks onto a symbol/coin to see the details for it, this is the moment we handle the WATCHLIST, that coin will be added to the user watchlist.
Now the logic at the WatchlistService is that first we make sure that this symbol/coin DOES existing and then check if user have a watchlist or not, if so a new one will be created, and this symbol/coin will be added to the users WATCHLIST.
At the Watchlist page, the user can see and remove the items from its list.
Here I didn’t use signalR for updating the data, and they are updated only when the page is reloaded.

User-Friendly Interface

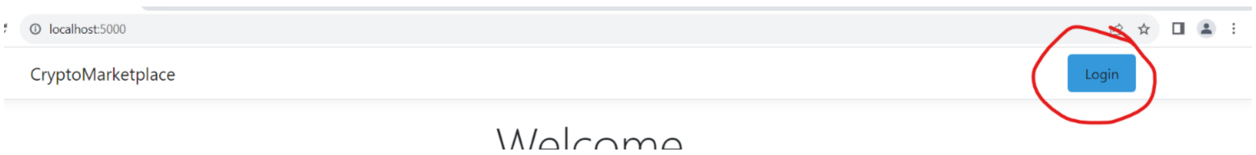
This is my least favorite part , I have to admit it!
For the UI I’ve tried to follow the common practices , such as to keep things simple and clear for the user so it could be easy to interact with the website.

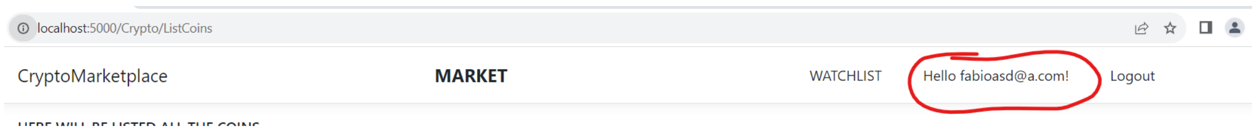
When the user will open the website , it will be directed to the home page that will display a welcoming message or some instruction how to use the application.
From here the user can redirect to login page either by clicking to the name of the application/logo or simply by clicking the LOGIN button , which is show at the top of the page with blue color (to be easily found).

When the user is redirected to the LOGIN page, it will require the EMAIL and PASSWORD , and also below the LOGIN button there is a link that will redirect the user to register page.

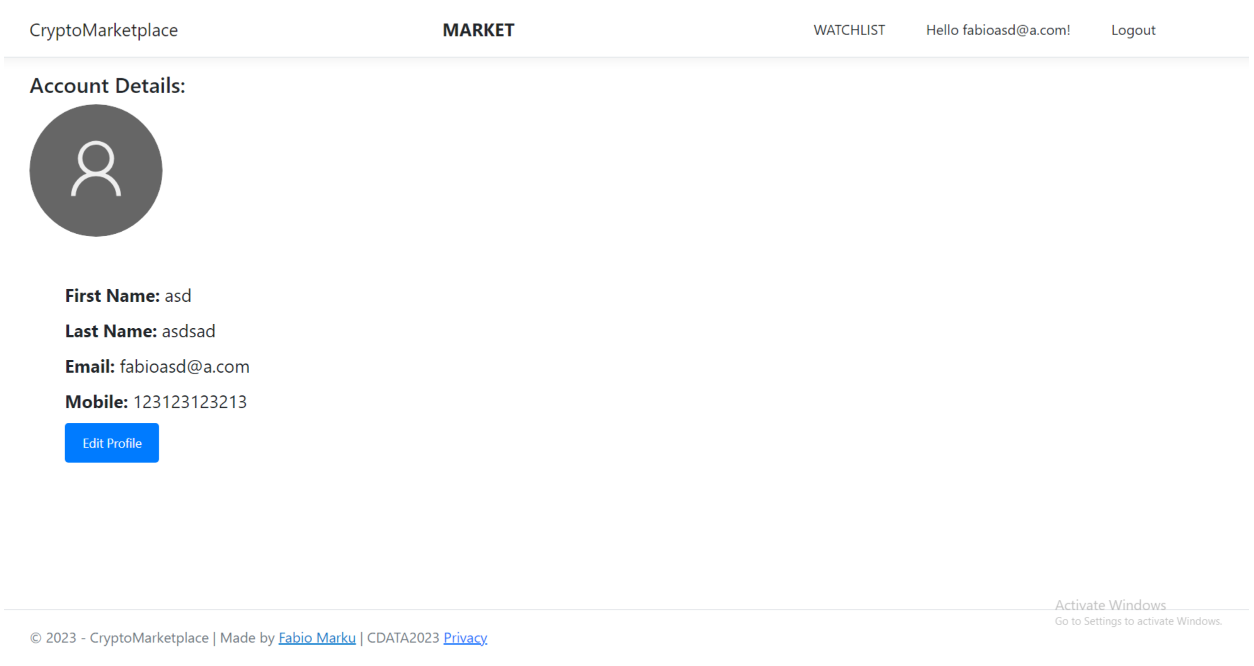
After the user is logged-in, it will redirected to the market page which contains all the cryptos with their real-time data such as last price, open price ,high price change for the 24 hours, volume and market cap.

At the top of page, the navigation bar has an element called WATCHLIST that will send the user to its watchlist page. Here the user can see its cryptocurrencies what has opened before, and also has the option to remove them.
Compared to the MARKET page where the data are being continuously updated with SignalR , here at the watchlist page the data for the cryptocurrencies are NOT CONTINOUSLY updated , the data ARE in real-time (the moment the page is opened or reloaded).





When user is logged-in , at the top of page it will be displayed an item :
Hello <name of the user> , and if the user clicks on this it will send to the User profile page , here the user can change its details .



CryptoMarketplace MARKET WATCHLIST Hello fabioasd@a.com! Logout

Account Details:

First Name: asd

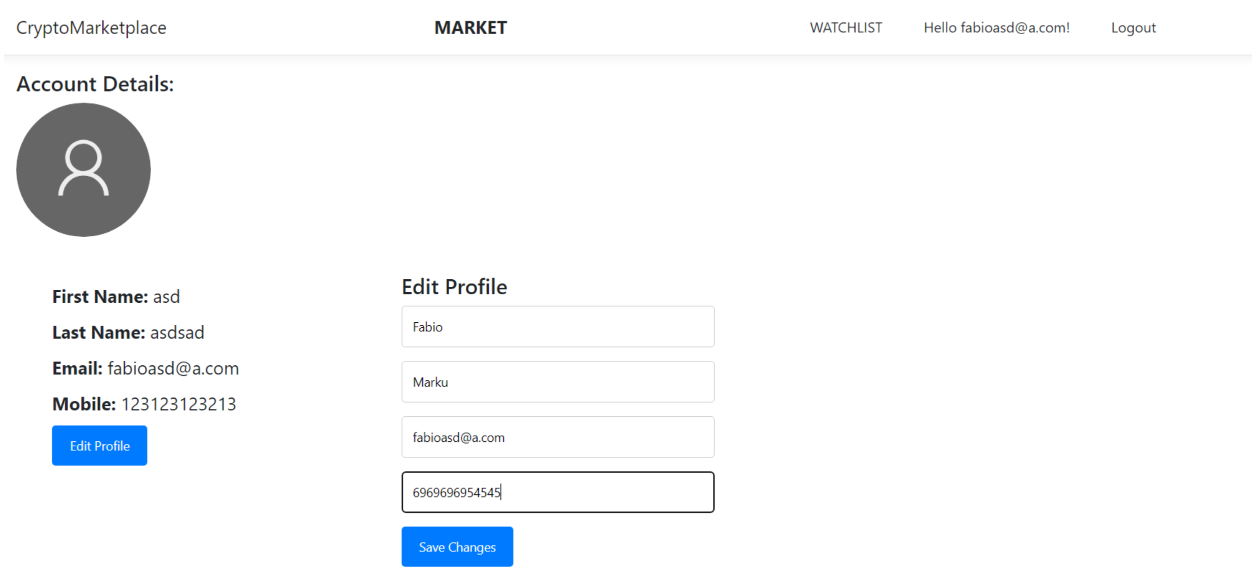
Last Name: asdsad

Email: fabioasd@a.com

Mobile: 123123123213

Edit Profile

© 2023 - CryptoMarketplace | Made by Fabio Marku | CDATA2023 Privacy



CryptoMarketplace MARKET WATCHLIST Hello fabioasd@a.com! Logout

Account Details:

First Name: asd

Last Name: asdsad

Email: fabioasd@a.com

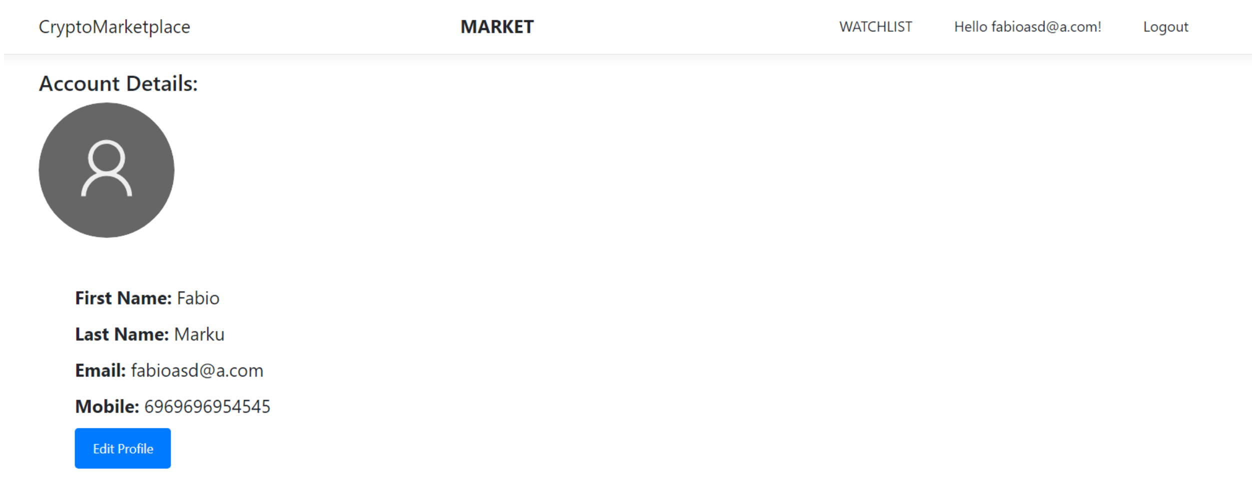
Mobile: 123123123213

Edit Profile

Edit Profile

Fabio
Marku
fabioasd@a.com
6969696954545

Save Changes



CryptoMarketplace MARKET WATCHLIST Hello fabioasd@a.com! Logout

Account Details:

First Name: Fabio

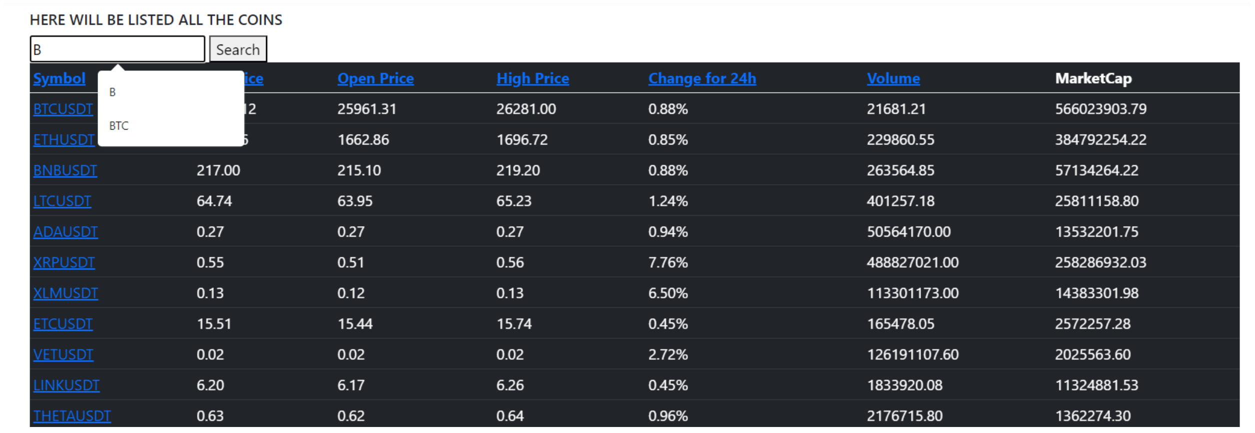
Last Name: Marku

Email: fabioasd@a.com

Mobile: 6969696954545

Edit Profile

Search & Sorting Functionality

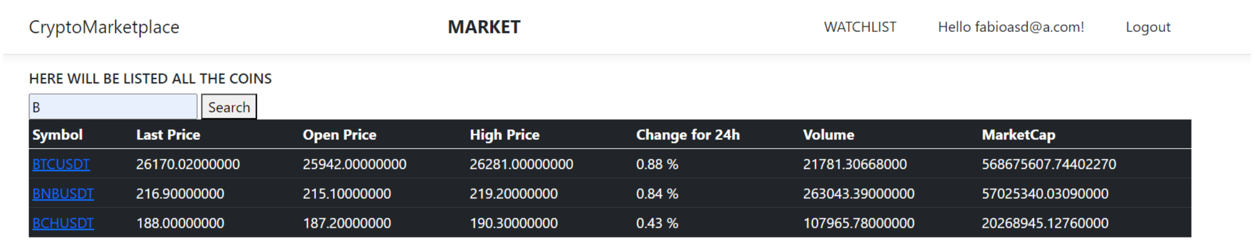


HERE WILL BE LISTED ALL THE COINS

B Search

Symbol	ice	Open Price	High Price	Change for 24h	Volume	MarketCap
BTCUSDT	2	25961.31	26281.00	0.88%	21681.21	566023903.79
ETHUSDT	5	1662.86	1696.72	0.85%	229860.55	384792254.22
BNBUSDT	217.00	215.10	219.20	0.88%	263564.85	57134264.22
LTCUSDT	64.74	63.95	65.23	1.24%	401257.18	25811158.80
ADAUSDT	0.27	0.27	0.27	0.94%	50564170.00	13532201.75
XRPUSDT	0.55	0.51	0.56	7.76%	488827021.00	258286932.03
XLMUSDT	0.13	0.12	0.13	6.50%	113301173.00	14383301.98
ETCUSDT	15.51	15.44	15.74	0.45%	165478.05	2572257.28
VETUSDT	0.02	0.02	0.02	2.72%	126191107.60	2025563.60
LINKUSDT	6.20	6.17	6.26	0.45%	1833920.08	11324881.53
THETAUSDT	0.63	0.62	0.64	0.96%	2176715.80	1362274.30

At the MARKET page where the cryptocurrencies are listed , at the top the table there is a search-bar that will be used to search for matching pattern or a specific symbol name.



HERE WILL BE LISTED ALL THE COINS

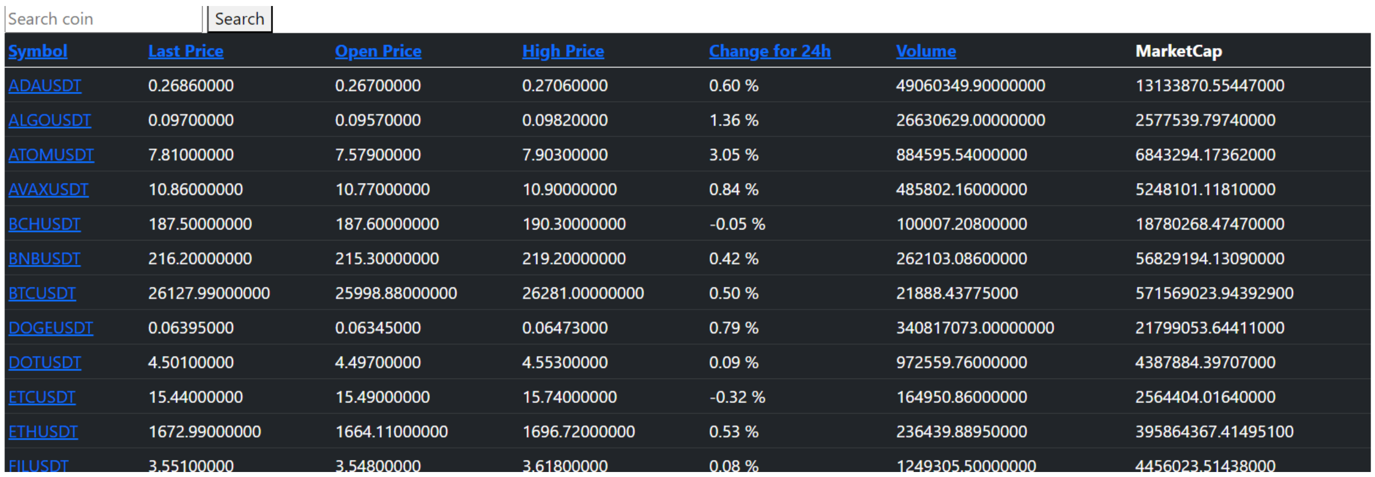
B Search

Symbol	Last Price	Open Price	High Price	Change for 24h	Volume	MarketCap
BTCUSDT	26170.02000000	25942.00000000	26281.00000000	0.88 %	21781.30668000	568675607.74402270
BNBUSDT	216.90000000	215.10000000	219.20000000	0.84 %	263043.39000000	57025340.03090000
BCHUSDT	188.00000000	187.20000000	190.30000000	0.43 %	107965.78000000	20268945.12760000

The logic of the search at the CryptocurrencyService is that first we check and load all the cryptos(since they are as default) , and then we select them based on the patter we have taken and then return the result. This doesn't seem pretty reliable , because another good solution would be like using Dapper , to take the pattern and then search directly at the database based on that pattern , without loading all the list of cryptos we have .

Now for the **sorting part** , I've implemented it in the way that when the user clicks in one of the headers of the table (except the MarketCap) , the data will be sorted based on that . The implementation of this is the same as the way all the data are listed . They share the same method logic implementation , but I've included a nullable variable that checks if the call is for sorting the data or not , so in this way we avoid the need to create redundant code for doing the same operation. BUT , the downfall of this is that they are only sorted in the ASCENDING order.

The initial idea was to make the headers as toggles one (with both descending and ascending) , so then clicking the headers would switch the sorting in both ways.
I could have solved it using a flag to check which order is it , and then invert it .
But I left It in that way !



Search coin | Search

Symbol	Last Price	Open Price	High Price	Change for 24h	Volume	MarketCap
ADAUSDT	0.26860000	0.26700000	0.27060000	0.60 %	49060349.90000000	13133870.55447000
ALGOUSDT	0.09700000	0.09570000	0.09820000	1.36 %	26630629.00000000	2577539.79740000
ATOMUSDT	7.81000000	7.57900000	7.90300000	3.05 %	884595.54000000	6843294.17362000
AVAXUSDT	10.86000000	10.77000000	10.90000000	0.84 %	485802.16000000	5248101.11810000
BCHUSDT	187.50000000	187.60000000	190.30000000	-0.05 %	100007.20800000	18780268.47470000
BNBUSDT	216.20000000	215.30000000	219.20000000	0.42 %	262103.08600000	56829194.13090000
BTCUSDT	26127.99000000	25998.88000000	26281.00000000	0.50 %	21888.43775000	571569023.94392900
DOGEUSDT	0.06395000	0.06345000	0.06473000	0.79 %	340817073.00000000	21799053.64411000
DOTUSDT	4.50100000	4.49700000	4.55300000	0.09 %	972559.76000000	4387884.39707000
ETCUSDT	15.44000000	15.49000000	15.74000000	-0.32 %	164950.86000000	2564404.01640000
ETHUSDT	1672.99000000	1664.11000000	1696.72000000	0.53 %	236439.88950000	395864367.41495100
FILUSDT	3.55100000	3.54800000	3.61800000	0.08 %	1249305.50000000	4456023.51438000

Cryptocurrencies sorted based on SYMBOL**

Evaluation Criteria

ASP.NET Core and C# Best Practices:

The project adheres to ASP.NET Core and C# best practices by implementing a well-structured architecture. The separation of responsibilities into different layers such as Entities, Repository, Service and Controllers demonstrates a strong understanding of architectural principles. Utilizing Entity Framework Core for database management aligns with industry best practices for data access in ASP.NET Core.

Completeness:

The project has successfully realized the creation of a Crypto Marketplace service, encompassing essential features as outlined in the requirements. These include real-time data visualization through SignalR and HangFire for background tasks, user watchlists, intuitive user interfaces for login and market information, search capabilities, and sortable data columns. Notably, the project also integrates interactive graphs with various time intervals to enhance the user experience.

Correctness:

The implementation of real-time data updates using SignalR and background tasks through HangFire demonstrates careful planning and accurate execution. The approach to managing user watchlists showcases well-considered logic. The use of these technologies ensures that data is updated seamlessly and that user interactions are appropriately managed.

Maintainability:

The project exhibits a commitment to maintainability through its adoption of a structured architecture. This involves the organization of code into separate class libraries for different layers, which facilitates code maintainability and testability. Furthermore, the separation of interfaces from implementations, as seen in the Service.Contracts and Repository interfaces, promotes modular development and allows for easier modifications in the future. Consideration of more comprehensive comments and documentation within the code could further enhance maintainability.

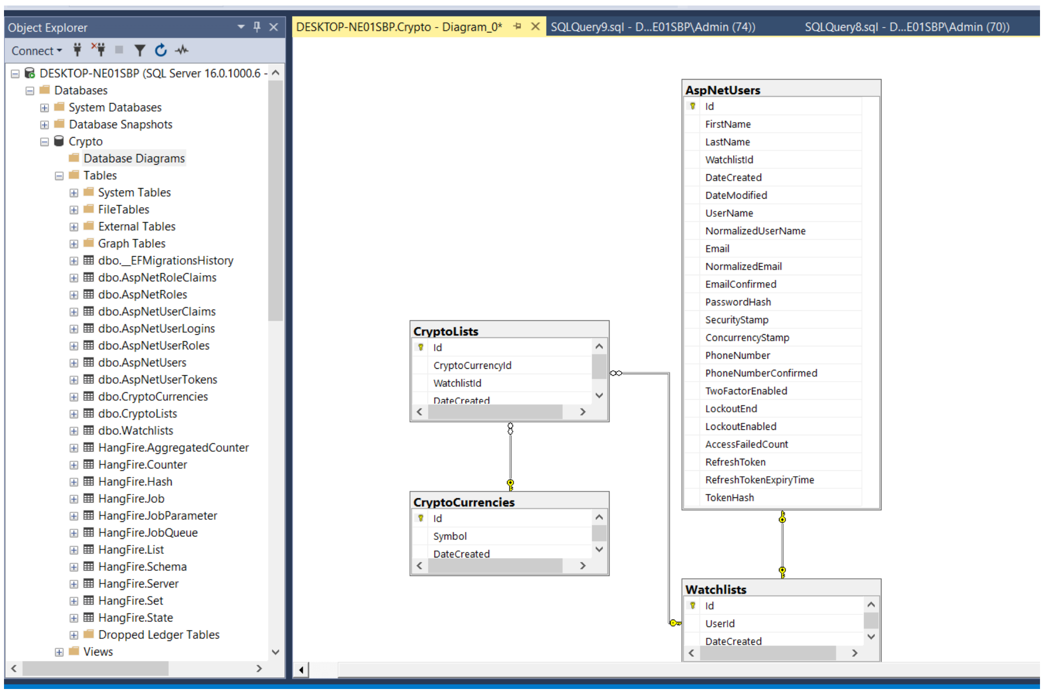
User Experience:

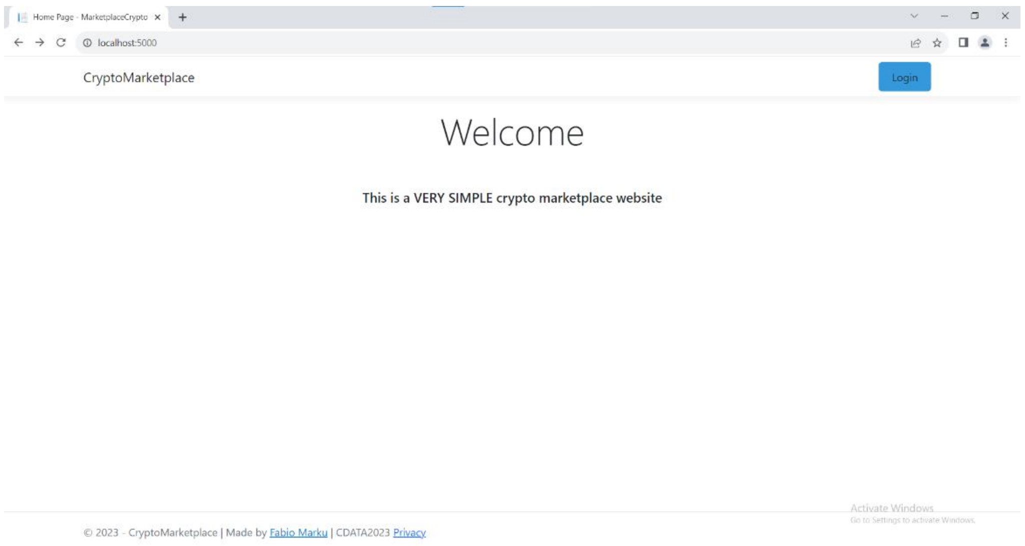
The user interface has been designed to offer a user-friendly and intuitive experience. Key features like real-time data updates, interactive graphs, and a user watchlist contribute to an engaging user experience. Design decisions aimed at keeping the UI simple and clear showcase a focus on enhancing user satisfaction. However, it's worth noting that the visual aesthetics and styling of the website may not be fully optimized. While the project's core functionality has been prioritized, future enhancements could include attention to the visual presentation of the application. Despite this, the project effectively meets the outlined objectives within the given constraints.

Conclusion

In conclusion, the project has made notable progress in the development of the Crypto Marketplace service. The chosen architecture and design decisions reflect a solid grasp of ASP.NET Core principles. While certain areas, such as visualization and the bidirectional sorting , could be refined for optimal results, the project aligns well with the objectives specified in the documentation.

Images





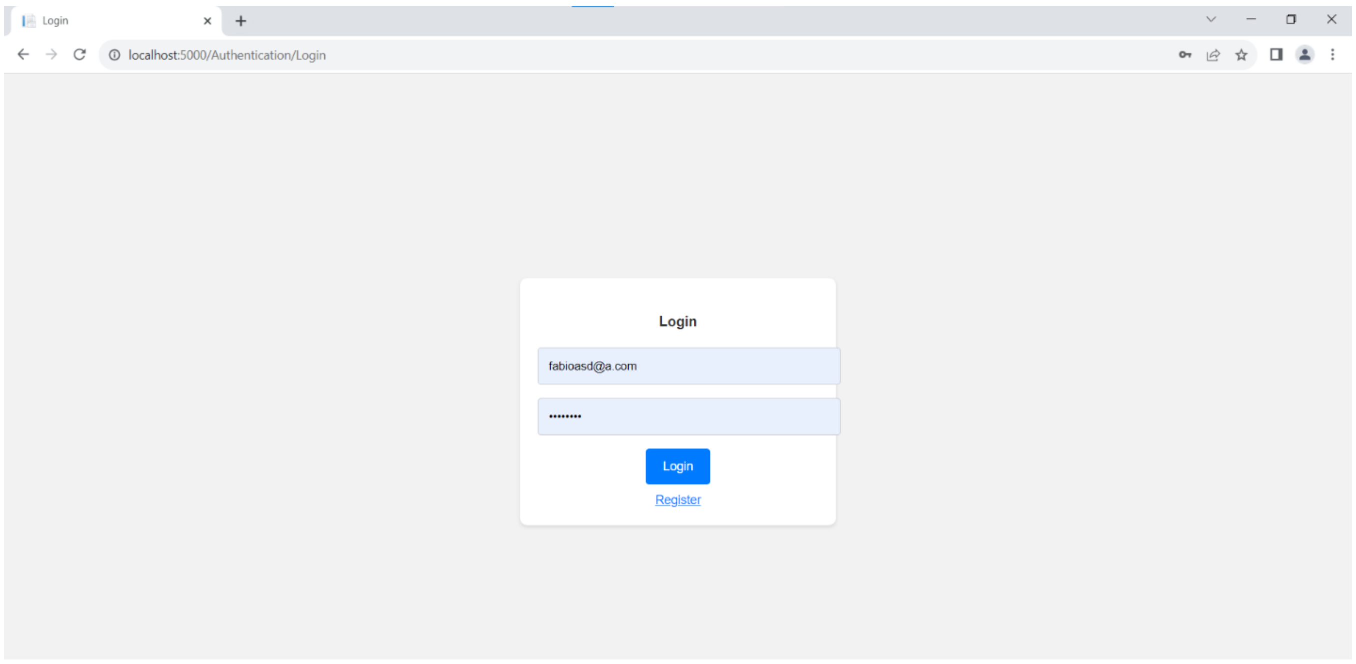
Home Page - MarketplaceCrypto
localhost:5000
CryptoMarketplace
Login

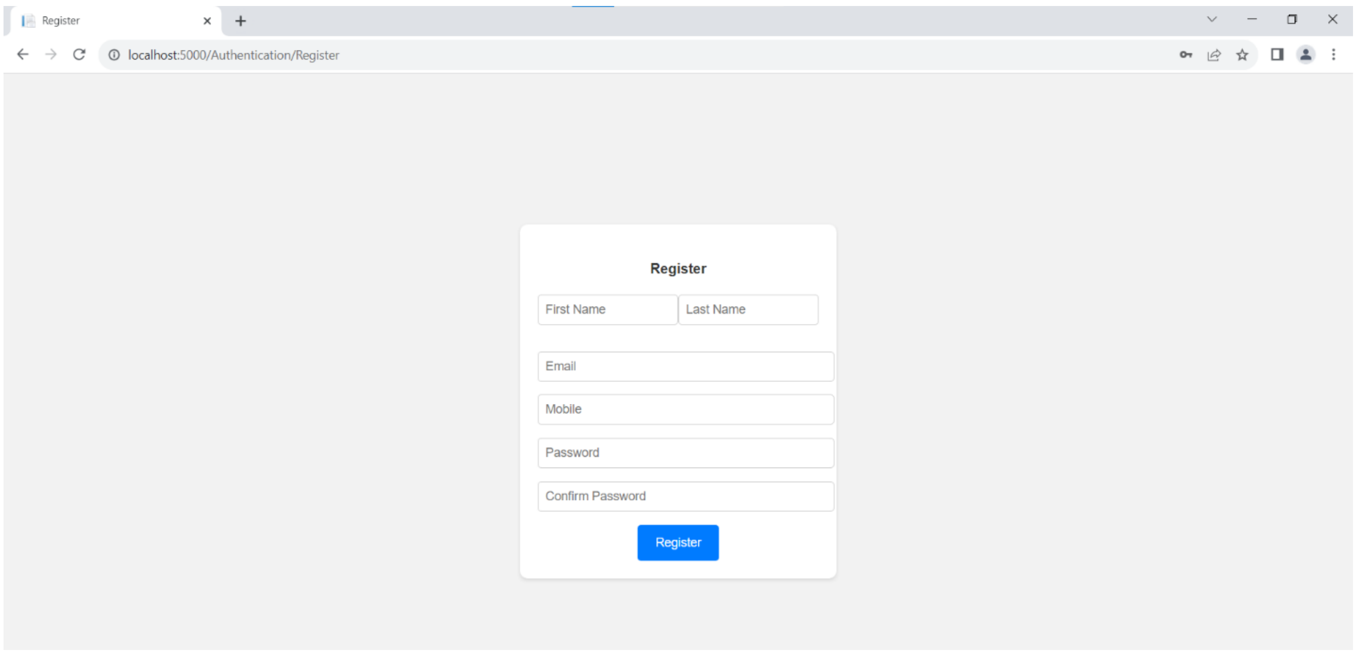
Welcome

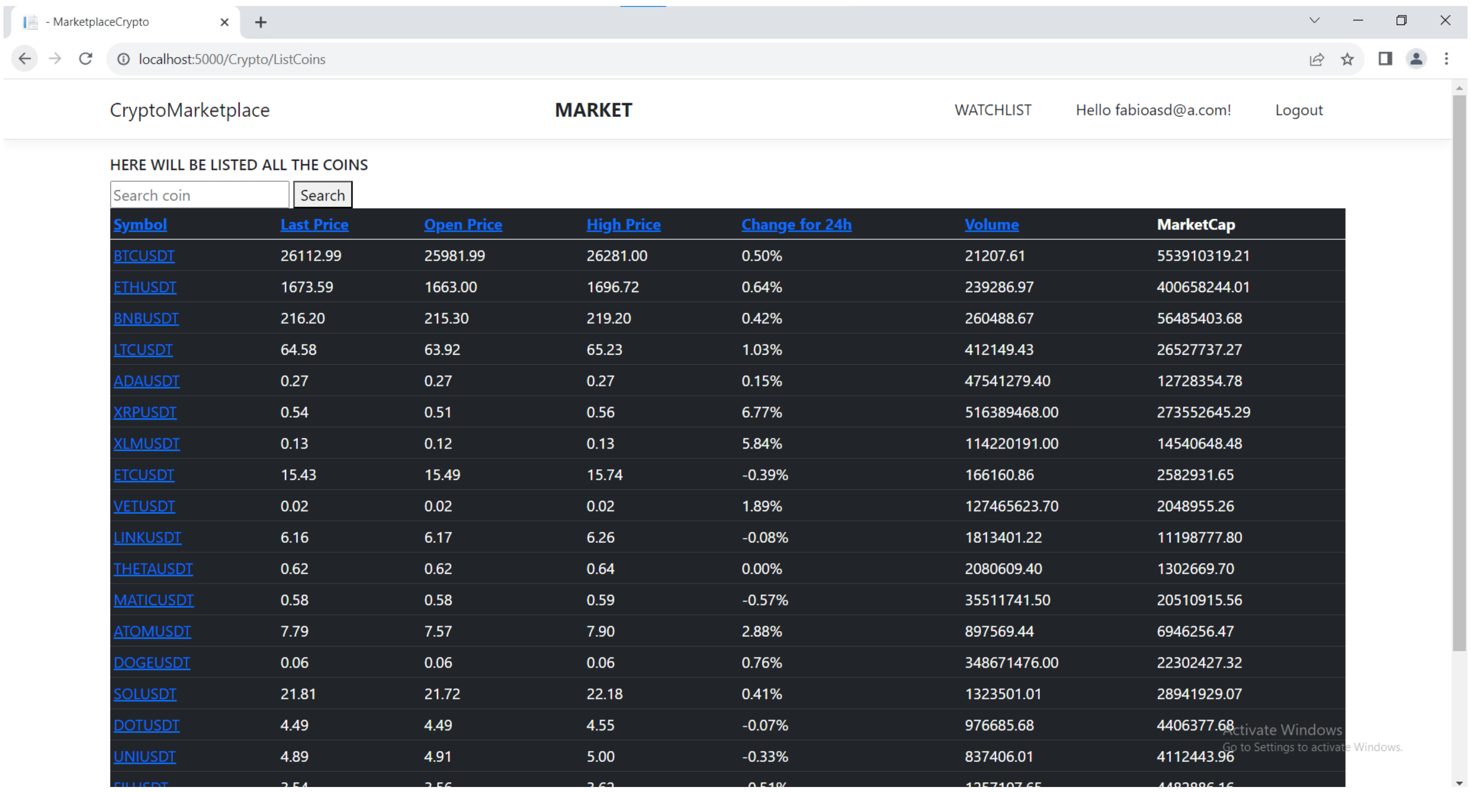
This is a VERY SIMPLE crypto marketplace website

© 2023 - CryptoMarketplace | Made by Fabio Marku | CDATA2023 Privacy

Activate Windows
Go to Settings to activate Windows.


Login
localhost:5000/Authentication/Login
Login
fabioasd@a.com
••••••••
Login
Register


Register
localhost:5000/Authentication/Register
Register
First Name
Last Name
Email
Mobile
Password
Confirm Password
Register

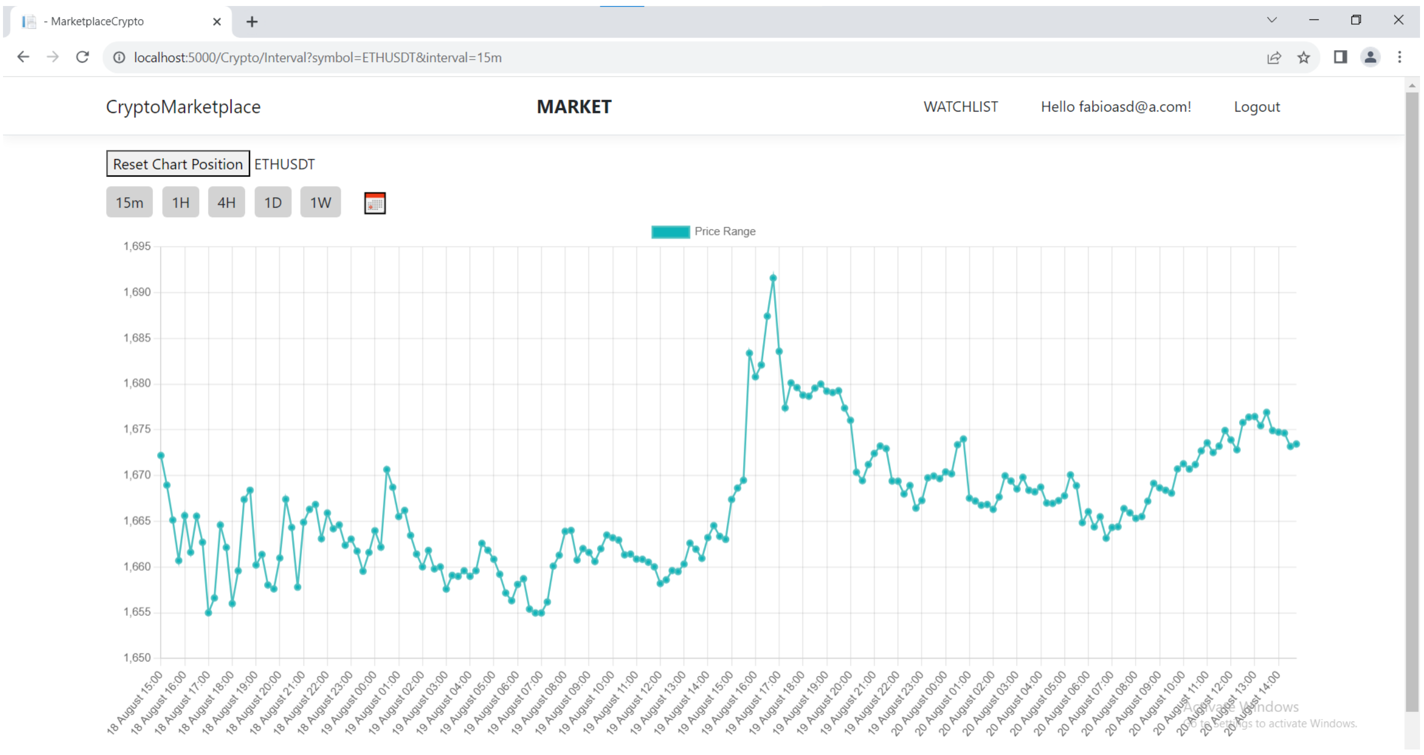


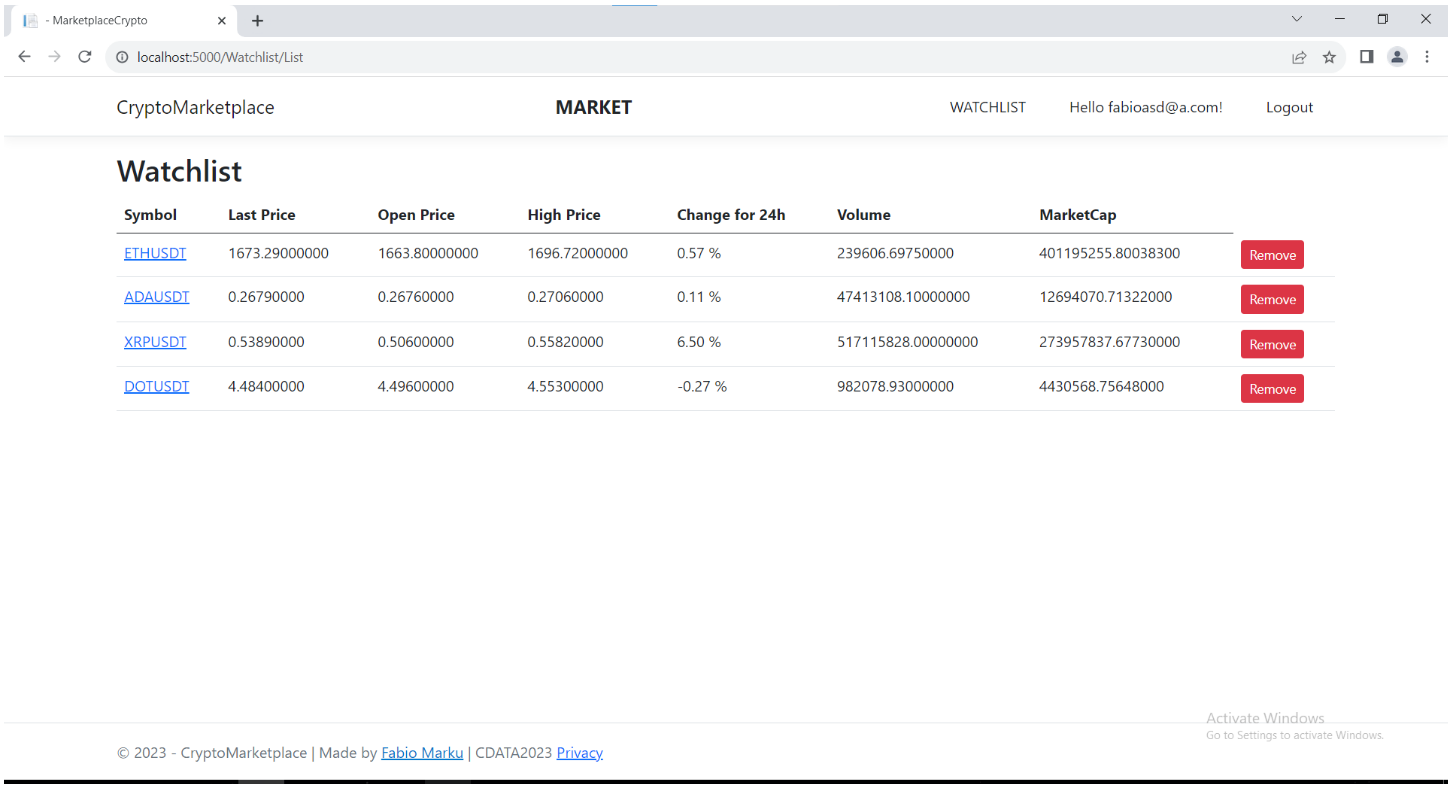
CryptoMarketplace MARKET WATCHLIST Hello fabioasd@a.com! Logout

HERE WILL BE LISTED ALL THE COINS

Search coin Search

Symbol	Last Price	Open Price	High Price	Change for 24h	Volume	MarketCap
BTCUSDT	26112.99	25981.99	26281.00	0.50%	21207.61	553910319.21
ETHUSDT	1673.59	1663.00	1696.72	0.64%	239286.97	400658244.01
BNBUSDT	216.20	215.30	219.20	0.42%	260488.67	56485403.68
LTCUSDT	64.58	63.92	65.23	1.03%	412149.43	26527737.27
ADAUSDT	0.27	0.27	0.27	0.15%	47541279.40	12728354.78
XRPUSDT	0.54	0.51	0.56	6.77%	516389468.00	273552645.29
XLMUSDT	0.13	0.12	0.13	5.84%	114220191.00	14540648.48
ETCUSDT	15.43	15.49	15.74	-0.39%	166160.86	2582931.65
VETUSDT	0.02	0.02	0.02	1.89%	127465623.70	2048955.26
LINKUSDT	6.16	6.17	6.26	-0.08%	1813401.22	11198777.80
THETAUSDT	0.62	0.62	0.64	0.00%	2080609.40	1302669.70
MATICUSDT	0.58	0.58	0.59	-0.57%	35511741.50	20510915.56
ATOMUSDT	7.79	7.57	7.90	2.88%	897569.44	6946256.47
DOGEUSDT	0.06	0.06	0.06	0.76%	348671476.00	22302427.32
SOLUSDT	21.81	21.72	22.18	0.41%	1323501.01	28941929.07
DOTUSDT	4.49	4.49	4.55	-0.07%	976685.68	4406377.68
UNIUSDT	4.89	4.91	5.00	-0.33%	837406.01	4112443.96

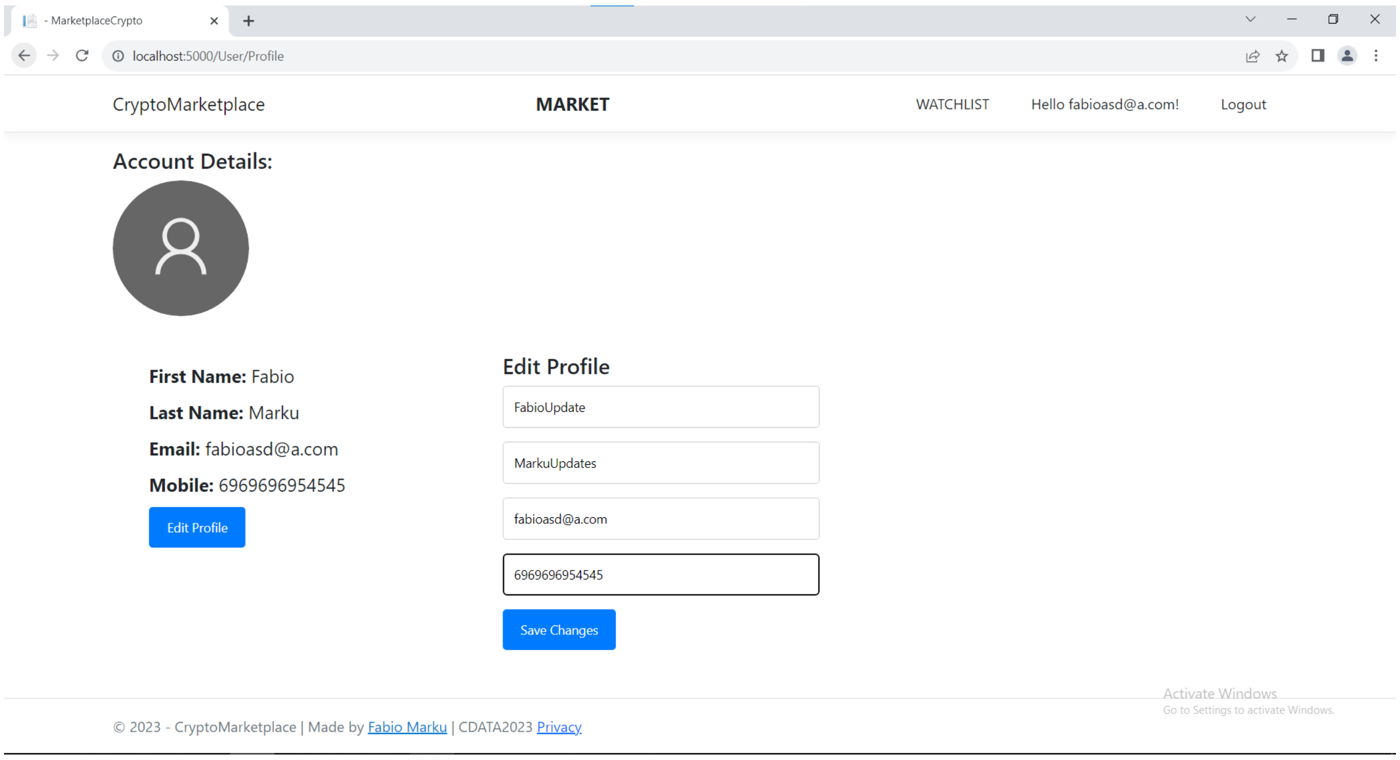




Watchlist

Symbol	Last Price	Open Price	High Price	Change for 24h	Volume	MarketCap	
ETHUSDT	1673.29000000	1663.80000000	1696.72000000	0.57 %	239606.69750000	401195255.80038300	Remove
ADAUSDT	0.26790000	0.26760000	0.27060000	0.11 %	47413108.10000000	12694070.71322000	Remove
XRPUSDT	0.53890000	0.50600000	0.55820000	6.50 %	517115828.00000000	273957837.67730000	Remove
DOTUSDT	4.48400000	4.49600000	4.55300000	-0.27 %	982078.93000000	4430568.75648000	Remove

© 2023 - CryptoMarketplace | Made by Fabio Marku | CDATA2023 Privacy



Account Details:

First Name: Fabio

Last Name: Marku

Email: fabioasd@a.com

Mobile: 6969696954545

Edit Profile

Edit Profile

FabioUpdate

MarkuUpdates

fabioasd@a.com

6969696954545

Save Changes

© 2023 - CryptoMarketplace | Made by Fabio Marku | CDATA2023 Privacy